INDICE

CAPITULO 1

INTRODUCCIÓN A SERVICIOS WEB

"Porque el Señor es bueno y su gran amor es eterno; su fidelidad permanece para siempre"

SAL 100:5

1.1 INTRODUCCIÓN

En los últimos años la mayoría de procesos de negocio han cambiado en flexibilidad, interconectividad y autonomía debido a las condiciones del mercado, a los nuevos modelos organizacionales y a los escenarios de uso de los sistemas de información.

En este contexto, Internet y el Web están cambiando la forma en la que se ofrecen los negocios y los servicios a la sociedad global, y en la que estos negocios interoperan, es así que el modelo Web ha sido adoptado más rápida y ampliamente que cualquier otra aproximación al desarrollo de aplicaciones distribuidas.

El extraordinario éxito del modelo Web puede atribuirse a una característica fundamental: es un modelo más débilmente acoplado que los modelos de programación distribuida tradicionales como RPC, DCOM y CORBA. *[www01]*

Este capítulo, pretende presentar al lector una detallada introducción a los Servicios Web, estudiando los temas necesarios como preámbulo a la llegada y aparición de ésta tan acertada tecnología.

Inicia con la definición de sistemas distribuidos, continúa con un breve estudio de modelos tradicionales de programación distribuida, presenta conceptos importantes como protocolo y middleware. Finalmente se presenta el concepto de computación distribuida y analiza la integración de aplicaciones, hasta llegar al concepto de Servicio Web.

1.2 INTRODUCCIÓN A LOS SERVICIOS WEB.

1.2.1 Sistemas Distribuidos (Conceptos Generales)

Definiciones.-

"Sistema en el cual múltiples procesadores autónomos, posiblemente de diferente tipo, están interconectados por una subred de comunicación para interactuar de una manera cooperativa en el logro de un objetivo global" *[Lelann, 1981]*

"Sistema en el cual componentes de Hardware y Software, localizadas en computadores en red, se comunican y coordinan sus acciones sólo por paso de mensajes" *[Coulouris, 2002]*

"Conjunto de computadores independientes que se muestran al usuario como un sistema único coherente". *[Tanembaum. 2001]*

Un sistema distribuido se define como un conjunto de procesadores autónomos conectados por una red; cada uno de los cuales tiene capacidad de procesamiento y de manejo de comunicaciones; que colaboran entre si para la obtención de un resultado común. Con el objetivo de obtener mejor desempeño, mayor fiabilidad y disponibilidad, compartición de recursos e información. Se establece la comunicación mediante un protocolo prefijado por un esquema cliente-servidor o n-capas". *[www01]*

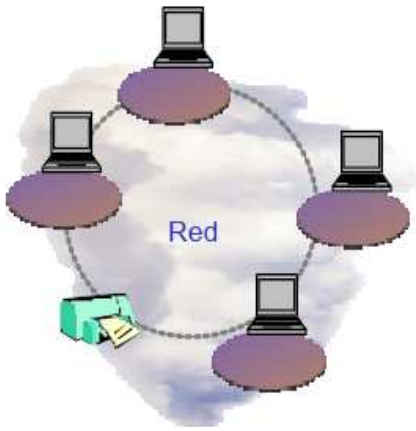


Figura 1.1. Esquema de Sistema Distribuido

Características:

Concurrencia.- Esta característica de los sistemas distribuidos permite que los recursos disponibles en la red puedan ser utilizados simultáneamente por los usuarios y/o agentes que interactúan en la red.

Carencia de reloj global.- Las coordinaciones para la transferencia de mensajes entre los diferentes componentes para la realización de una tarea, no tienen una temporización general, esta más bien distribuida a los componentes.

Fallos independientes de los componentes.- Cada componente del sistema puede fallar independientemente, con lo cual los demás pueden continuar ejecutando sus acciones. Esto permite el logro de las tareas con mayor efectividad, pues el sistema en su conjunto continua trabajando.

1.2.1.1 Sistemas de Objetos Distribuidos

La Computación de Objetos Distribuidos es la convergencia de dos paradigmas de computación que en su momento han revolucionado la manera en que se desarrollan sistemas, la Orientación a Objetos y la Arquitectura Cliente/Servidor.

1.2.1.1.1 Orientación a Objetos Distribuidos

La Orientación a Objetos permite representar la realidad de manera más natural, lo cual facilita las actividades de análisis y diseño de sistemas, así como su programación. Esto es posible mediante el uso de objetos, que son unidades de software que encierran un estado, datos, métodos, y un comportamiento, que puede manipular esos datos.

Ejemplos de objetos pueden ser: objetos físicos, elementos de interfaces gráficas de usuario, tipos de datos, y cualquier otro concepto abstracto susceptible de ser descrito como un objeto.

Además, una aplicación orientada a objetos está basada en las interacciones que estos objetos mantienen entre sí, lo cual se basa en el intercambio de mensajes. (Un mensaje es la solicitud que un objeto le hace a otro para que ejecute uno de sus métodos o funciones.)

1.2.1.1.2 Cliente/Servidor

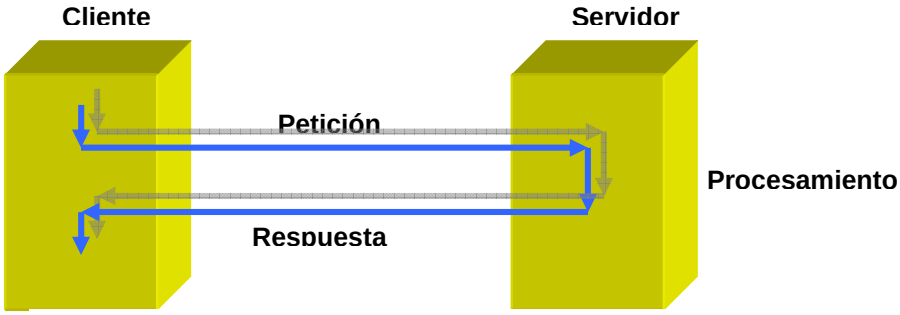


Figura 1.2 Arquitectura de Cliente Servidor de Software Distribuido

En un esquema Cliente-Servidor se denomina Cliente a la máquina que solicita un determinado servicio y Servidor a la máquina que lo proporciona. El servicio puede ser la ejecución de un determinado algoritmo, el acceso a determinado banco de información o el acceso a un dispositivo hardware. *[Monge, Raúl, 2004]*

Paso de Mensajes

Los modelos de comunicación basados en cliente servidor con paso de mensajes responden al esqueleto, como muestra la figura:

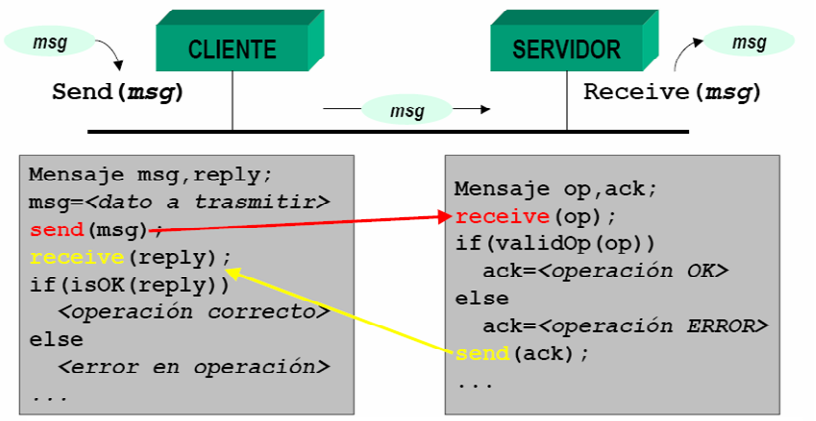


Figura 1.3 Cliente Servidor – Paso de Mensajes

Los modelos cada pareja Send /Receive transmite un mensaje entre cliente y servidor. Por lo general de forma asíncrona. Los mensajes intercambiados pueden ser: Mensajes de texto (por ejemplo: HTTP), Mensajes con formato (binarios). *[www02]*

1.2.1.2 Capas de los Sistemas Distribuidos

De manera general las capas de los sistemas distribuidos se enmarcan de esta forma:

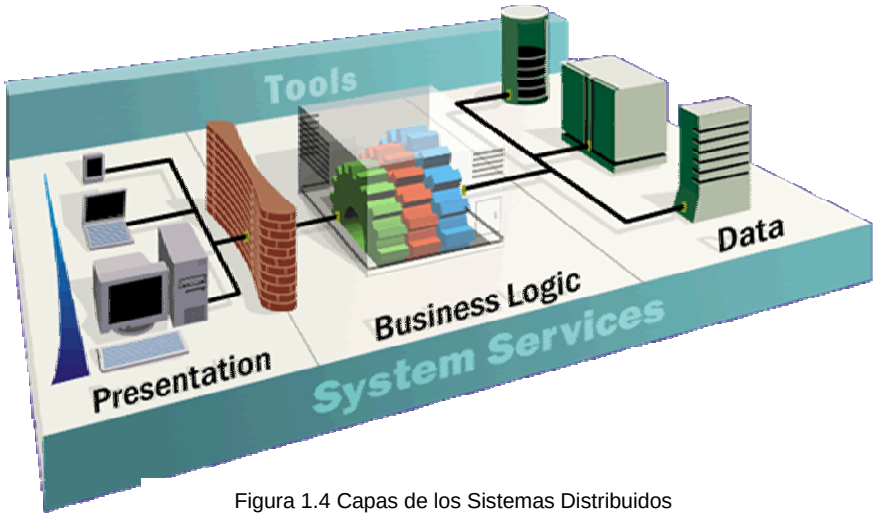


Figura 1.4 Capas de los Sistemas Distribuidos

Presentation -Presentación.- Tiene que ver con la presentación al usuario de un conjunto de objetos visuales y llevar a cabo el procesamiento de los datos producidos por él mismo y los devueltos por el servidor.

Bussiness Logic- Lógica del negocio, determina lo que hace realmente el sistema. Este cuida de que se cumplan las reglas del negocio y ejecuta los procesos del negocio. La lógica de la aplicación puede tomar muchas formas: programas, restricciones, procesos de negocio, etc.

Data - Gestión de recursos, trata con la organización (almacenamiento, indexación, y recuperación) de los datos necesarios para soportar la lógica de la aplicación. Suele ser una base de datos, pero puede ser un sistema de recuperación de texto o cualquier otro sistema de gestión de datos que ofrezca posibilidades de interrogación y persistencia.

La capa denominada Lógica del negocio puede ser dividida en varias capas más, incluyendo la necesidad de un Servidor de aplicaciones, etc; dando así origen al concepto aplicaciones de **N Capas**

1.2.1.3 Tecnologías Orientadas a Objetos Distribuidos

En los sistemas Cliente/Servidor, un objeto distribuido es aquel que esta gestionado por un servidor y sus clientes invocan sus métodos utilizando un "método de invocación remota". El cliente invoca el método mediante un mensaje al servidor que gestiona el objeto, se ejecuta el método del objeto en el servidor y el resultado se devuelve al cliente en otro mensaje. *[www01]*

Las tres tecnologías importantes y más usadas en este ámbito son:

RMI.- *Remote Method Invocation*.- Fue el primer framework para crear sistemas distribuidos de Java. El sistema de *Invocación Remota de Métodos (RMI)* de Java permite, a un objeto que se está ejecutando en una Máquina Virtual Java (VM), llamar a métodos de otro objeto que está en otra VM diferente.
Esta tecnología está asociada al lenguaje de programación Java, es decir, que permite la comunicación entre objetos creados en este lenguaje.

DCOM.- *Distributed Component Object Model*.- El *Modelo de Objeto Componente Distribuido*, esta incluido en los sistemas operativos de Microsoft. Es un juego de conceptos e interfaces de programa, en el cual los objetos de programa del cliente, pueden solicitar servicios de objetos de programa servidores en otros ordenadores dentro de una red.
Esta tecnología esta asociada a la plataforma de productos Microsoft.

CORBA.- *Common Object Request Broker Architecture*.- Tecnología introducida por el Grupo de Administración de Objetos OMG, creada para establecer una plataforma para la gestión de objetos remotos independiente del lenguaje de programación.
[www01]

1.2.2 PROTOCOLO

Definición:

Es un conjunto bien conocido de reglas y formatos que se utilizan para la comunicación entre procesos que realizan una determinada tarea. Se requieren dos partes:

- Especificación de la secuencia de mensajes que se han de intercambiar.
- Especificación del formato de los datos en los mensajes.

Un protocolo permite que componentes heterogéneos de sistemas distribuidos puedan desarrollarse independientemente, y por medio de módulos de software que componen el protocolo, haya una comunicación transparente entre ambos componentes. Es conveniente mencionar que estos componentes del protocolo deben estar tanto en el receptor como en el emisor.

Protocolos usados en los sistemas distribuidos:

IP: Protocolo de Internet.- Protocolo de la capa de Red, que permite definir la unidad básica de transferencia de datos y se encarga del direccionamiento de la información, para que llegue a su destino en la red.

TCP: Protocolo de Control de Transmisión.- Protocolo de la capa de Transporte, que permite dividir y ordenar la información a transportar en paquetes de menor tamaño para su transporte y recepción.

HTTP: Protocolo de Transferencia de Hipertexto.- Protocolo de la capa de aplicación, que permite el servicio de transferencia de páginas de hipertexto entre el cliente WEB y los servidores.

SMTP: Protocolo de Transferencia de Correo Simple.- Protocolo de la capa de aplicación, que permite el envío de correo electrónico por la red.

POP3: Protocolo de Oficina de Correo.- Protocolo de la capa de aplicación, que permite la gestión de correos en Internet, es decir, le permite a una estación de trabajo recuperar los correos que están almacenados en el servidor.

1.2.2.1 RPC

RPC (*Remote Procedure Call*, *Llamada a Procedimiento Remoto*). Es un protocolo (*conjunto de reglas que especifican el intercambio de datos*) que permite a un programa de ordenador ejecutar código en otra máquina remota, sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De

esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro del paradigma *cliente-servidor*. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación al cliente.

Hay distintos tipos de RPC, muchos de ellos estandarizados como pueden ser el RPC de Sun (RFC 1057), Distributed Computing Environment (DCE), DCOM de Microsoft. No siendo compatibles entre sí. La mayoría de ellos utilizan un lenguaje de descripción de interfaz (IDL Interface Definition Language) que define los métodos exportados por el servidor.

Hoy en día se está utilizando el XML como lenguaje para definir el IDL y el HTTP como protocolo de red. Dando lugar a lo que se conoce como servicios web. Ejemplos de éstos pueden ser SOAP o xml-rpc. *[www03]*

1.2.3 MIDDLEWARE

Definición:

Capa de software intermedio entre el cliente y el servidor. Es la capa de software que nos permiten gestionar los mecanismos de comunicaciones. Por ejemplo, si se hace la petición de una página web desde un browser en el cliente, el middleware determina la ubicación y envía una petición para dicha página. El servidor Web, interpreta la petición y envía la página al software intermedio, quien la dirige al navegador de la máquina cliente que la solicitó.

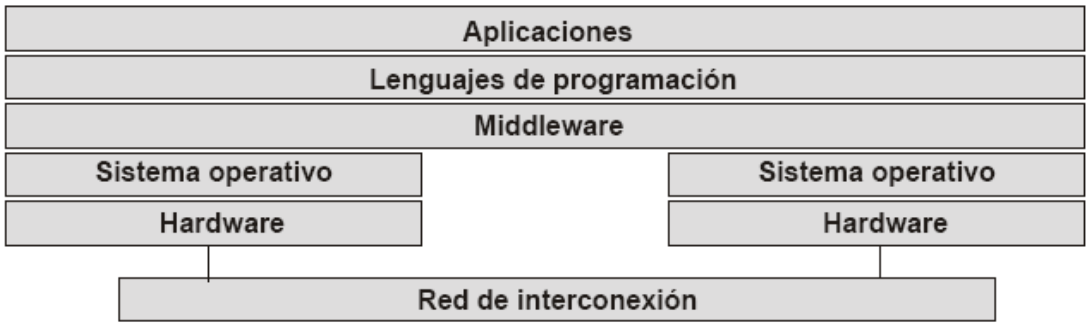


Figura 1.5 Middleware

Existen dos tipos:

- **Software intermedio general**. Servicios generales que requieren todos los clientes y servidores, por ejemplo: software para las comunicaciones usando el TCP/IP, software parte del sistema operativo que, por ejemplo, almacena los archivos distribuidos, software de autenticación, el software intermedio de mensajes de clientes a servidores y viceversa.
- **Software intermedio de servicios**. Software asociado a un servicio en particular, por ejemplo: software que permite a dos BD conectarse a una red cliente/servidor (ODBC: Conectividad abierta de BD), software de objetos distribuidos, por ejemplo la tecnología CORBA permite que objetos distribuidos creados en distintos lenguajes coexistan en una misma red (intercambien mensajes), software intermedio para software de grupo, software intermedio asociado a productos de seguridad específicas (Conexiones Seguras: Sockets), etc.

Características:

- Independiza el servicio de su implantación, del sistema operativo y de los protocolos de comunicaciones.
- Permite la convivencia de distintos servicios en un mismo sistema.
- Permite la transparencia en el sistema.
- Modelo tradicional: Monitor de teleproceso o CICS, Tuxedo, Encina.
- Modelo OO: CORBA.

1.2.4 Computación distribuida

Definición

La *computación distribuida* es una tecnología que permite a un número indeterminado de máquinas trabajar con fragmentos de información. La ventaja de este sistema es clara, ya que en vez de tener unos pocos ordenadores dedicados por entero a un proyecto, pueden tenerse infinidad de equipos dedicándose parcialmente a ello. Esto conlleva un ahorro económico impresionante, y una multiplicación de los recursos.

Es así que, si en un proyecto solo pueden trabajar dos ordenadores al 100%, con la computación distribuida podrían trabajar 300 equipos al 1%. O más. *[www04]*

Entendemos a la computación distribuida en un sentido muy amplio, incluyendo a cualquier sistema en el que múltiples agentes autónomos, cada uno con capacidades de computo individual, se comunican entre si y afectan mutuamente su comportamiento. Los agentes, usualmente llamados procesadores, procesos o nodos, pueden ser desde computadoras completas hasta autómatas celulares con capacidad de cómputo y memoria muy limitados. Se pueden comunicar mediante mensajes o memoria compartida. *[www05]*

La computación distribuida o informática en rejilla, es un nuevo modelo para resolver problemas de computación masiva utilizando un gran número de computadoras organizadas en racimos incrustados en una infraestructura de telecomunicaciones distribuida.

La informática en rejilla consiste en compartir recursos heterogéneos (basadas en distintas plataformas, lenguajes de programación, arquitecturas de equipos y programas), situados en distintos lugares y pertenecientes a diferentes dominios de administración sobre una red que utiliza estándares abiertos.

Dicho brevemente, consiste en virtualizar los recursos informáticos. *[www06]*

1.2.5 Integración de Aplicaciones - Conectando Aplicaciones en la Web

La mayoría de los departamentos de las empresas cuenta con sistemas informáticos para la realización de su trabajo. Las aplicaciones evolucionan con el tiempo pero lo hacen de forma independiente, sin las relaciones adecuadas para abarcar todo el proceso productivo. Con la integración de aplicaciones éstas son capaces de reutilizar los datos y la funcionalidad de las demás. *[www07]*

La integración de aplicaciones y datos es un proceso que implica la fusión de datos y funcionalidades de una aplicación con los de otra. Uno de los desafíos más frecuentes a los que se enfrenta una empresa es la integración de una aplicación heredada con otra nueva. Las bases de datos y archivos heredados muchas veces no disponen de una interfaz estándar de acceso a los datos o no están preparados para intercambiar fácilmente datos con bases de datos relacionales.

Figura 1.6 Integración de aplicaciones

La integración de aplicaciones consiste en coordinar múltiples aplicaciones que han sido desarrolladas de manera independiente, posiblemente empleando tecnologías no compatibles y que por lo tanto no se gestionan de forma coordinada.

Existen ciertas dificultades muy comunes en diversas empresas, que deben ser superadas para racionalizar y mejorar la gestión de los procesos de negocio, que deben ser tomados en cuenta como motivos para la integración mediante la utilización de determinada tecnología, entre los que podemos destacar:

1) Sistemas heterogéneos deben intervenir en un mismo proceso. Estos sistemas no están suficientemente integrados, lo que no hace posible la comunicación entre los mismos. La falta de integración introduce ineficiencias, como por ejemplo la necesidad de reintroducir datos manualmente de un sistema a otro, o la necesidad de tareas manuales que, de otro modo, podrían ser automatizadas.

2) La gestión del proceso es (parcialmente) manual; los actores deben conocer el proceso y, manualmente, utilizar los sistemas necesarios para llevar a cabo las distintas actividades del proceso. En último término, el control del proceso recae en los actores que intervienen en el mismo, lo que aumenta la posibilidad de errores y dificulta su gestión.
3) No existen datos suficientes sobre la realización de los procesos, como por ejemplo tiempos de realización de cada actividad, sistemas involucrados, requisitos de disponibilidad, etc. Esto resulta en una monitorización insuficiente de los procesos, lo que dificulta la detección de problemas e ineficiencias.

4) Los sistemas de la entidad no son suficientemente visibles; no hay información explícita sobre la función de los mismos, dependencias mutuas, e intervención de estos sistemas en los procesos de la entidad.

5) La comunicación entre los niveles de negocio y de tecnología de la entidad presenta ciertas dificultades en lo que respecta a la comunicación mutua de requerimientos de negocio y disponibilidad de tecnología. Esto dificulta la puesta en marcha de nuevos procesos y la mejora de procesos existentes que involucren sistemas de la entidad.
[www08]

En este contexto, señalo tecnologías como EAI y arquitecturas como SOA que pueden contribuir en alguna medida a salvar las dificultades mencionadas y a lograr el objetivo de una gestión de procesos eficaz.

1.2.5.1 Integración de Aplicaciones Corporativas (EAI)

La EAI persigue el permitir compartir, sin ninguna restricción, los datos y procesos entre aplicaciones y fuentes de datos en una empresa.

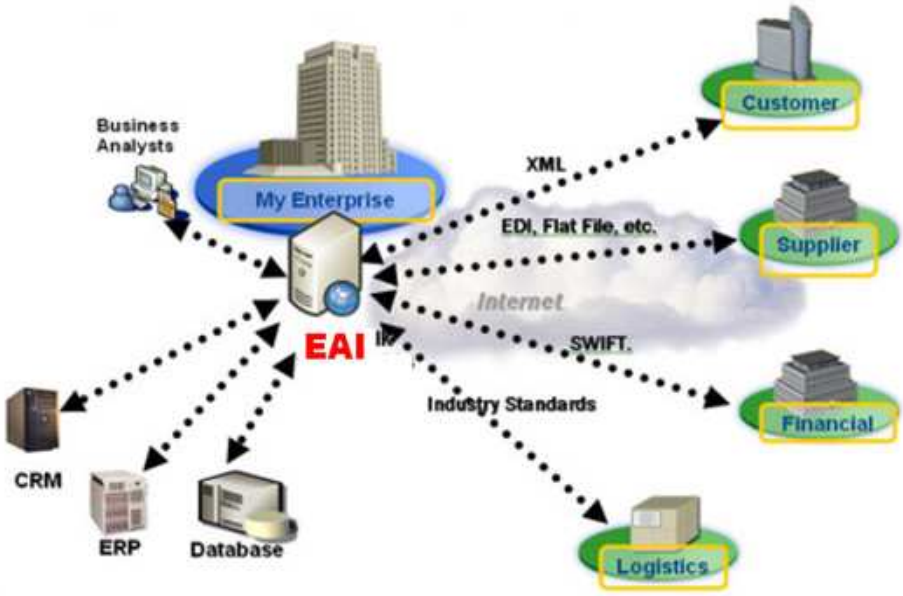


Figura 1.7 EAI Integración de Aplicaciones

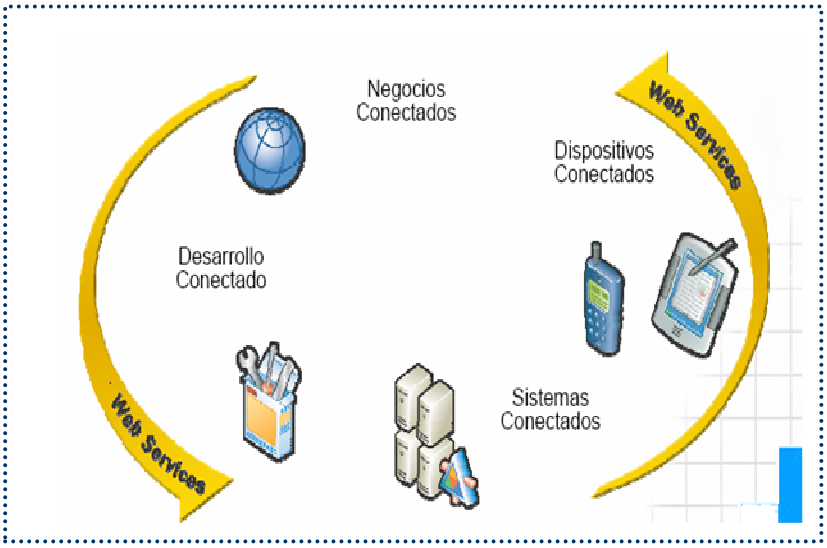
Ya que cualquier empresa razonablemente grande dispondrá de varias aplicaciones, probablemente escritas usando tecnologías diferentes. El coste de mantener dichos sistemas por separado es muy grande y probablemente, los beneficios de hacerlo así disminuirán con el tiempo.

EAI (Enterprise Application Integration), son un paso en la evolución de los middleware abordando aspectos de integración.
En arquitecturas de 3-niveles se facilita la integración de gestores de recursos diferentes, desarrollando la lógica de la nueva aplicación en el middleware. La funcionalidad resultante puede ser expuesta como un nuevo servicio, que puede ser integrado por servicios de más alto nivel, y así sucesivamente.

Por lo tanto Web Services, se considera una tecnología fundamental para dominar y manejar la complejidad y heterogeneidad de los Sistemas de Información Empresariales. Web Services ofrece una forma estándar de comunicación e interoperabilidad entre aplicaciones. Permite a las aplicaciones exponer sus servicios en la Web, independientemente del lenguaje o la plataforma en la cual fueron desarrollados. Web Services permite reutilizar activos de información al ofrecer al desarrollador métodos estándar para acceder e integrar servicios de las aplicaciones. *[www09]*

CAPITULO 2

SERVICIOS WEB



"Cantaré al Señor toda mi vida; cantaré a mi Dios mientras tenga aliento"
SAL 104:33

2.1 INTRODUCCIÓN

Las necesidades y aplicaciones de negocios del mundo de hoy requieren de un conjunto de componentes distribuidos cada vez más complejos a través de las redes de telecomunicaciones. Aunque la complejidad estriba en el software/hardware que ponen a funcionar a esos componentes, todo esto va enfocado a facilitar a los usuarios sus diversas actividades diarias.

Sin que seamos conscientes de ello, en Internet navegan dos tipos de entes muy distintos: las personas que visitan páginas web o acceden a servicios interactivos, y las aplicaciones distribuidas. En la web hay miles de programas que conversan entre sí, intercambiándose datos de forma automática, sin mediación humana.

Usando servicios web, un programador puede implementar aplicaciones basándose en rutinas y datos proporcionados desde un servidor distante.

Así, por ejemplo, existen servidores que proporcionan rutinas que permiten conocer la previsión meteorológica de una localidad o las cotizaciones en bolsa de una empresa, etc. Esas rutinas pueden ser usadas, por ejemplo, para simplemente mostrar información en una página web, o pueden ser usadas como los datos de entrada en un programa de predicción o de ayuda a la toma de decisiones. Si el acceso a dichas rutinas y a los datos que generan se hace usando ciertos protocolos estandarizados, entonces es cuando hablamos de **servicio web**. *[www10]*

De esta manera inicia el estudio a los servicios web topando temas como su surgimiento, conceptos, ciclo de vida, arquitectura, ventajas, etc.

2.2 SURGIMIENTO DE LOS SERVICIOS WEB

La década de los 80's fue marcada por el surgimiento de la PC y de la interfase grafica. En la década de los 90's Internet permitió conectar computadoras en una escala global. En principio la conexión fue entre PCs y servidores por medio del explorador de Internet. A comienzos de este siglo es clara la necesidad de permitir a las computadoras conectadas a Internet comunicarse entre ellas.

Desde entonces se va dando forma al nuevo modelo de computación distribuida llamado servicios Web basados en XML. El objetivo es permitir comunicarse entre sí a sistemas heterogéneos dentro y fuera de una empresa. Esta comunicación es independiente del sistema operativo, lenguaje o modelo de programación.

La simplicidad de las interacciones en el modelo de programación Web posibilita construir sistemas incrementalmente. A diferencia del acoplamiento fuerte de RPC y de los sistemas de objetos distribuidos, que requieren la implantación de todas las piezas de una aplicación de una vez, podemos añadir tantos clientes y servidores a sistemas basados en Web como necesitemos. Podemos establecer fácilmente conexiones a aplicaciones nuevas de un modo descentralizado, sin ninguna coordinación central más allá del registro de nombres DNS, y con un grado de interoperabilidad, escalabilidad y capacidad de gestión extraordinariamente alto. *[www11]*

La siguiente figura 2.1 nos muestra el comportamiento de las arquitecturas durante su evolución.

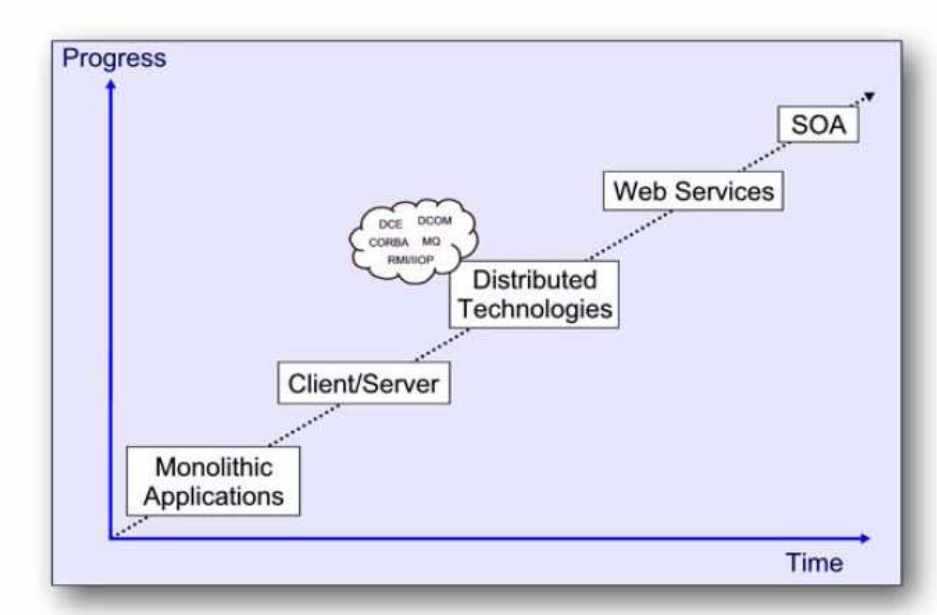


Figura 2.1 Evolución de la Arquitectura de software

2.3 SERVICIO WEB

Definiciones.-

"Aplicaciones de negocio modulares y autocontenidas que tienen interfaces abiertos, orientados a Internet y basados en interfaces estándares".

Consorcio UDDI [www14].

"Una aplicación software identificada por un URI, cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML. Un servicio Web soporta interacciones directas con otros agentes software utilizando mensajes XML intercambiados mediante protocolos basados en Internet".

W3C[www13].

"Una forma estándar de integrar aplicaciones basadas en Web utilizando los estándares abiertos XML, SOAP, WSDL y UDDI. XML se utiliza para etiquetar los datos, SOAP para transferir los datos, WSDL para describir los servicios disponibles y UDDI para listar que servicios están disponibles".

Webopedia [www17]

Los servicios web (SW) son colecciones de funciones u objetos distribuidos que son presentados como una sola entidad la cual es anunciada en la red para ser usada por otros programas. Con un interfaz definido y conocido, al que se puede acceder a través de Internet, al igual que una página web está definida por un URL *(Uniform Resource Locator)*, un servicio web está definido por un URI *(Uniform Resource Identification)* y por su interfaz, a través del cual se puede acceder a él.

El concepto de servicio web es un paradigma de la **computación distribuida** que consiste en un conjunto de protocolos de comunicación que permiten el intercambio de datos entre aplicaciones remotas. *[www11]*

Objetivo.-

Su objetivo es proporcionar interoperatividad sobre una plataforma Internet entre toda clase de aplicaciones y sistemas que se integran de forma transparente siempre que se acojan a los estándares propuestos.

Permiten la interconexión de plataformas heterogéneas de forma transparente y desconociéndose mutuamente. Es una arquitectura ideal para la integración de procesos de negocio de empresas con sus clientes y proveedores.

Todo ello queda reflejado en la definición de Servicio WEB del Stencil:

> Los Servicios WEB son componentes de software reutilizables, ligeramente acoplados que semánticamente encapsulan funcionalidades discretas que son distribuidas y accesibles a nivel de programación a través de protocolos de Internet.

El concepto "ligeramente acoplado" hace referencia a que en las aplicaciones "clásicas" es muy difícil sustituir en componente por otro. Es, además, el concepto de servicio básico para diseñar las aplicaciones distribuidas. Observe también que "encapsular" funcionalidades discretas no es más que otra definición de servicio*. [www13]*

Figura 2.2 Servicios Web, Servidores y Clientes

A diferencia de los modelos tradicionales cliente/servidor, los servicios web no proveen al usuario con una interfase gráfica. Los servicios web en vez de eso, comparten lógica de negocios, procesos y datos a través de una interfase programática a través de la red. Los servicios web permiten que diferentes aplicaciones de diferentes fuentes puedan comunicarse entre sí.

> El consorcio de tecnologías web, W3C (http://www.w3c.org/), define a los servicios web como un sistema de software identificado por un URL, cuyas interfaces públicas y enlaces

están definidos y descritos usando XML. Su definición puede ser descubierta por otros sistemas de software. Estos sistemas pueden interactuar con el servicio Web de la manera preestablecida en su definición, usando mensajes basados en XML y transportados por los protocolos de Internet.

Para simplificar la definición, podemos usar la analogía de un servicio web con una biblioteca de funciones que un proveedor ofrece. Esta biblioteca por sí sola no genera un valor completo, se utiliza en conjunto con otros componentes para crear aplicaciones completas. Siguiendo con la analogía, si se quisiera crear un software ERP, entonces podríamos contratar con diferentes proveedores un SW de manejo de inventario, un SW de control de producción, un SW de planeación de compras; los interconectamos y listo, ya tenemos nuestro ERP. Claro, para que esto funcione deben existir estándares que nos permitan asegurar las operaciones mínimas que deben proveer los SW anteriormente mencionados.

Los servicios web publican una interfaz con todos los servicios que ofrecen, y cuando son invocados por los clientes, se ejecutan en su propia infraestructura (CPUs, DBMS, etc.), retornando sólo el resultado del procesamiento, es por esto que las aplicaciones construidas con servicios web caen dentro de la definición de aplicaciones distribuidas.

[www18].

Los servicios web ofrecen un medio de intercambio de datos entre distintos dispositivos, lo cual los hace interesantes para dar soporte a multitud de aplicaciones distribuidas, como son las relacionadas con la telemonitorización, teleasistencia o, en general, con la telemedicina.

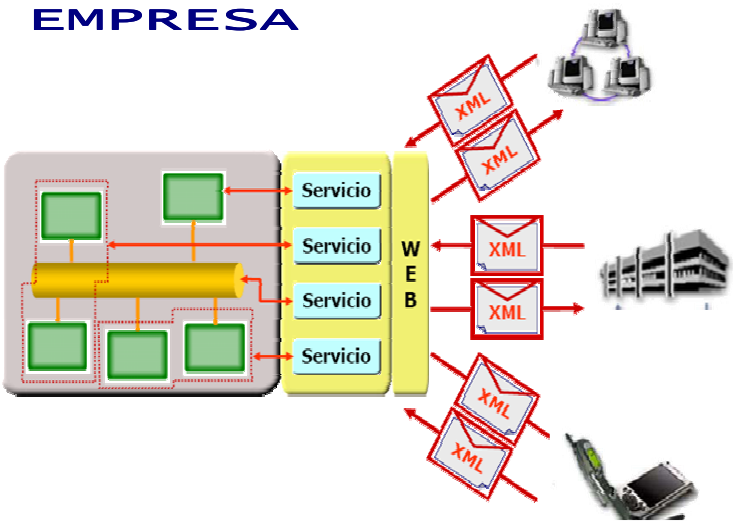


Figura 2.3 Servicios Web interactúan con diversos dispositivos

2.4 ARQUITECTURA FUNCIONAL DE LOS SERVICIOS WEB

La arquitectura se basa en tres tipologías de servicios como se muestran en la figura:

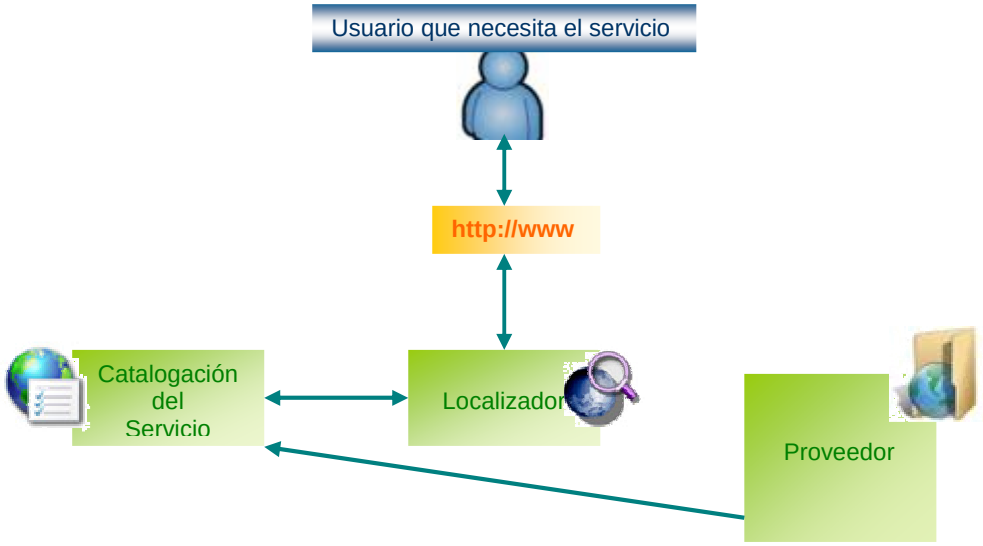


Figura 2.4 Arquitectura funcional de un Servicio Web

1. Servicios de Catalogación.

Sirven al proveedor para publicar un servicio en la red. Los aporta la Agencia.

2. Servicios de Localización.

Sirven al usuario para localizar funcionalmente el servicio que necesita.

La localización y descubrimiento del servicio puede ser:

- Estática, navegando el futuro cliente.
- Dinámica en tiempo de diseño o ejecución utilizando un servicio UDDI.

3. Servicios de Utilización

Una vez escocido el servicio y encontrado el proveedor, permiten pedir e instanciar el objeto que debe proporcionar el servicio.

2.5 ESTÁNDARES DE LOS SERVICIOS WEB

El consorcio de Internet http://www.w3c.org se encargó de crear y mantener estos estándares. Que permiten hacer uso de los Servicios Web basados en XML. *[www12].*

A continuación señalo brevemente estos protocolos que serán mencionados mas adelante en el documento.

- ***XML:*** (Lenguaje de Marcado eXtensible) Es un formato universal para representar los datos. XML-RPC: son protocolos sobre los que se establece el intercambio.
  Los Servicios Web **se basan en XML** para estructurar la información, lo que permite:

  - Homogeneidad para facilitar la comprensión de las máquinas
  - Diferentes plataformas / marcos de trabajo

- ***WSDL:*** (Lenguaje de Descripción de Servicios Web) Lenguaje por medio del cual un servicio Web describe entre otras cosas qué hace o qué funcionalidad implementa. Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.

- ***SOAP***: (Protocolo Simple de Acceso a Objetos) Es un protocolo que permite mover los datos entre aplicaciones y sistemas. Es el mecanismo por medio del cual los servicios Web son invocados e interactúan.

- ***UDDI***: (Descubrimiento, Descripción e Integración Universal) Lenguaje que permite publicar, encontrar y usar los Servicios Web basados en XML. Es la 'Página Amarilla' de los servicios Web, es decir un directorio para poder encontrarlos. Puede ser accedido con un explorador en http://www.uddi.org o programáticamente.

- ***WS-Security:*** Protocolo de seguridad aceptado como estándar por OASIS. Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados...

En la figura siguiente se puede observar como interactúan los protocolos anteriormente mencionados en una tecnología de Servicios Web.

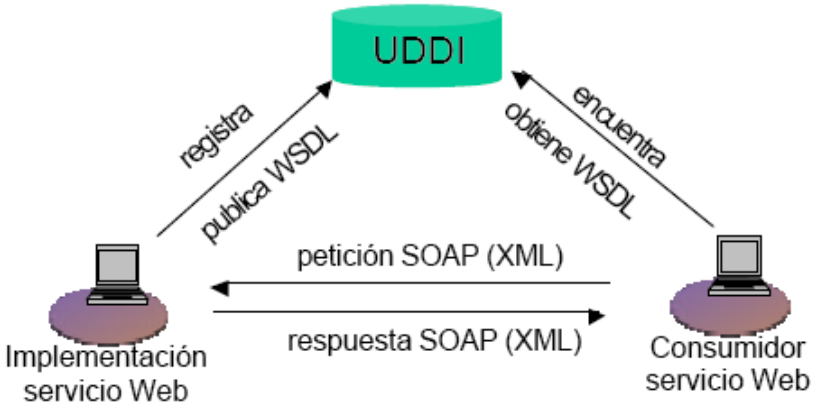


Figura 2.5 Vocabularios XML

2.6 CICLO DE VIDA DE LOS SERVICIOS WEB

El ciclo de vida de los Servicios consiste en los siguientes 6 pasos importantes, como muestra la figura.

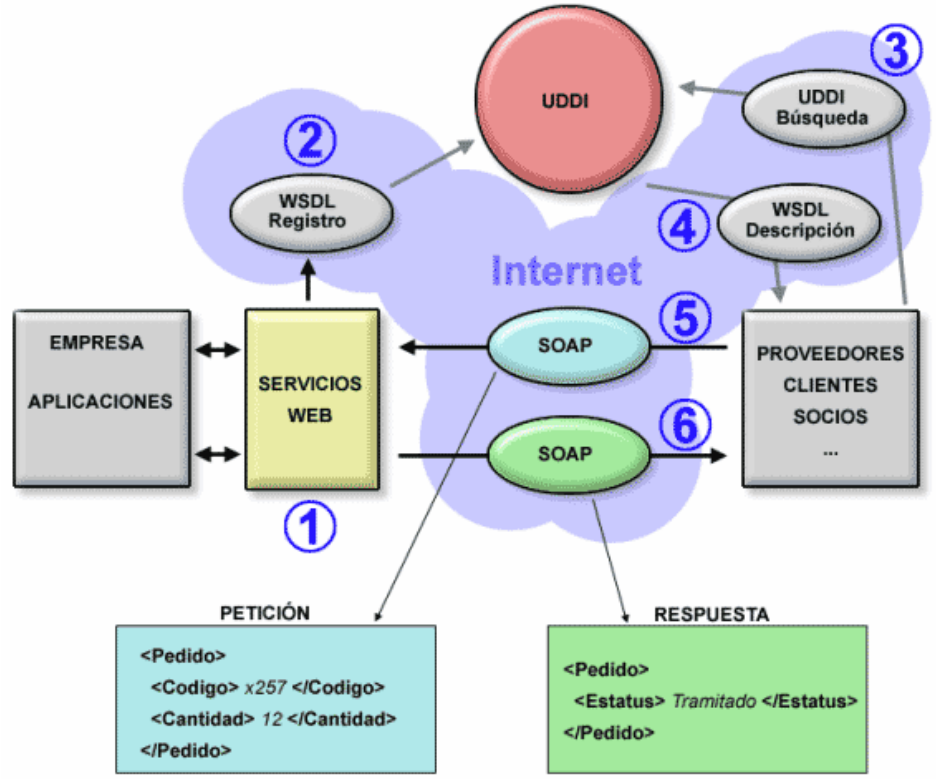


Figura 2.6 Ciclo de vida de un Servicio Web

(1) El ciclo se origina cuando las empresas se deciden a desarrollar y exponer la funcionalidad de sus aplicaciones en forma de Servicio Web.

(2) Una vez que los Servicios Web se han desarrollado, deben ser registrados en un nodo UDDI para poder ser localizado por los potenciales usuarios. En dicho registro se aportaran datos sobre la empresa, los Servicios Web que se ofrecen etc. y también la descripción de las interfaces de uso de cada Servicio Web (WSDL). Cuando algún consumidor solicite dicho Servicio Web, el servidor UDDI le redirigirá a la URI proporcionada por el fabricante.

(3) Los posibles consumidores (proveedores, clientes, socios...) se conectan al servidor UDDI para buscar los Servicios Web que les interesan.

(4) Una vez que encuentran el Servicio Web que desean, obtienen la descripción de sus interfaces de uso (WSDL).

(5) Gracias a la descripción de las interfaces de uso, los consumidores son capaces de elaborar paquetes SOAP para comunicarse con el proveedor del Servicio Web.

(6) El proveedor del Servicio Web elabora un paquete SOAP como respuesta a la petición del consumidor del Servicio Web. *[www14].*

Analizando profundamente, para hacer realidad esta tecnología, se requiere de tres entidades participantes:

- Proveedores de Servicios,
- Agentes y
- Solicitantes.

El Proveedor anuncia sus servicios con un **Agente**, cuando un **Solicitante** busca en un Agente un servicio, encuentra al Proveedor y establece el enlace para hacer uso de los servicios, como muestra la figura siguiente:

Figura 2.7 Servicios Web

El **Proveedor** construye el **Servicio** con el lenguaje y el Middleware necesario.
Define la **Descripción del Servicio** que incluye, con un documento escrito con Servicios WEB Description Language (WDSL):

- Las prestaciones.
- La utilización del servicio por terceros.
- La localización

Publica la oferta del servicio en las páginas amarillas del Universal Description, Discovrey and Integration (UDDI). El fabricante también puede encontrar aquí otros servicios ya creados que le faciliten su trabajo. La **Agenda** UDDI fue creada en septiembre de 2000 por IBM, Ariba y Microsoft y posteriormente se sumaron otros actores como Compaq y SAP.

El usuario final, conocido como el **solicitante**, **localiza** y **enlaza** el servicio WEB a través de SOAP (Simple Object Access Protocol) mediante un mecanismo de tipo RPC sobre el protocolo HTTP y un intercambio de mensajes XML.

Se baja un applet con el **objeto** que constituye el **Servidor Web,** y que una vez **instanciado** en la plataforma solicitante, le va a conectar con el proveedor. A partir de ese momento, puede empezar a **interactuar** con el servicio, **desinstanciando** el objeto y **desconectándose** cuando ha acabado.
Los Web Servicies **no disponen de interfase gráfica** ya que no se han creados para dialogar directamente con los usuarios. Son utilizados como servidores por otros componentes de software del sistema distribuido.

El objetivo final de los servicios web es la creación de directorios en línea que puedan ser localizados de un modo sencillo con un alto nivel de fiabilidad.

XML es utilizado para etiquetar los datos, SOAP es usado para transferir los datos, WDSL es utilizado para describir los servicios disponibles y UDDI es usado para listar qué servicios están disponibles. *[www15].*

2.7 ARQUITECTURA DE LOS SERVICIOS WEB

Los servicios WEB se basan en una arquitectura de **objetos distribuidos** en Internet bajo arquitectura C/S, que se comunican por parámetros XML y utilizando como transportista los protocolos estándar de Internet. El concepto en si es el servicio de siempre, pero en un mercado de ámbito y accesibilidad mundial y con un coste de distribución bajo.
Por debajo hay una arquitectura de delegación y distribución de servicios, en que un servicio invocado puede llamar a otros de forma transparente al peticionario. *[www11]*

La World Wide Web no es sólo un espacio de información, también es un espacio de interacción. Utilizando la Web como plataforma, los usuarios, de forma remota, pueden solicitar un servicio que algún proveedor ofrezca en la red.

Pero para que esta interacción funcione, deben existir unos mecanismos de comunicación estándares entre diferentes aplicaciones. Estos mecanismos deben poder interactuar entre sí para presentar la información de forma dinámica al usuario.

Se precisa, pues, una arquitectura de referencia estándar que haga posible la interoperabilidad y extensibilidad entre las distintas aplicaciones y que permita su combinación para realizar operaciones complejas. *[www13].*

Esta arquitectura se basa en los siguientes componentes:

2.7.1 Diseñar un marco de mensajería

- **Simple SOAP**: Simple Object Access Protocol permite intercambiar información estructurada en un ambiente descentralizado y distribuido. "Messaging Framework" define, usando tecnologías XML, un marco extensible de mensajería que contiene una construcción del mensaje que se pueda intercambiar con una variedad de protocolos subyacentes. *[www19].*

- **Web Services Addressing (WS-Addressing):** Direccionamiento de Servicios Web. La dirección de los servicios Web proporciona mecanismos neutrales para transportar los servicios web y los mensajes. Define un sistema de características abstractas y una representación de XML para referirse a servicios de la Web y para facilitar la dirección final de los mensajes. Esta especificación permite a los sistemas de mensajería soportar la transmisión del mensaje a través de redes que incluyen el procesado de nodos tales como gestión final, cortafuegos y pasarelas mediante una forma de transporte neutro. *[www20].*

- **SOAP Message Transmission Optimization (MTOM)** Descripción de la Optimización de la Transmisión del Mensaje. Describe una característica abstracta y una puesta en práctica concreta para optimizar el formato de la transmisión y/o de la vía de los mensajes SOAP. *[www21]*

2.7.2 Descripción de los Servicios

- **Web Services Description Language (WSDL)**: Lenguaje de Descripción de los Servicios Web. La especificación define el lenguaje básico que puede usarse para describir servicios Web basados en un modelo abstracto de lo que ofrece el

servicio. También define los criterios de conformidad de los documentos en relación a este lenguaje. *[www22].*

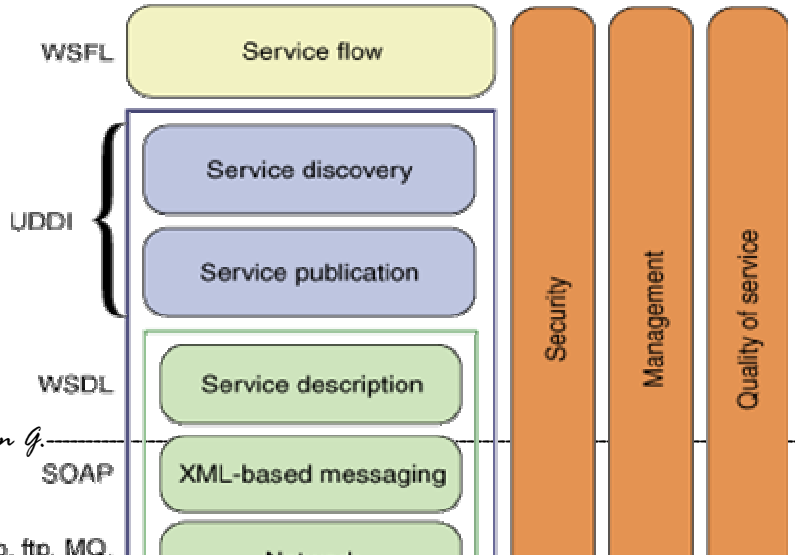
- **Web Services Choreography Description Language (WS-CDL)**: Lenguaje de Descripción de la Coreografía de los Servicios Web. Es un lenguaje basado en XML que describe colaboraciones *peer to peer* de los participantes definiendo, desde un punto de vista global, un comportamiento observable común y complementario; donde ordenado el mensaje, intercambia el resultado de acuerdo a un objetivo de negocios común. *[www23]*

Los servicios web que se basan en XML permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado dichas aplicaciones e independientemente del sistema operativo o plataforma en que se ejecuten y de los dispositivos utilizados en el acceso.

2.7.3 Capas de los servicios Web

Los servicios web se componen de varias capas entre las que destacan:

- *Servicios de transporte* (constituidos por los protocolos del nivel más bajo, que codifican la información independientemente de su formato, y que pueden ser comunes a otros servicios),
- De *mensajería*,
- De *descripción* y
- De *descubrimiento*.



En la capa inferior se encuentran los servicios de transporte, que son los encargados de establecer la conexión y el puerto utilizado. Lo más común es emplear el protocolo de hipertexto HTTP, pero también se pueden usar otros protocolos como SMTP (Simple Mail Transfer Protocol o Protocolo de Transmisión de Correo Simple que es el protocolo que nos permite recibir correos electrónicos), o el protocolo FTP (File Transfer Protocol).

En la capa siguiente se encuentran los servicios de mensajería que especifican cómo se tiene que codificar el mensaje que contiene los datos que se intercambian entre el ordenador cliente y el ordenador servidor. Como se ha afirmado, el protocolo más utilizado en esta capa es SOAP que permite utilizar cualquiera de los protocolos de transporte antes mencionados y que utiliza el lenguaje XML para especificar los mensajes.

Por su parte, la función del lenguaje **WSDL (Web Service Description Language)** es decirle a una aplicación qué formato usar para comunicarse, especificando por medio de un lenguaje estándar, tanto la dirección del servicio como la interfaz que se va a utilizar.

- Un documento WSDL usa los siguientes elementos en la definición de servicios en red:
- Tipos (*Types*): un contenedor para definiciones del tipo de datos que usan algunos tipos de sistemas (tal como XSD).
- Mensaje (*Message*): una definición abstracta tipo del dato que está siendo comunicado.
- Operación (*Operation*): una descripción abstracta de una acción soportada por el servicio.
- Tipo de puerto (*Port Type*): un conjunto abstracto de operaciones soportadas por uno o más puntos finales.
- Conexión (Binding): un protocolo concreto y una especificación de formato de datos para un tipo de puerto particular.

- Puerto (*Port*): un punto final individual definido como una combinación de una conexióny una dirección de la red.
- Servicio (*Service*): una colección de puntos finales relacionados. *[lib01].*
-

Por último, en la capa superior se encuentra ***UDDI (Universal Description, Discovery and Integration)***. Provee un mecanismo para que los clientes encuentren de forma dinámica otros servicios web creando una plataforma interoperable estándar que permite a las compañías usar de forma rápida, fácil y dinámica los servicios Web. Usando la interfaz de UDDI, pueden conectarse dinámicamente la empresas con los servicios proporcionados por socios externos.

Para ello es necesario registrarse en UDDI y los registros pueden tener diversos propósitos y usarse en distintos contextos. Existen 2 tipos de clientes:

- Compañías que desean publicar un servicio (y su interfaz de uso) y

- Clientes que desean obtener cierta clase de servicios por medio de una conexión.

UDDI se monta sobre SOAP y asume que las consultas y las respuestas son objetos de UDDI enviados como mensajes de SOAP.

El lenguaje WSFL o Web Services Flow Language es un lenguaje XML para describir la composición de los servicios web como parte de una definición del proceso de negocio. Fue diseñado por IBM como parte de un marco tecnológico de servicios web y para completar las especificaciones existentes. WSDL considera 2 tipos de servicios web: el primer tipo especifica un proceso de negocio ejecutable conocido como Modelo de flujo (flowModel) y el segundo tipo es un negocio en colaboración conocido como Modelo global (globalModel). *[www24].*

2.8 FUNCIONALIDAD DE LOS SERVICIOS WEB

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario.

Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

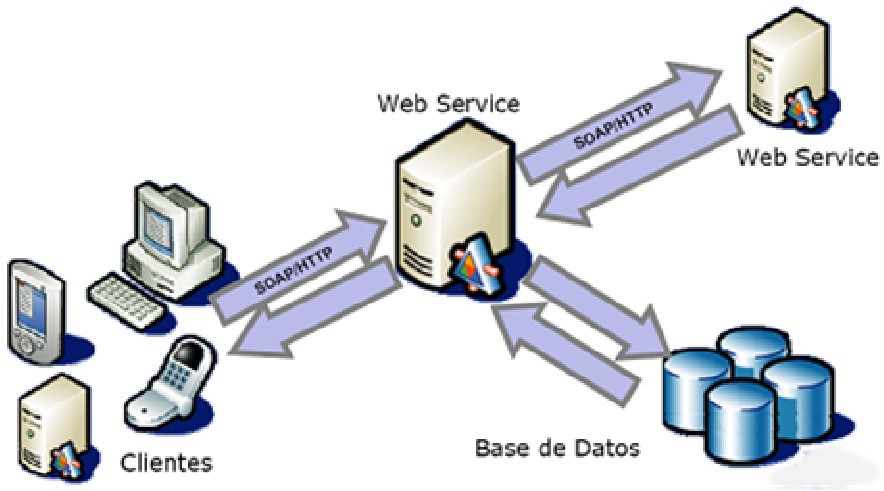


Figura 2.8 Funcionalidad de los Servicios Web

2.9 VENTAJAS DE LOS SERVICIOS WEB

✓Los Servicios Web son multiplataforma, es decir, son independientes tanto de la arquitectura (pueden comunicar ordenadores, PDAs, teléfonos móviles, estaciones de trabajo, etc) como del Sistema Operativo y el lenguaje de programación que se use, ya que se basan en XML como estándar para el intercambio de datos, por lo que a veces también se les llama Servicios Web XML.

✓Las aplicaciones basadas en servicios Web son extremadamente portables y pueden fácilmente interactuar con diferentes tipos de software.

✓Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

✓Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

✓Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

✓Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar. *[www17]*

✓Se comunica a través de http, por lo que puede utilizar Internet.

✓Su desarrollo e implementación son bastante sencillos, comparados con protocolos de integración anteriores (CORBA, RMI, DCOM...)

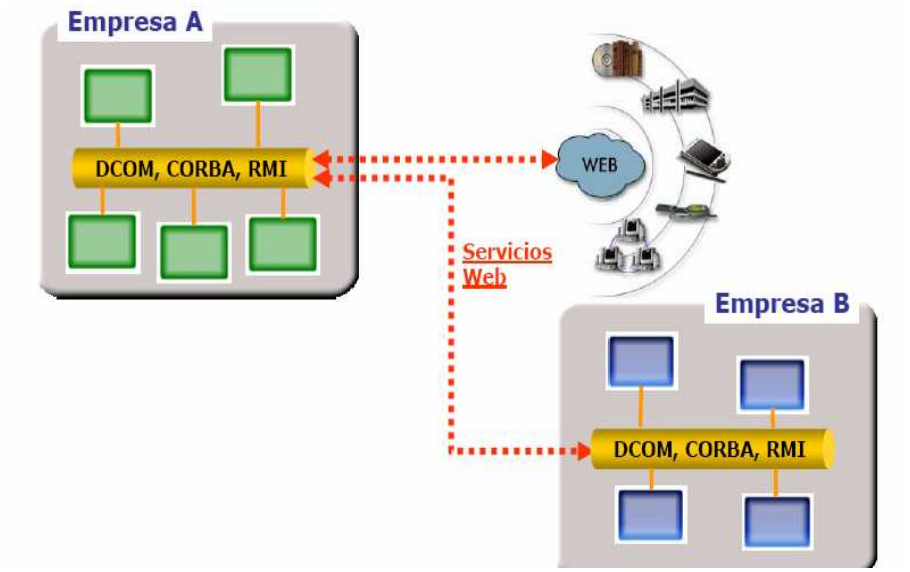


Figura 2.9 Interoperabilidad

2.10 RAZONES PARA CREAR SERVICIOS WEB

☑ La principal razón para usar servicios Web es que se basan en http sobre TCP en el puerto 80. Dado que las organizaciones protegen sus redes mediante firewalls, que filtran y bloquean gran parte del tráfico de Internet-, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web se vehiculan por este puerto, por la simple razón de que no resultan bloqueados.

☑ Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otros ordenadores en red. Las que había eran *ad hoc* y poco conocidas, tales como EDI, RPC, u otras APIs.

☑ Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno, no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más usada. *[www19]*

2.11 HERRAMIENTAS - SERVICIOS WEB

Existen varias herramientas que permiten la creación, generación y publicación de servicios web por ejemplo:

- Axis y el servidor Jakarta Tomcat (de Apache)
- ColdFusion MX de Macromedia
- Java Web Services Development Pack (JWSDP) de Sun Microsystems (basado en Jakarta Tomcat)
- JOnAS (parte de *ObjectWeb* una iniciativa de código abierto)
- Microsoft .NET
- Novell exteNd (basado en la plataforma J2EE)
- WebLogic
- WebSphere
- Zope es un servidor de aplicaciones Web orientado a objetos desarrollado en el lenguaje de programación Python

- VERASTREAM de AttachmateWRQ para modernizar o integrar aplicaciones host IBM y VT
- Mono *[www27]*

2.11 EJEMPLOS DE SERVICIOS WEB

Los servicios Web XML, aunque sean independientes entre sí, pueden vincularse para realizar una tarea.

Por ejemplo, Google, utiliza un Servicio Web -Google Web APIs- basado en los estándares SOAP y WSDL que permite programar en Java, Perl ó Visual Studio.NET y que sirve para la recuperación de información permitiendo utilizar este buscador en distintas plataformas y Servicios Web*. [www25]*

Por su parte, Amazon Web Services ofrece una serie de de aplicaciones de referencia que permiten a los desarrolladores acceso directo a la plataforma de tecnología de Amazon y construir aplicaciones propias. Una lista promenorizada de muchos de los servicios web existentes en la actualidad los ofrece XMethod: http://www.xmethods.com *[www26]*

CAPITULO 3

ARQUITECTURA ORIENTADA A SERVICIOS

"Aleja de tu corazón el enojo y echa fuera de ti la maldad, porque confiar en la juventud es vanidad"

EC 11:10

3.1 INTRODUCCIÓN

Desde la introducción de las tecnologías de información (TI) en la industria, las empresas han construido sus sistemas de información en base a las necesidades específicas de las unidades de negocio, conformando para esto infraestructuras de datos que más tarde se convirtieron en "silos de información", usualmente aislados o pobremente cohesionados.

Dicha situación llegó a un punto crítico cuando las empresas comenzaron a enfrentar complejos procesos de expansión o fusión, obligándolos a organizar toda esta información y transferirla de manera eficiente y consolidada a los ahora diversos actores de la organización, además de las necesidades constantes de alimentar los canales con clientes y proveedores.

Los Sistemas de Información de las organizaciones son el resultado de la sedimentación de aplicaciones y tecnologías heterogéneas de las últimas décadas.

Cada vez se demandan aplicaciones más complejas, con menos tiempo de desarrollo, presupuesto, y en muchos casos se requiere reimplementar funcionalidades preexistentes.

Tratar de reutilizar estas funcionalidades resulta una labor difícil de realizar debido a que no fueron diseñadas para integrarse, o bien se desarrollaron sobre plataformas o tecnologías incompatibles entre sí.

El concepto de "integración" se convierte en la llave maestra necesaria para facilitar la gestión estratégica, táctica y operacional de los diversos actores de una empresa, lo que le permitirá reducir los esfuerzos de desarrollo y mantención de las TI, poniéndolas definitivamente al servicio del crecimiento de su empresa. *[www28]*

SOA se presenta en este escenario como una solución adecuada, definiendo una arquitectura donde todas las actividades o los procesos están diseñados para ofrecer un Servicio. Se entiende por Servicio componentes de software con interfaces bien definidas e independientes de su implementación.

Este modelo de arquitectura representa una forma de organizar los sistemas de información que permite la interacción entre los diferentes Servicios ofrecidos.

De esta forma inicio este capitulo dedicado al estudio de la tecnología de integración SOA, para descubrir sus funcionalidades, características, ventajas, etc.

3.2 SOA EN LA INDUSTRIA

- "La recompensa potencial de SOA es enorme para las empresas que entiendan esta evolución y se muevan hacia estas arquitecturas. ... La tecnología de computación distribuida promete ser lo suficientemente flexible y elegante para responder a las necesidades de negocios y proporcionar la agilidad de negocios que las compañías han anhelado tanto tiempo, pero siempre ha estado fuera de alcance".

  [*The Rational Edge, 2004]*

- "SOA ha surgido como la mejor manera de afrontar el desafío de hacer más con menos recursos. Promete hacer la re-utilización y la integración mucho más fáciles, ayudando a reducir el tiempo de desarrollo y aumentando la agilidad organizacional. No sorprendentemente, el 80% de las organizaciones de IT están implementando aplicaciones usando SOA con web services subyacentes. SOA proporciona mayor flexibilidad para afrontar los cambios tanto en el ambiente de negocios como en la infraestructura tecnológica". *[M7 Corporation]*

- "Comprender el rol y el significado de SOA, más allá del *hype* simplista, es imperativo para cualquier arquitecto de software empresarial. ... Hacia 2008, SOA y Web Services serán implementados juntos en más del 75% de los proyectos que utilicen SOA y Web Services (probabilidad 0.7)" *[Gartner, 2003]*

- "Giga recomienda a los arquitectos considerar SOA como la prioridad número uno en sus esfuerzos de planeamiento arquitectónico"

  [Giga *IT Trends 2003: Application architecture and design*]

[www11-www29]

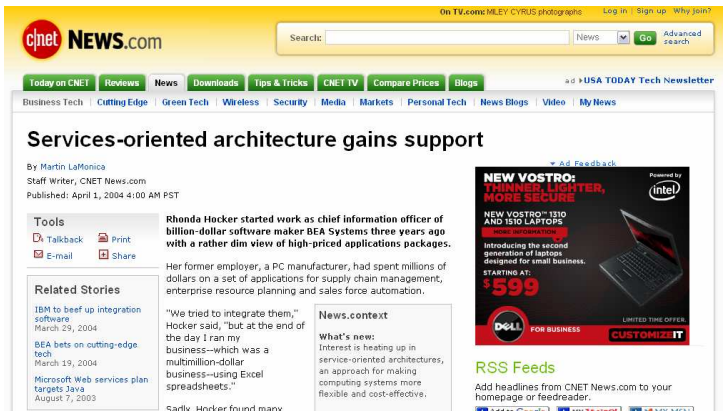


Figura 3.1 SOA en la Industria

3.3 SOA UNA TECNOLOGÍA DE INTEGRACIÓN

Debido a la complejidad creciente de sistemas y aplicaciones la aproximación tradicional de definir conexiones punto a punto entre dos sistemas resulta ser ineficiente y demasiado cara. ¿Cómo responder a este reto de la integración? La respuesta parece provenir de la siguiente constatación, al cliente no le preocupan la tecnología o las funcionalidades de una aplicación, le preocupan los servicios y su realización eficaz (Kalakota et al, 2003).

Esta idea de servicios ha encontrado su forma de expresión mediante los Servicios Web.

Un Servicio Web es una representación en un lenguaje estándar (XML) de un programa, objeto, bases de datos o cualquier otro tipo de función de negocios. La tecnología de Servicios Web define los siguientes estándares:

1. *El lenguaje en que los servicios son descritos (XML).*
2. *El protocolo de comunicación de mensajes escritos en XML (SOAP).*
3. *La descripción de los tipos de datos y estructuras de los Servicios Web (WSDL).*
4. *El mecanismo para publicar y encontrar los Servicios (UDDI).*

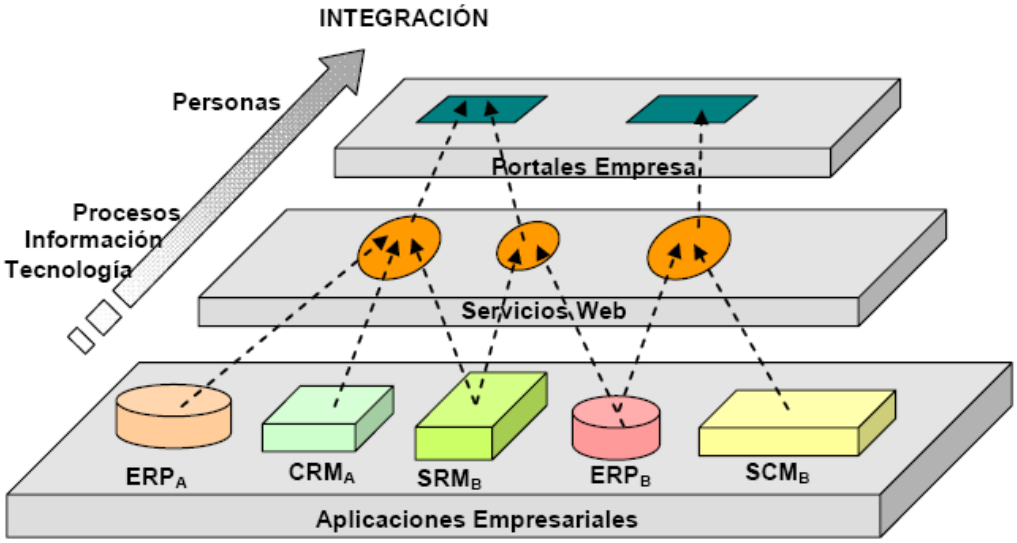


Figura 3.2 Integración mediante SOA

La tecnología de Servicios Web soporta la llamada Arquitectura Orientada a Servicios (SOA) (como muestra la figura 3.2), que da respuesta al reto de la integración mediante la integración de:

1. Las personas (tecnología de portales de empresa).

2. La información (Data Warehousing).

3. Los procesos de negocio (Business Process Management).

4. Y la tecnología (Web Services).

Si el objetivo es la construcción de servicios de negocio basados en la integración de diferentes aplicaciones empresariales, que además pueden cambiar rápidamente como cambian las relaciones entre empresas, parece razonable desplazar la lógica de la integración (personas, información, procesos) fuera de las aplicaciones en un nivel de abstracción superior que hace posible la tecnología de Servicios Web.

El propósito de esta plataforma abierta de servicios es el de hacer transparente la complejidad de la integración de las diferentes aplicaciones. Los usuarios no tendrán que acceder a las aplicaciones empresariales a través de las tradicionales GUI estándar, sino que lo harán a través de portales de empresa personalizados. *[www30]*

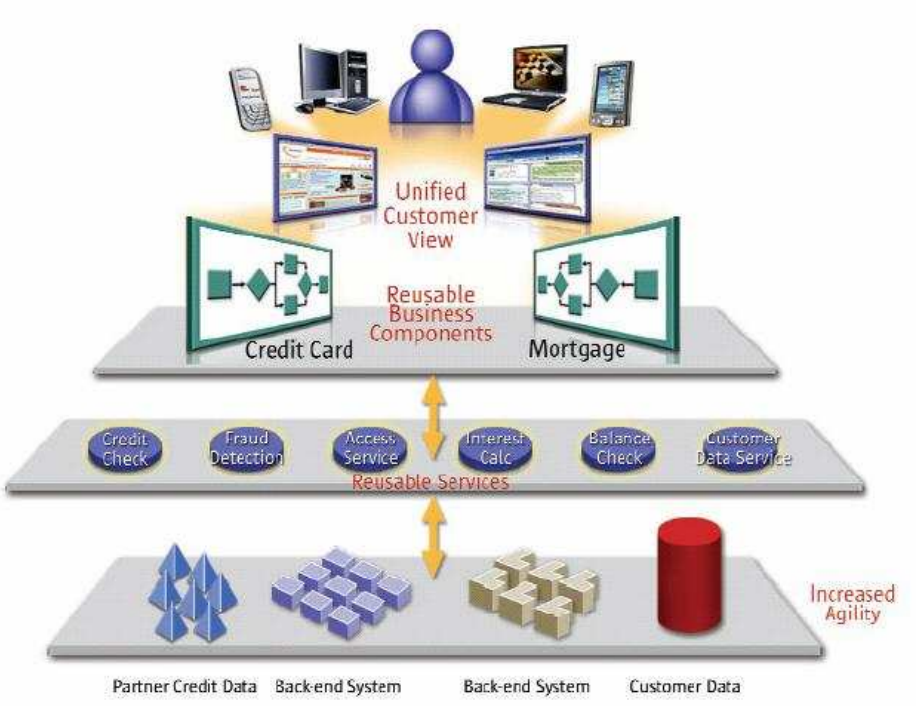


Figura 3.3 SOA la solución a la integración de aplicaciones empresariales

3.4 OBJETIVOS DE UNA ARQUITECTURA ORIENTADA A SERVICIO (SOA)

Se contemplan dos objetivos diferentes, desde el punto de vista empresarial y desde el punto de vista de la tecnología.

3.4.1 Desde el punto de vista empresarial

Cuando una empresa decide hacer uso de una arquitectura SOA es porque tiene objetivos específicos de negocio que cubrir, reducir costes, aumentar ingresos, mejorar la productividad, comunicación inter-empresarial con varias empresas y ajustar los sistemas a los requerimientos del negocio. *[www31].*

También se puede decir que una arquitectura SOA consiste en una forma de modularizar los sistemas y aplicaciones en componentes de negocio que pueden combinarse y recombinarse con interfaces bien definidas para responder a las necesidades de la empresa.
Con el uso de entornos orientados a servicios las empresas pretenden mejorar la interacción con los clientes, partners, proveedores, empleados y también reducir el ROI (Return of Investment) retorno de la inversión, es decir, conseguir una mayor rentabilidad
de las inversiones tecnológicas. Para las empresas se abre un abanico amplio de aplicación, desde la utilización en la cadena de suministro, entornos B2B o servicios de seguridad.

3.4.1.1 Beneficios para el negocio

- ***Eficiencia***. Transforma los procesos de negocio en servicios compartidos con un menor coste de mantenimiento.
- ***Capacidad de respuesta***. Rápida adaptación y despliegue de servicios, clave para responder a las demandas de clientes, partners y empleados.
- ***Adaptabilidad***. Facilita la adopción de cambios añadiendo flexibilidad y reduciendo el esfuerzo.

3.4.2 Desde el punto de vista tecnológico

Las arquitecturas SOA pretenden concebir las aplicaciones desde otro punto de vista, una aplicación orientada a servicios combina datos en tiempo real con otros sistemas capaces de fusionar los procesos de negocio.

Las aplicaciones basadas en SOA utilizan tecnología totalmente estándar como es XML y servicios Web para la mensajería. Estándares como SOAP, Web Services Description Language (WSDL) y Business Process Execution Language (BPEL) estandarizan así la compartición de información, el modelo de integración de procesos y la cooperación entre aplicaciones. *[www12]*

Realizando aplicaciones orientadas a servicio se pueden conectar aplicaciones heterogéneas con el aumento de flexibilidad que supone, y un punto muy importante es que permite que las organizaciones interactúen cuando realmente lo requieran, sin necesidad de tener conexiones permanentes. *[www19]*

Como una arquitectura SOA se basa en estándares, el tiempo de aprendizaje de utilización de las tecnologías sobre las que se apoya se reduce drásticamente.

3.4.2.1 Beneficios Tecnológicos

- Reduce la complejidad gracias a la compatibilidad basada en estándares frente a la integración punto a punto.
- Reutiliza los servicios compartidos que han sido desplegados previamente.
- Integra aplicaciones heredadas limitando así el coste de mantenimiento e integración.
- Beneficios en el desarrollo, ya que las aplicaciones son reutilizables, más fácil de mantener y tienen la capacidad de ampliación de las funcionalidades del sistema, exponiéndolas de una forma segura.
- Permite la creación y cambio de servicios de forma incremental, evitando proyectos de larga duración y alto coste. *[lib02]*

3.5 ARQUITECTURA ORIENTADA A SERVICIOS

Definiciones.-

"SOA es un modelo de componentes que interrelaciona las diferentes unidades funcionales de las aplicaciones, denominadas servicios, a través de interfaces y contratos bien definidos entre esos servicios. La interfaz se define de forma neutral, y debería ser independiente de la plataforma hardware, del sistema operativo y del lenguaje de programación utilizado. Esto permite a los servicios, construidos sobre sistemas heterogéneos, interactuar entre ellos de una manera uniforme y universal." *[www32]*

Una arquitectura orientada a servicios (SOA) es una evolución de la llamada computación distribuida, basada en el paradigma de pregunta/respuesta para aplicaciones sincrónicas y asincrónicas.

En ella la lógica de negocios o las funciones individuales son modularizadas y presentadas como servicios para aplicaciones consumidoras/clientes. Lo que es clave de estos servicios es su naturaleza desacoplada; la interfaz de servicios es independiente de la implementación.

Una estrategia de aplicaciones debe facilitar su integración. Además que debe motivar la construcción de servicios, mas que aplicaciones. Estos servicios se encargarían de exponer una funcionalidad bien definida a la aplicación que la requiera.

SOA propone una aplicación final que simplemente orquesta la ejecución de un conjunto de estos servicios, añade su lógica particular y le presenta una interfaz al usuario final. *[www33].*

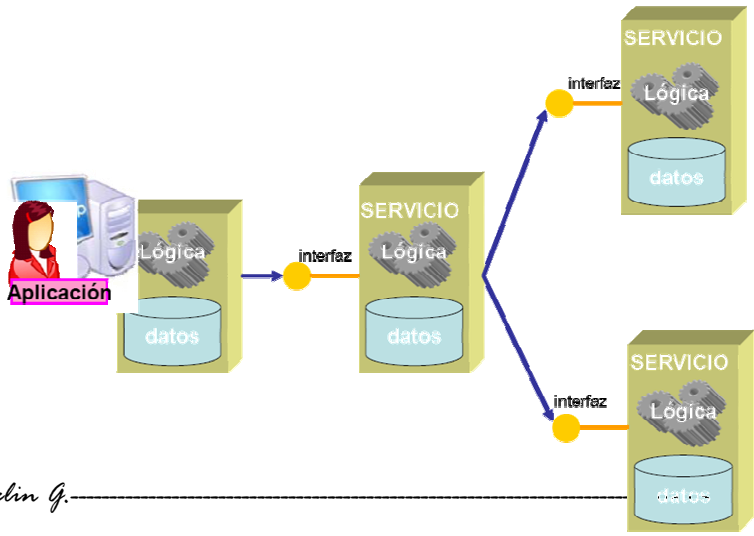


Figura 3.4 Integración- una sola interfaz para el usuario.

Exponer procesos de negocio como servicios es la clave de la flexibilidad de la arquitectura. Esto permite que otras piezas de funcionalidad (incluso también implementadas como servicios) hagan uso de otros servicios de manera natural, sin importar su ubicación física. Así un sistema evoluciona con la adición de nuevos servicios y con su mejora continua. Donde cada servicio evoluciona de una manera independiente. La Arquitectura Orientada a Servicios resultante, define los servicios de los cuales estará compuesto el sistema, sus interacciones, y con que tecnologías serán implementados. *[www34].*

Funcionamiento.-

SOA puede verse como un "Paradigma de Arquitectura Corporativa".

En su momento, el Paradigma de Orientación a Objetos supuso un cambio en el análisis, diseño y construcción de aplicaciones. SOA, sin embargo, nos propone un nuevo modelo de estructuración, orientado a atacar la definición de arquitecturas en vez de la definición de aplicaciones individuales.

Desarrollar procesos de negocio como Servicios es la clave de la flexibilidad de la arquitectura, esto permite que otros componentes funcionales hagan uso de diferentes servicios de manera natural con independencia de su ubicación.

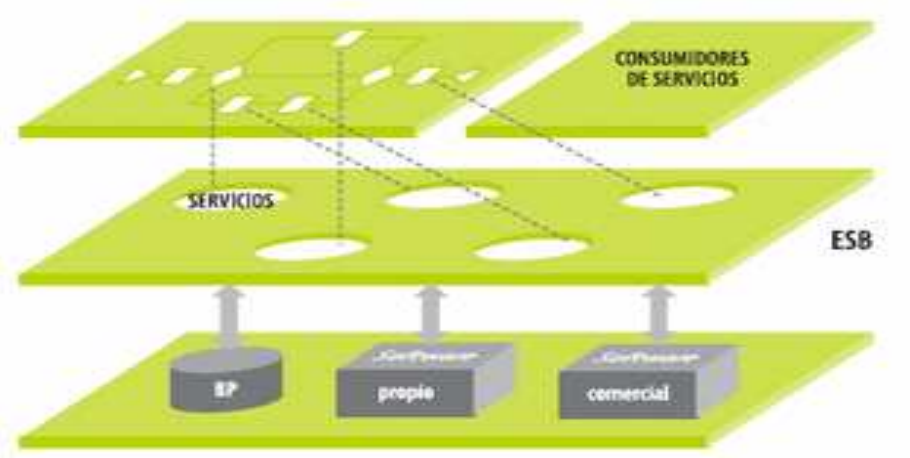


Figura 3.5 SOA - Arquitectura

El diseño que sigue los parámetros de esta arquitectura es fácilmente escalable, sobre ella se pueden añadir futuros Servicios y permite que cada Servicio evolucione de manera independiente.

SOA plantea aplicaciones distribuidas, multi-nivel con capas de presentación, lógica de negocios y persistencia.

Propone varias capas de servicios que exponen funcionalidades de negocio que a su vez permiten la composición de aplicaciones a partir de los mismos.

A la hora de desarrollar, en lugar de crear aplicaciones enormes y muy complejas, se desarrollan componentes reutilizables, (Servicios) que son fáciles de mantener y probar. Las aplicaciones se crean diseñando un proceso que interactúa con estos servicios, donde se reutilizan lo componentes existentes y sólo se desarrollan aquellos componentes no implementados.

Esta arquitectura basada en Servicios requiere una infraestructura de comunicaciones escalable y segura entre los componentes, que se convierte en el eje vertebrador de todos los sistemas de la organización y que se conoce como *Entreprise Service Bus o Bus empresarial*.

Para SOA es indiferente la tecnología que utilicemos, sin embargo, la tecnología de Web Services o Servicios Web es una de las que más notoriedad ha conseguido. En muchos casos y contextos SOA y Servicios Web es tratado como un mismo concepto, pero Web Services no es sinónimo de SOA. Por el contrario, es posible utilizar Web Services y seguir un esquema de integración punto a punto, así como es posible implantar un esquema SOA sin utilizar Web Services. Sin embargo, tanto Web Services como los estándares asociados (XML, SOAP, WSDL, UDDI, WS-I) conforman una base de tecnologías y estándares que facilitan la implementación de SOA.

En este contexto, SOA brinda beneficios tangibles en las organizaciones al margen de su tamaño. No obstante, la clave de éxito pasa por un cambio organizacional y cultural, además de un cambio tecnológico. Nokia, British Telecom, UPS, son organizaciones que han implementado SOA con éxito.

Es necesario un largo camino de transformación en una organización debido a que SOA no es algo que compramos y que ya esté hecho, no es un producto o una tecnología por sí mismo, sino que es una forma totalmente diferente de dirigir y gestionar la empresa. Este enfoque supone un cambio estructural importante para los sistemas de información de la mayoría de organizaciones, su implantación es por tanto una tarea ardua y costosa donde los verdaderos beneficios de este enfoque no empiezan a ser evidentes hasta que el cambio se ha generalizado y se supera la dura barrera que supone la adaptación de los sistemas preexistentes.

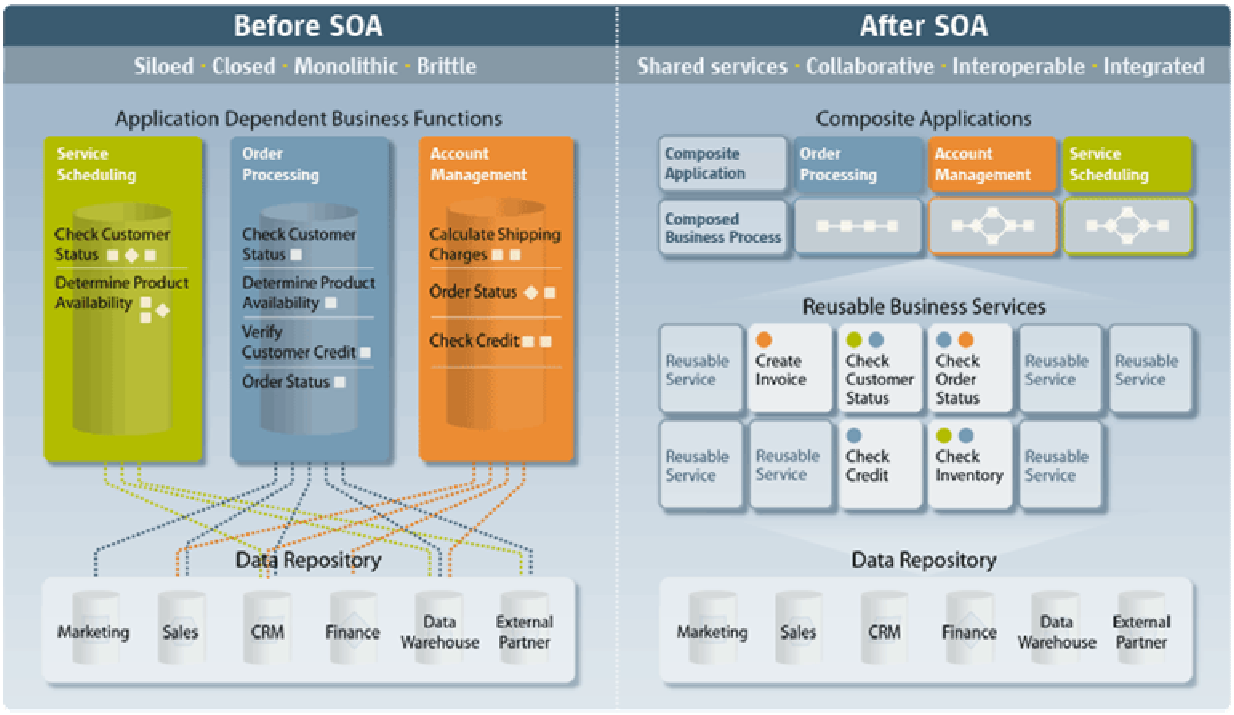


Figura 3.6 Antes y Después de aplicar SOA en una empresa con TI existente

A pesar de los problemas y costes que este cambio de enfoque pueda suponer, la organización de nuestros servicios de información basada en SOA se presenta como la solución idónea para obtener una buena interoperabilidad de sistemas de información en entornos complejos y etéreos. *[www31].*

3.6.1 ELEMENTOS DE SOA

Los componentes de una Arquitectura Orientada a Servicios son:

- Servidores

- Repositorio de Servicios
- Bus de servicios
- Consumidores

3.6.1.1 Servidores

Un servicio de negocio es un componente reutilizable de software, con significado funcional completo, y que está compuesto por:

- **Contrato:** especificación de la finalidad, funcionalidad, forma de uso y restricciones del servicio.
- **Interfaz:** mecanismo de exposición del servicio a los usuarios.
- **Implementación**: debe contener la lógica o el acceso a datos.

Figura 3.7 Elementos de SOA -Servidores

Tipos de servicios.- pueden existir varios tipos de servicios, según su finalidad

Servicios básicos: pueden estar centrados en datos o en lógica y encapsulan funcionalidades como cálculos complejos, acceso a datos y reglas complejas de negocio.

Servicios intermediarios: servicios adaptadores, *façades*, etc.

Servicios de proceso: servicios de negocio que encapsulan la lógica de proceso. Pueden residir en herramientas BPM.

Servicios públicos: servicios accesibles por terceros (fuera de la organización). *[www31].*

3.6.1.2 Repositorio de Servicios

Un repositorio de servicios proporciona facilidades para descubrir servicios y adquirir la información necesaria para su uso, en particular fuera del alcance temporal y funcional del proyecto en el que se crearon.

Además de la propia información de contrato, los repositorios pueden proporcionar información acerca de:

- Localización.
- Personas de contacto.
- Restricciones técnicas.
- Service Level Agreements (SLAs). *Acuerdos de Nivel de Servicio.*

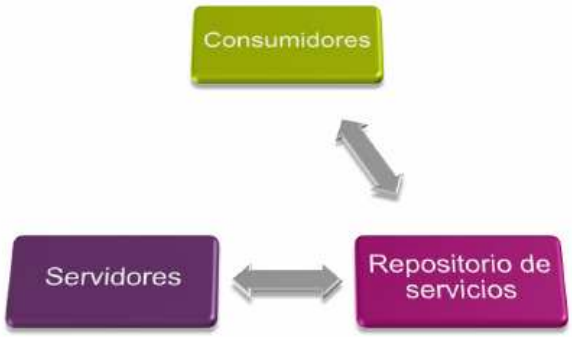


Figura 3.8 Elementos de SOA – Repositorio de Servicios

3.6.1.3 Bus de Servicios

La intersección de la arquitectura orientada a servicios con la integración de aplicaciones y el modelado de procesos de negocio, dan lugar a un nuevo producto de nominado bus de servicios conocido también como ESB *(Enterprise Service Bus- Bus Empresarial de Servicios).*

El ESB es un elemento de software, un middleware, una infraestructura basada en estándares, que proporciona servicios para la construcción de arquitecturas más complejas basadas en eventos y en un motor de mensajería (el BUS).

El bus de servicios es el elemento de las arquitecturas SOA que conecta los servicios con sus consumidores y que proporciona:

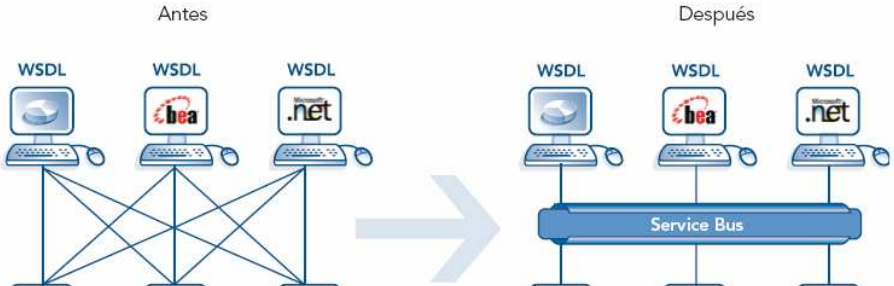
Conectividad: el propósito principal de un bus de servicios es interconectar a los participantes de una arquitectura SOA.

Soporte a la heterogeneidad de tecnologías: debe ser capaz de conectar a participantes basados en distintos lenguajes de programación, sistemas operativos, entornos de ejecución y protocolos de comunicación.

Soporte a la heterogeneidad de paradigmas de comunicación: debe ser capaz de mantener distintos modos de comunicación (por ejemplo comunicaciones síncronas y asíncronas).

Figura 3.9 Elementos de SOA – Bus de Servicios

El ESB permite la integración de aplicaciones de forma rápìda, directa y basada en estándares. Es una suite de productos independientes de la infraestructura de facilita el procesado, la transformación de datos, el enrutamiento y la orquestación de procesos usando Servicios Web.



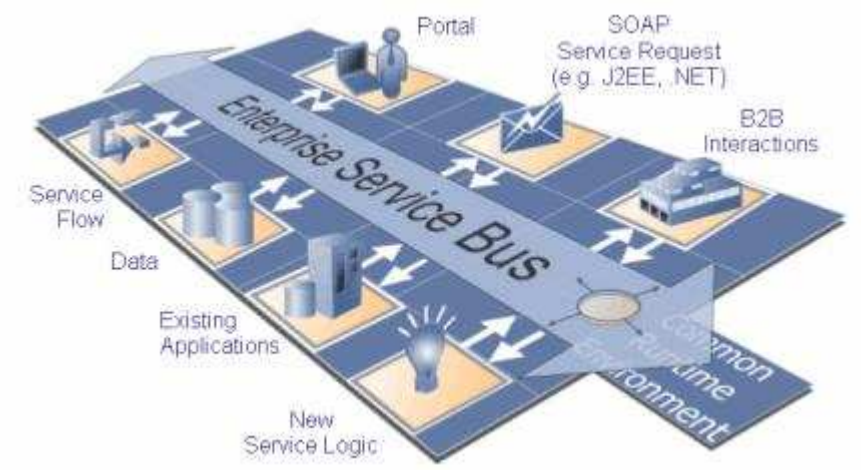


Figura 3.11 El ESB en la integración de aplicaciones

El ESB está integrado por aplicaciones proveedoras de servicios. Cada una de las aplicaciones puede ser independiente de las demás. Esta posibilidad de integración es independiente del modo de licencia del software, libre o propietario. Esta arquitectura nos permite, por ejemplo, a través del ESB utilizar con una herramienta de software propietario unos servicios desarrollados sobre plataformas de software libre. *[www32].*

3.4.1.3.1 Como funciona el ESB

El modelo de ESB se plasma en una red integrada por nodos de servicios colaboradores, desplegados en "contenedores de servicios".

Los contenedores de servicios se despliegan en partes específicas de la red, en función de la localización de los puntos extremos de la aplicación y de los puntos de servicios de integración requeridos, tales como la transformación o el encaminamiento inteligente. Estos

contenedores de servicios se conectan en una topología de bus lógico mediante los servidores de comunicación.

Mensajes XML

Las aplicaciones interactúan mediante mensajes XML, que meten y sacan contenedores de servicios en los puntos extremos. Las aplicaciones no necesitan ser conscientes de los protocolos de comunicaciones subyacentes ni de las localizaciones físicas; tan sólo ven simples "cajas" que entran y salen. Así, los servicios pueden ser actualizados, trasladados o reemplazados sin tener que interrumpir los sistemas de negocio ni modificar las aplicaciones.

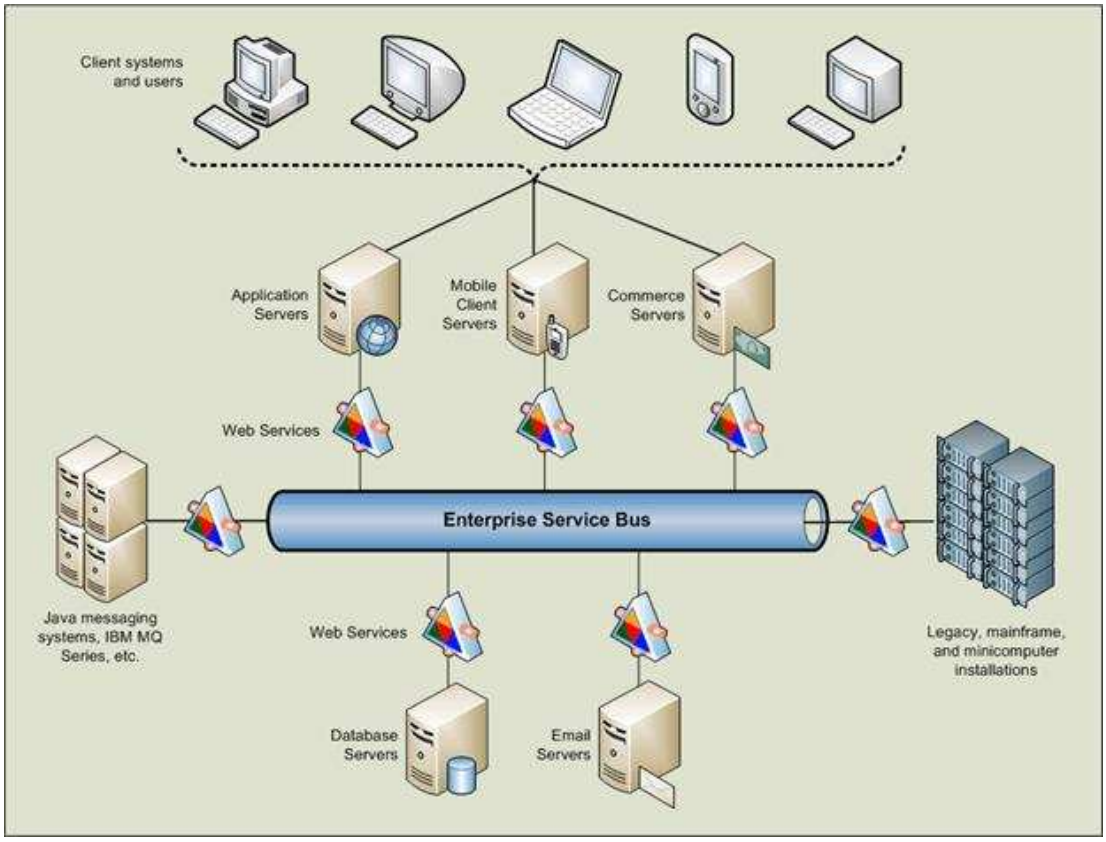


Figura 3.12 La interacción de Servicios Web con el ESB

El uso de XML en ESB proporciona una gran flexibilidad y hace a la infraestructura más resistente a los cambios de los negocios y las aplicaciones. Por ejemplo, usando hojas de estilo XML, ESB puede pasar los contenidos de los mensajes de un formato a otro. Las aplicaciones no necesitan adherirse a un formato específico ni hay que enviar los datos a un punto central para su transformación. *[lib03].*

ESB trata a todas las aplicaciones como servicios, con independencia de cómo se conecten al bus, permitiendo a las empresas migrar paulatinamente a una arquitectura basada en servicios con un riesgo mínimo y una eficaz planificación de las inversiones. Es sencillo desarrollar interfaces de servicio para aplicaciones creadas en entornos .Net de Microsoft o en Java 2 Platform Enterprise Edition utilizando herramientas de terceros.

Es más, ESB proporciona diversas opciones para tratar las aplicaciones existentes. Un enfoque común consiste en usar adaptadores específicos de aplicación o adaptadores file-drop.

Un archivo file-drop utiliza mensajes XML para interactuar con un ESB al tiempo que emplea ficheros planos para interactuar con la aplicación pretendida. Los adaptadores de aplicaciones generalmente están escritos por terceras firmas y proporcionan el enlace entre la interfaz de servicio basada en mensajes y las interfaces de código nativo de la aplicación concreta de que se trate.

Encaminamiento inteligente

Cada servicio está descrito en un directorio común. Los desarrolladores conectan las aplicaciones buscando los servicios en el directorio y orquestando sus interacciones mediante encaminamiento inteligente. Un itinerario XML contiene las órdenes necesarias para crear la secuencia de servicios que el mensaje debe pasar para completar un proceso.

El encaminamiento de mensajes puede cambiar de acuerdo a eventos en tiempo real y a sus propios contenidos. Por ejemplo, el desarrollador puede emplear una combinación de expresiones XPATH y de reglas de encaminamiento basadas en JavaScript para dirigir la entrega del mensaje.

Es importante tener en cuenta que las tecnologías usadas en ESB están basadas en estándares, lo que tiene un significativo impacto: los proyectos de integración requerirán mucho menos outsourcing. *[www35]*

3.4.1.3.2 Elementos esenciales del Bus de Servicios

Estos elementos juntos resuelven los problemas a los que se enfrentan los clientes y los proveedores de servicios en un entorno de SOA.

Mensajería distribuida. El núcleo del ESB lo constituye una aplicación de middleware orientada al mensaje, como el software Enterprise Message Service™ Este núcleo proporciona un método de transporte fiable y distribuido que emplea un mecanismo de almacenamiento y reenvío gracias al cual se garantiza la entrega de los mensajes incluso en caso de anomalías en la red.

Transparencia de las ubicaciones. Con la mediación entre servicios, un servicio cliente que invoque al proveedor de servicio solo necesita saber que el servicio existe; el cliente no necesita saber dónde se está ejecutando el servicio. El ESB localiza el servicio cuando se invoca. Esto proporciona un cierto nivel de virtualización de los servicios y de transparencia de las ubicaciones, de forma que si un equipo falla, o si se cambia la ubicación de un proveedor de servicio, no es preciso notificar el cambio a cada uno de los clientes individuales. Esto puede contribuir significativamente a la reducción de los costes de gestión de las TI y a minimizar los riesgos.

Transparencia del transporte. En los enfoques tradicionales de la integración punto a punto todos los componentes y objetos están muy estrechamente acoplados. En la SOA, los servicios están repartidos por todo el entorno de TI y su acoplamiento es menos estricto, gracias a la transparencia de las ubicaciones. Además de apoyarse en la transparencia de las ubicaciones para conectar clientes y proveedores de servicios, el ESB también proporciona protocolo de trasporte físico para hacer posible la comunicación entre servicios utilizando transportes diferentes.

Soporte multiprotocolo. Debido a que plantea cuestiones de fiabilidad inherentes y solamente funciona bien con patrones de intercambio de mensajes (MEP) sincrónicos, el modelo de transporte HTTP no satisface los requisitos de todos los servicios y aplicaciones. Por ejemplo, el servicio de mensajes de Java (JMS) además de poseer características asincrónicas, ofrece más fiabilidad en el transporte que HTTP. Para compatibilizar el comportamiento de las aplicaciones individuales, algunos sistemas recurren a SOAP a través de JMS. También se usan otros tipos de modelos de transporte, entre los que se cuentan los sistemas de transporte propietarios de algunos de los principales proveedores de sistemas y soluciones de planificación de recursos empresariales. ESB necesita, por lo tanto, ser capaz

de soportar muchos tipos de sistemas de transporte para integrar sistemas dispares y gestionar el transporte de comunicaciones complejas eficazmente. *[www36].*

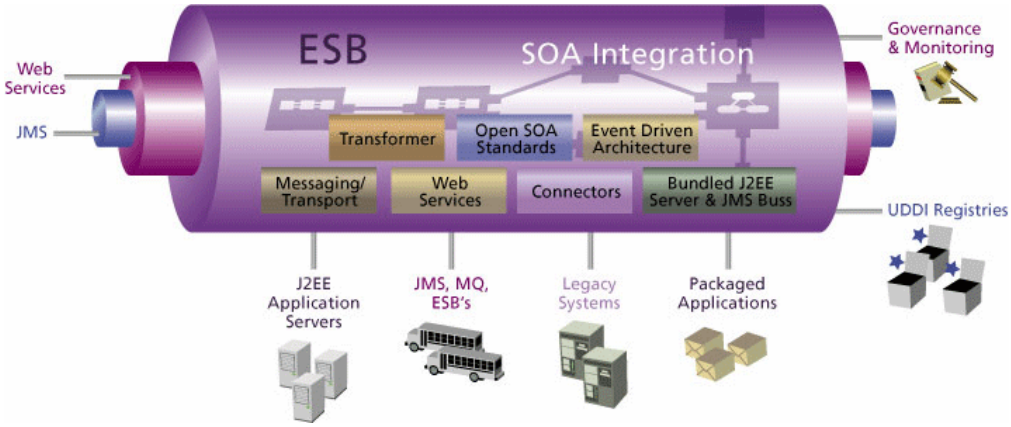


Figura 3.13 ESB ineractúa con las diferentes tecnlogías.

Calidad de servicio. En las aplicaciones empresariales, la calidad de servicio (QOS) hace referencia, fundamentalmente, a su fiabilidad. La entrega de los mensajes y la fiabilidad del servicios de invocación son funciones de misión crítica de cualquier sistema. Aun los servicios Web por sí solos no ofrecen un servicio de entrega garantizada. Un ESB, por otro lado, puede proporcionar un servicio de alta fiabilidad garantizando la entrega del mensaje de principio a fin que va más allá de la fiabilidad que puede ofrecer transportes como JMS. Asimismo, los métodos empleados para alcanzar un alto nivel de QOS deben satisfacer los estándares existentes, como, por ejemplo, ser compatibles con la especificación WS-ReliableMessaging. *[www36].*

Patrones de intercambio de mensajes. En la actualidad, la mayoría de los ESB se basan en un paradigma de solicitud/respuesta usando SOAP sobre HTTP; esto significa que el servicio cliente lanza un mensaje de solicitud al usuario y espera a recibir la respuesta. Esto se conoce como un MEP sincrónico. Sin embargo, en el MEP de publicación/suscripción, el servicio cliente puede enviar un mensaje y suscribirse a la respuesta, en lugar de esperar a

recibirla. El MEP de publicación/suscripción puede responder de forma más eficaz a eventos en un contexto empresarial, en particular cuando el ciclo de vida de una acción de servicio tiene lugar durante períodos de tiempo prolongados. Un ESB debe ser capaz de manejar ambos paradigmas.

Enrutamiento basado en el contenido. Existen dos tipos de enrutamiento dentro de un ESB.

- El primer servicio de enrutamiento se produce cuando la invocación de un servicio entra en el ESB y éste encamina la respuesta al proveedor de servicio apropiado, sin necesidad de que el servicio cliente conozca la ubicación del proveedor del servicio. Así es cómo se logra la transparencia de las ubicaciones que antes hemos comentado.
- El otro tipo, enrutamiento basado en el contenido, introduce una serie de reglas o una lógica de negocio que se aplica al contenido del mensaje en la etapa del enrutamiento y que hacen posible que el ESB encamine los mensajes a proveedores de servicio específicos basándose en su contenido; dando prioridad, por ejemplo, a los pedidos de determinados clientes o marcando los pedidos de gran tamaño para darles un tratamiento especial.

Esto ofrece a las empresas un servicio muy valioso, ya que puede contribuir a reducir el coste de la gestión de la Información, garantiza que se respeten los acuerdos a nivel de servicio y permite a las empresas centrarse en actividades para mejorar la satisfacción de sus clientes.

Transformación. Si bien la tarea de un ESB es dirigir mensajes de un servicio al siguiente, hay ocasiones en que el formato de los datos de un servicio no satisface los requisitos del siguiente servicio. Por ese motivo, el ESB debe ser capaz de transformar los datos de un formato a otro. *[www36].*

3.6.1.4 Consumidores de Servicios

Definimos consumidores de servicios como aquellos elementos de una arquitectura SOA que:

- Pueden descubrir servicios a través de un repositorio.

- Realizan llamadas a los mismos de acuerdo al contrato y a través del interfaz definido a tal efecto. *[lib02].*

Figura 3.14 Elementos de SOA – Consumidores de servicios.

3.7 CAPAS DE LA ARQUITECTURA

Como cualquier aplicación distribuida, las aplicaciones orientadas a servicios son multicapas, y tienen capas de presentación, lógica de negocio y persistencia. La figura siguiente muestra una arquitectura típica de una aplicación orientada a servicios. Las dos capas clave en SOA son la de servicios y la de procesos de negocio. *[www37]*

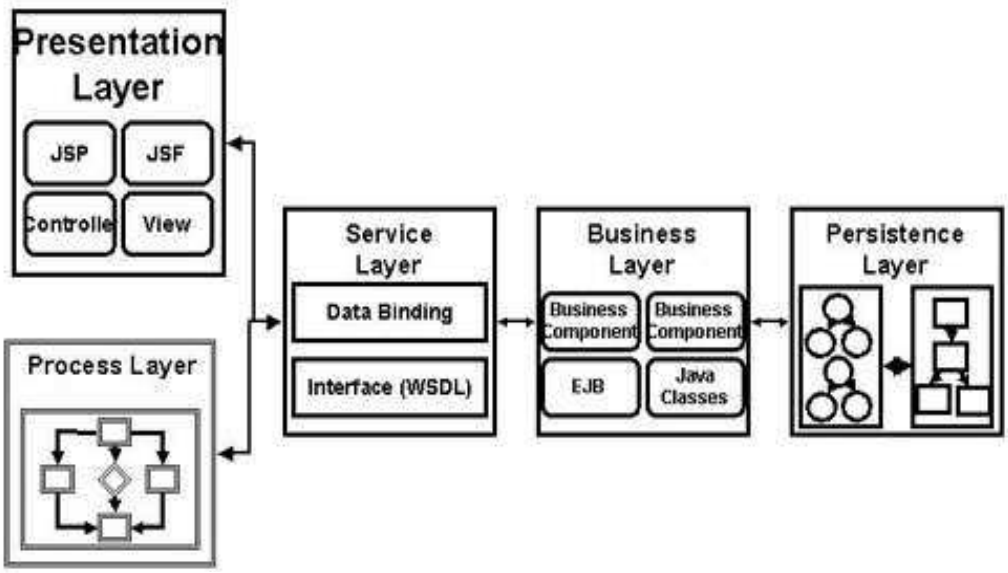


Figura 3.15 Capas de SOA

3.7.1 La capa de servicios

Los servicios son los elementos básicos de SOA. Los servicios son análogos a los objetos Java y a los componentes tales como EJBs. A diferencia de los objetos, los servicios son autocontenidos, tienen estado y poseen poco acople.

El gran desafío de crear aplicaciones SOA es diseñar las interfaces con el nivel de abstracción correcto. Mientras analiza los requisitos del negocio, debe considerar cuidadosamente los servicios existentes y cuales componentes se deben ensamblar a partir de estos. Generalmente los servicios deben tener una funcionalidad significativa.

Por ejemplo, un componente que procese una orden de compra es un buen candidato para publicarlo como un servicio, al contrario de un componente que simplemente actualice un atributo de esa orden de compra.

Existen dos opciones para crear un servicio:

- Top- down (arriba- abajo)
- Bottom- up (abajo- arriba).

La opción top- down requiere identificar y describir los mensajes y operaciones que el servicio deba ofrecer y entonces implementarlo. Esta modalidad es recomendable cuando se crea un servicio completamente nuevo, pues permite elegir la tecnología preferida para su implementación. Esta modalidad también promueve la máxima interoperabilidad de los servicios, ya que se pueden eliminar los problemas de implementación que la imposibiliten (por ejemplo tipos de datos que no tienen representación interoperable).

La opción bottom-up es un poco más popular por que da la posibilidad de reutilizar componentes de negocio existentes. Por ejemplo, hay herramientas para exponer procedimientos almacenados y así podría exponer uno que verifique si un cliente tiene derecho a cierto descuento.

El aspecto más importante de un servicio es su descripción. Cuando se usan servicios web para implementar una aplicación SOA, existe WSDL (Web Service Description Languaje) para

describir los mensajes, tipos de datos y operaciones del servicio; este mismo es el "contrato" que se debe cumplir para usar el servicio.

Este es un ejemplo de WSDL para un servicio web simple:

```
<?xml version="1.0" encoding="UTF8"?>
<definitions name="MyTimeService"
  targetNamespace="urn:oraclews"
  xmlns:tns="urn:oraclews"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<type/>

<message name="TimeService_getDateTime">
  <part name="String_1" type="xsd:string"/>
</message>

<message name="TimeService_getDateTimeResponse">
  <part name="result" type="xsd:string"/>
</message>

<portType name="TimeService">
  <operation name="getDateTime" parameterOrder="String_1">
      <input message="tns:TimeService_getDateTime"/>
      <output message="tns:TimeService_getDateTimeResponse"/>
  </operation>
</portType>

<binding name="TimeServiceBinding" type="tns:TimeService">
   <operation name="getDateTime">

      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="urn:oraclews"/>
      </input>

      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="urn:oraclews"/>
      </output>

      <soap:operation soapAction=""/>

   </operation>

   <soap:binding
    transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>

 </binding>

 <service name="MyTimeService">
  <port name="TimeServicePort" binding="tns:TimeServiceBinding">
   <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
  </port>
 </service>
```

Si lee cuidadosamente, se dará cuenta que el WSDL tiene una descripción completa del servicio web TimeService, incluyendo el puerto, las operaciones y los tipos de datos de los mensajes. Un servicio web basado en este WSDL debe proveer estas operaciones y mensajes, ya sea que esté construido con alguna tecnología Oracle o Microsoft .NET por ejemplo. *[www37]*

3.7.1.1 Construcción de la capa de servicios en Java

Java brinda una plataforma completa para construir la capa de servicio en una aplicación SOA. J2EE 1.4 estandariza los APIs para crear servicios web.
La siguiente tabla muestra una lista de APIs disponibles para crear servicios web con Java.

API Java	Descripción
JAXP	Interpretación de XML
JAXB	Asociación de XML con objetos Java
JAX-RPC (JSR 101)	API para llamado de procedimientos remotos (RPC) con XML
SAAJ	API para procesar mensajes SOAP
JAXR	API para manejar registros XML
JSR109	Modelo de despliegue de servicios
EJB2.1	EJB de sesion sin estado

Figura 3.16 APIs para crear servicios web con Java

De todas estas tecnologías, JAX-RPC puede pensarse como el núcleo del API para crear y desplegar servicios web con J2EE. JAX-RPC provee una plataforma simple y robusta para construir aplicaciones que usen servicios web ocultándole al desarrollador la complejidad de mapear entre tipos de datos XML y los tipos de datos Java, además de los detalles de bajo nivel del manejo de los mensajes SOAP.

JAX-RPC introduce un nuevo paradigma ofreciendo dos modelos de programación: un modelo del lado del servidor para desarrollar servicios web proveedores usando clases Java o componentes EJB de sesión sin estado, y un modelo del lado del cliente para crear consumidores que accedan a los servicios web como si fueran objetos locales.

JAX-RPC 1.1 obliga el uso de SOAP 1.1 y la interoperabilidad con servicios web hechos en otras tecnologías tales como Microsoft .NET. Varios servidores de aplicaciones compatibles con J2EE 1.4, como Oracle Application Server Containers for J2EE (OC4J) 10.1.3 , Sun One Application Server e IBM WebSphere V6, soportan JAX-RPC.
Si ya tiene EJBs de sesión sin estado o una clase Java que realiza la lógica de negocio, J2EE 1.4 permite exponerla como un servicio usando JAX-RPC. Si está creando servicios web usando J2EE, puede que necesite varios elementos adicionales al estándar WSDL:

- Un archivo de mapeo, es un XML que especifica los mapeos Java - WSDL. En la mayoría de los casos este archivo es requerido sólo para los mapeos más complejos o personalizados.

- Una interface que el servidor J2EE usa para exponer el servicio web proveedor.
  La interface debe heredar de java.rmi.RemoteInterface, como en el siguiente ejemplo:

```
package time;
import java.rmi.RemoteException;
import java.rmi.Remote;
public interface TimeService extends Remote{
public   String   getDateTime   (String   name)   throws
RemoteException;
}
```

- Un descriptor de despliegue de servicios web como *webservices.xml,* el cual describe la ubicación de la interface proveedora, el WSDL y el archivo de mapeo Java – WSDL.

- Un descriptor específico de despliegue, tal como *oracle- webservices.xml* para Oracle Application Server.

A primera vista, estos requisitos para un servicio parecen desalentadores. Sin embargo, la mayoría de vendedores proveen herramientas que facilitan el desarrollo de los servicios.

Por ejemplo, Oracle ofrece facilidades de desarrollo de servicios web dentro de Oracle Jdeveloper10g, o WebSphere Application Developer Studio de IBM simplifica el desarrollo de aplicaciones SOA.

Finalmente, como se mencionó anteriormente, la interoperabilidad es uno de los aspectos clave de las aplicaciones orientadas a servicios. Si desea que sus servicios sean interoperables, asegúrese de probarlos no sólo con plataformas J2EE sino también con otras como .NET.

Realmente no se puede anticipar quienes serán sus consumidores, después que sus servicios estén desplegados y registrados en un servicio de directorios (tal como UDDI) cualquiera puede encontrarlos y usarlos. Así, cuando haya creado sus servicios, asegúrese que cumpla el WS-I Basic Profile.

WS-i es un consorcio abierto de plataformas SOA de proveedores como Oracle, IBM, Microsoft y Sun que se enfocan en la interoperabilidad de los servicios web entre las plataformas, sistemas operativos y lenguajes. J2EE 1.4 requiere compatibilidad con WS-I Basic Profile, si esto se cumple, su servidor de aplicaciones debería generar servicios interoperables. *[www37]*

3.7.2 Capa de Procesos de Negocio

Otra promesa de SOA es poder crear aplicaciones a partir de servicios existentes. El principal beneficio que ofrece SOA es la estandarización del modelado de procesos de negocio, generalmente llamado orquestación de servicios. Se puede hacer una capa de servicios web basada en la abstracción de sistemas legacy aprovechándolos para ensamblar procesos de negocio.

Adicionalmente, los vendedores de plataformas SOA poseen herramientas y servidores para diseñar y ejecutar estos procesos. Este esfuerzo ha sido estandarizado por el consorcio OASIS bajo el nombre de BPEL (Business Process Execution Languaje o Lenguaje de

Ejecución de Procesos de Negocio); la mayoría de proveedores de plataformas se han acogido a este estándar.

BPEL es esencialmente un lenguaje de programación pero se representa en XML.
Este es un ejemplo de un proceso definido con BPEL:

```
<process>
 <!– Definición y roles de los procesos participantes >
 <partnerLinks> ... </partnerLinks>

 <!Datos/estados definidos dentro de los procesos >
 <variables> ... </variables>

 <!Properties that enable conversations >
 <correlationSets> ... </correlationSets>

 <!Manejo de excepciones >
 <faultHandlers> ... </faultHandlers>

 <!Recuperación de errores – deshacer acciones >
 <compensationHandlers> ... </compensationHandlers>

 <!Eventos concurrentes>
 <eventHandlers> ... </eventHandlers>

</process>
```

Estructura de BPEL:

· *partnerLinks* para los servicios con los que interactúa el proceso.
· *variables* para manipular los datos.
· *correlationSets* para relacionar mensajes entre invocaciones asíncronas
· *faultHandlers* definición de mensajes para los errores.
· *compensationHandlers* se ejecutan en caso de problemas.
· *eventHandlers* permiten que el proceso interactúe con eventos anticipados en casos exitosos.

Aunque la sintaxis de BPEL es más o menos sencilla, es preferible una representación gráfica de estos procesos de negocio y será mucho más sencillo ensamblarlos a partir de los

servicios existentes. De esta forma el crear procesos de negocio es una tarea relativamente sencilla si se comprenden y hay servicios disponibles para ser usados.

Oracle BPEL Designer , puede ser usado como plugin en Eclipse o JDeveloper para ayudarlo a crear procesos facilitando el diseño y desarrollo de servicios.

Adicionalmente, necesitará un servidor o un entorno de procesamiento de alto rendimiento para ejecutar los procesos de negocio generados, y una herramienta para monitorear y probar los servicios desplegados. La mayoría de vendedores de 9 plataformas SOA como Oracle o IBM tienen un robusto conjunto para desplegar procesos de negocio. Por ejemplo, Oracle provee Oracle BPEL Process Manager para desplegar y ejecutar procesos de negocio y Oracle BPEL Console para probarlos y monitorearlos. *[www37]*

3.7.3 La Capa de Presentación: El problema de asociación de datos

La capa de presentación es sumamente importante desde la perspectiva del usuario.
Esta capa puede ser hecha con tecnologías como JSP, JSF, portlets o clientes Java independientes. Para los desarrolladores, es una cruzada conseguir desacople entre la capa de presentación y la lógica de negocio o la capa de servicios.

Varios frameworks MVC, bastante usados brindan este desacople. Esto permite cambiar cualquiera que sea la vista o el modelo con un mínimo impacto, permitiéndonos alcanzar el desacople que deseamos entre la presentación y la capa de servicios. El principal problema es que no hay un estándar para enlazar los datos entre los diferentes tipos de clientes (como JSP, clientes Java, y servicios como EJB o servicios web), y éstos deben conocer exactamente la tecnología subyacente en la implementación de la capa de servicios.

Por ejemplo los servicios hechos ya sea con servicios web o EJBs puros, tienen diferentes interfaces para iniciar ya sea el servicio o la transferencia de datos, lo que *derrota* el objetivo de SOA. El JSR-227 pretende estandarizar el enlace de datos de los servicios en aplicaciones J2EE definiéndolo a través de un conjunto de definiciones de metadatos basados en XML.

El JSR-227 permitirá a los desarrolladores crear controles de datos usando servicios del negocio, los cuales podrán ser un servicio web, EJB de sesión sin estado o clases Java

estándar. Los controles de datos describirán las colecciones (atributos y operaciones soportadas por el servicio usando metadatos XML). Estos controles de datos posteriormente pueden ser usados por componentes de interfaces de usuario por medio de la declaración de enlaces y de esta forma se logrará el desacople completo entre la capa de presentación y la de servicios. *[www37]*

3.8 Inconvenientes a tener en cuenta en el diseño SOA

La implementación ideal de un servicio exige resolver algunos inconvenientes técnicos inherentes a su modelo:

- Los tiempos de llamado no son despreciable, gracias a la comunicación de la red, tamaño de los mensajes, etc. Esto necesariamente implica la utilización de la mensajería confiable.
- La respuesta del servicio se ve afectada directamente por aspectos externos como problemas en la red, configuración, etc. Estos se deben tener en cuenta en el diseño, desarrollándose los mecanismos de contingencia que eviten le parada de las aplicaciones y servicios que dependen de él.
- Debe manejar comunicaciones no seguras, mensajes imprescindibles, reintentos, mensaje fuera de secuencia, etc.

Según lo anterior, se puede ver que la construcción de un servicio es una tarea mucho mas complicada que la de un simple componente distribuido.

El servicio debe publicar una interfaz (por ejemplo, utilizando WSDL o proxies) fácilmente localizable en la red. Esta interfaz debe servir como un contrato de servicio. Donde se describen cada una de las funciones que provee, e incluso los niveles de prestación de servicio (SLA). Esta interfaz se debe documentar de forma clara, de manera que sea muy fácil implementar una conexión.

Un diseño exitoso de una arquitectura basada en servicios debe estar basado en una plataforma de mensajería segura, que aísle de la implementación funcional muchos de los problemas anteriormente mencionados. Algunas de las responsabilidades de un mecanismo así, incluyen:

- Entrega garantizada de mensajes.

- Enrutamiento de peticiones a un servicio disponible
- Seguridad del contenido de los mensajes
- *QoS* (quality of service) Calidad del Servicio
- Escenarios fuera - de – línea

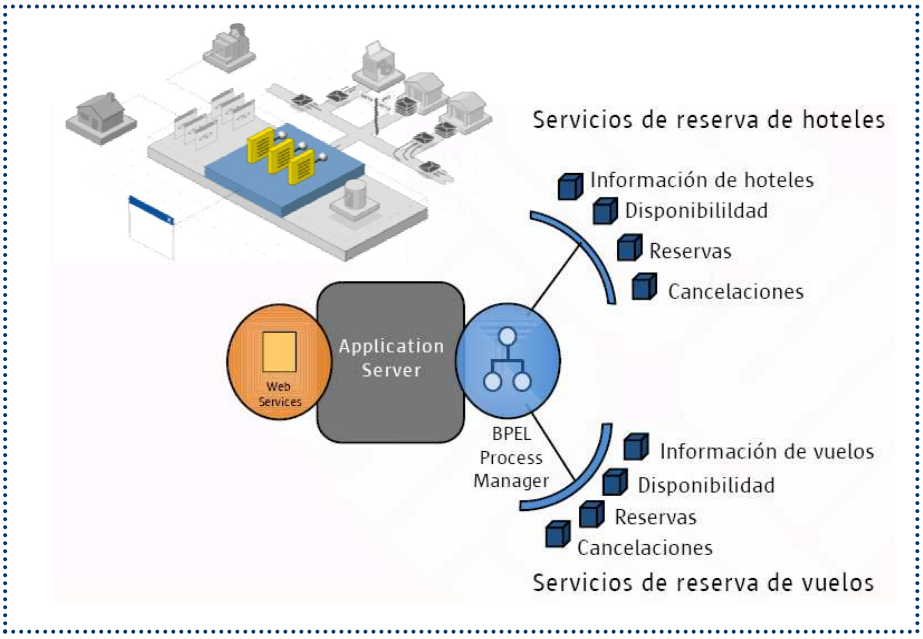
Cuantas más de las características mencionadas soporte la tecnología elegida, menos problemas tendrá la solución final en operación.

Una comunicación basada en mensajes por lo general implica que no existen sesiones. Por lo tanto, los clientes no guardan estado en el servicio (mayor escalabilidad), y la autenticación se debe dar a nivel de cada mensaje.

Por último, los servicios deben ser diseñados para controlar internamente la transaccionalidad de sus propias operaciones. No es recomendable que una transacción traspase los límites se un servicio. *[www36]*

CAPITULO 4

ORQUESTACIÓN DE SERVICIOS



"El Señor se deleitará en tí con gozo, te renovará con su amor... como en los días de fiesta"

SOF. 3:17

4.1 INTRODUCCION

Un proceso de negocio es una actividad del mundo real que consta de un conjunto de tareas lógicamente relacionadas, que cuando se realizan en la secuencia apropiada y siguiendo las reglas del negocio, producen una salida válida para el negocio (realizar un pago, realizar una extracción de efectivo de una cuenta bancaria, etc.)

BPM (Business Process Management) es el nombre de un conjunto de sistemas de software, herramientas y metodologías enfocadas hacia la manera en que las organizaciones identifican, modelizan, desarrollan, distribuyen y gestionan tales procesos de negocios. Entre las soluciones BMP conocidas se cuentan desde los workflow hasta los servicios de orquestación Web modernos.

SOA (Service Oriented Architecture) basado en Servicios Web, ayudan a alcanzar el objetivo de BPM más rápida y fácilmente. Desde el punto de vista opuesto, BMP simplifica el problema de cómo combinar la ejecución de múltiples Servicios Web para resolver un caso de negocio particular. BPM aísla no solamente el proceso respecto de los datos sino que explicita el flujo del mismo y permite separarlo de la forma en que se ejecuta. Esta característica hace que BPM tienda a adoptarse como metodología para conceptualizar problemas más allá de lo tecnológico.

En los orígenes de la ciencia de la computación todo el énfasis estuvo puesto en desarrollar sistemas que automaticen tareas que se hacían manualmente. Este era suficiente desafío. La tendencia hoy es hacia un paradigma orientado a procesos, donde las aplicaciones deben cubrir la actividad global de la empresa y las herramientas son los BPMS (Business Process Management Sistemas o Sistemas de Gestión de Procesos de Negocio.

Los Servicios Web son procesos de negocio modulares y autocontenidos que se basan en tecnologías de estándares de la industria, pero ninguno de estos estándares les otorga semántica de negocio. Para otorgar esta semántica es necesario especificar restricciones acerca del uso de las operaciones de de los Servicios Web y su comportamiento, es decir, especificar procesos de negocio.
BPEL (Business Process Execution Language) es un lenguaje de especificación para representar flujos de procesos de una manera adecuada para que una máquina BPEL pueda

leerla e interpretarla. Es decir es un lenguaje de especificación completamente ejecutable. *[www39].*

Este capítulo presenta una visión integradora de varios conceptos tecnológicos que van de la mano en el proceso de implementación de SOA.

4.2 BPM (BUSINESS PROCESS MANAGEMENT).

BPM (Gestión de Procesos de Negocio).Un proceso de negocio es una actividad del mundo real que consiste en un conjunto de tareas lógicamente relacionadas que si se realizan en la secuencia apropiada y de acuerdo a las reglas de negocio correctas, producen una salida de negocio.

BPM determina la manera en que una organización puede identificar, modelar, desarrollar, distribuir y administrar sus procesos de negocio, incluyendo aquellos procesos que involucran sistemas de IT e interacción humana.

BPM tiene sus raíces en el workflow y ha progresado sobre los sistemas de orquestación de WS (Web Services o Servicios Web) actuales.

Los objetivos y beneficios de BPM son:

- Reduce los errores de obstrucción entre requerimientos de negocio y los sistemas de IT ya que los usuarios de negocio modelan los procesos y luego el departamento de IT provee la infraestructura para ejecutarlos.
- Incrementa la productividad de los empleados.
- Incrementa la flexibilidad y agilidad corporativa separando la lógica del proceso de otras reglas de negocio. Esto absorbe mejor los cambios en los requerimientos.
- Reduce los costos de desarrollo con lenguajes de programación grafica de alto nivel. *[www08]*

4.3 Orquestación de Servicios

Al ser componentes, los servicios Web pueden verse como cajas negras con la capacidad de ser utilizadas y reutilizadas sin necesidad de conocer como fue implementada su funcionalidad. Dicha propiedad facilita la composición de servicios.

El proceso de orquestación consiste en "*relacionar, organizar y administrar las interacciones entre los servicios Web referentes a la lógica del proceso de negocios" [www39].*

Los elementos básicos para llevarla a cabo son:

1. Procesos. Vistos como una serie de actividades cuyo fin es la ejecución de una tarea determinada, por ejemplo, un plan de viaje, la compra de artículos electrónicamente, etc.

2. Actividades. Representan reglas bien definidas del proceso de negocios. Por ejemplo en un proceso de plan de viaje deberían ser contempladas las actividades: consulta de hospedaje y reservación de boletos de avión, entre otras.

3. Flujo de datos. Describe la información intercambiada entre actividades.

4. Flujo de control. Describe el orden en que serán ejecutadas las actividades del flujo. Dicho orden sería especificado en términos de construcciones usuales de programación como la composición: secuencial, concurrente y condicional.

Típicamente la orquestación de servicios Web se ejecuta por un sólo nodo coordinador, el cual sería responsable de administrar el flujo de datos y el flujo de control entre los componentes (servicios Web). Es decir, éste recibiría las peticiones de los clientes, haría las transformaciones de los datos requeridos e invocaría los componentes en base a su especificación. A este modo de ejecución se conoce como *orquestación centralizada*, la cual tiene como principal característica que todos los datos transferidos entre componentes se realizan vía el coordinador, en lugar de ser transferidos directamente desde el punto de generación al punto de consumo. *[www40]*

En la figura 4.1 podemos observar gráficamente cómo se lleva a cabo la orquestación de servicios Web, donde el rectángulo mayor representa el proceso que será orquestado, los rectángulos sombreados pequeños las actividades involucradas en el mismo, las flechas horizontales el flujo de datos intercambiado y las flechas verticales el flujo de control del proceso.

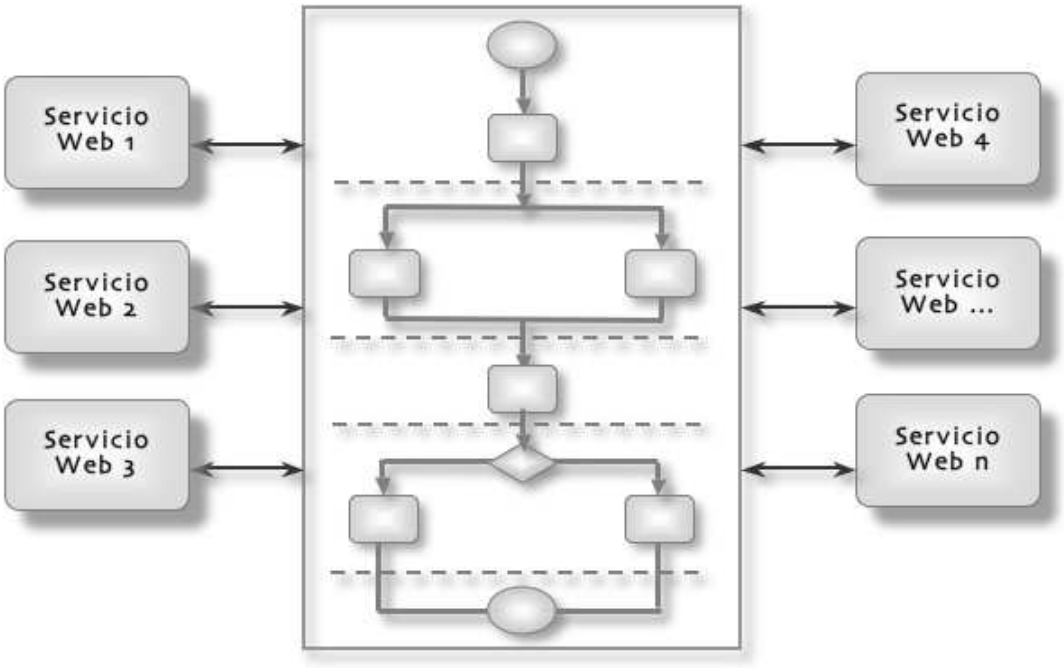


Figura 4.1 Orquestación de Servicios Web

Al orquestar servicios, como primer paso necesitaremos elegir un lenguaje de coordinación que cumpla con un conjunto de características deseables para coordinar correctamente los componentes involucrados. Dicho lenguaje debería *ser simple, abstracto, dar soporte a enlace dinámico de recursos y a actividades primitivas y estructuradas* y además debería garantizar *persistencia y correlación* a lo largo de las transacciones del flujo.

Finalmente, como segundo paso sería necesario definir una *infraestructura de coordinación*, la cual ofrecería todos los mecanismos necesarios en cómputo para dar soporte a las construcciones del lenguaje y además llevaría a cabo el descubrimiento, publicación y ejecución de servicios Web orquestados. *[www40]*

4.3.1 BPEL Business Process Execution Language - *Lenguaje de Ejecución de Procesos de Negocio*

La composición de servicios Web es un nuevo paradigma que intenta incrementar el potencial de las aplicaciones Web. Diversos proveedores de tecnologías de información como IBM, Microsoft y Sun MicroSystems han diseñado lenguajes y técnicas para implementar dicho paradigma. Sin embargo el representante más significativo en nuestros días es BPEL4WS o BPEL *(Business Process Execution Language for Web Services)* desarrollado por IBM, Microsoft y BEA Systems *[lib04].*

BPEL es un lenguaje estático composicional orientado al modelado de procesos orquestados, el cual cuenta con su propio motor de ejecución conocido como BPWS4J.

BPEL distingue entre procesos abstractos y ejecutables. Los procesos abstractos definen protocolos de negociación, especificando los mensajes intercambiados entre los diferentes participantes del proceso orquestado, mientras que los procesos ejecutables especifican el orden de ejecución de los elementos que constituyen dicho proceso (proveedores de servicios, mensajes intercambiados, manejo de exepciones, etc.).

Los procesos abstractos son útiles para describir protocolos de comunicación, mientras que los procesos ejecutables deben ser compilados para obtener servicios invocables.

Otras características de BPEL son: el uso de referencias a tipos de puertos *(portType Interface abstracta que define el conjunto de operaciones asociadas a un servicio Web.)* contenidos en documentos WSDL, el manejo de compensación de transacciones y la disponibilidad de un mecanismo de captura y gestión de excepciones similar al proporcionado por Java.

BPEL está relacionado con otras dos especificaciones:

- WS-Coordination (WS-C) y
- WS-Transaction (WS-T).

WS-Coordination soporta, integra y unifica diferentes modelos de coordinación permitiendo a diferentes sistemas interoperar. Provee además mecanismos estándares para crear y registrar servicios usando los protocolos definidos en WS-T.

WS-Transaction coordina la ejecución de operaciones distribuidas en el ambiente de servicios Web a través de la definición de protocolos para transacciones atómicas, de negocios, compensación, etc. *[www35]*

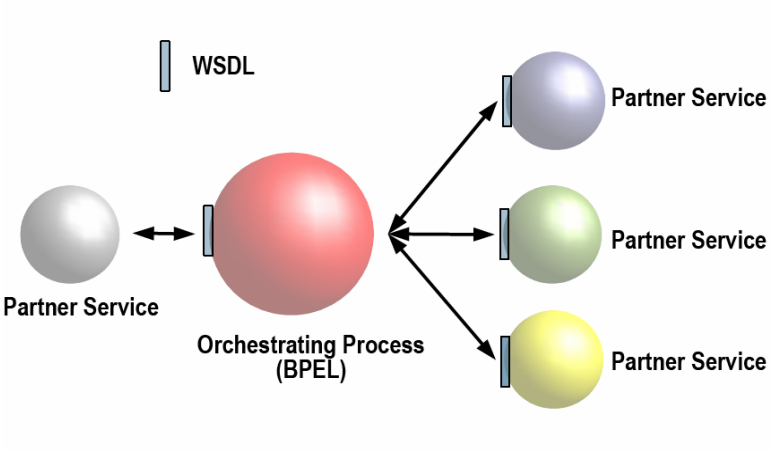


Figura 4.2 BPEL Relación con Socios

4.3.1.1 Elementos de BPEL

BPEL posee los siguientes elementos:

1. Socios ("Partners"): Entidades.
2. Variables.
3. Actividades.
4. Manejadores:
    - Manejadores de eventos.
    - Manejadores de fallos (excepciones).
    - Manejadores de compensación (operaciones deshacer).

Un Flujo BPEL4WS ofrece a su vez una interfaz Web Service hacia el exterior.
Normalmente un proceso BPEL4WS se compone de:

Un fichero con el proceso a ejecutar (.BPEL)
Una serie de ficheros WSDL de apoyo (definiciones).

4.3.1.1.1. Socios

Los socios representan a los servicios web invocados por el proceso. Se basan en tres elementos:

- Tipo de Enlace del Socio (Partner Link Type):
  - Define un enlace genérico para una categoría de servicios web.
  - Similar a la definición de una clase en lenguajes OO.
  - Ejemplo: obtención de cartelera, búsqueda de libros, compra de libros.

- Enlace del Socio (Partner Link):
  - Define el servicio web que realmente se invocará.
  - Similar a una instancia de una clase en lenguajes OO.
  - Amazon_search (Enlace del tipo "búsqueda de libros"),
  - Amazon_purchase (enlace del tipo "compra de libros"),
  - Cinentradas_get (enlace del tipo "obtener cartelera").

- Socio (Partner):
  - Pueden usarse para agrupar todos los enlaces del mismo "socio físico".
  - Socio "Amazon" agrupa los enlaces Amazon_search y Amazon_purchase.

"Partner Link Types". En lugar de definir la relación entre dos servicios web en términos de "proceso" y "servicio externo" (que adopta el punto de vista del servicio "proceso"), se define de forma neutra. Define una colección de roles. Cada rol indica una lista de tipos de puerto (portTypes). Un servicio web puede jugar ese rol si implementa esos tipos de puerto. Un enlace de socio (Partner Link) es especificado dándole un nombre, indicando un "partner link type" y especificando el rol jugado por el partner en ese "partner link type".

Definición de un PartnerLinkType:

Normalmente se definen en ficheros WSDL utilizados por el proceso BPEL, y no directamente en el proceso BPEL.

Definición tipo de puerto (WSDL externo):

```
<portType name="bookSearchPT">
     <operation name="bookSearch">
          <input message="searchdef:bookSearchMessage"/>
          <output message="searchdef:bookSearchResults"/>
     </operation>
```

```
</portType>
```

Definición PartnerLinkType (WSDL externo)

```
<plnk:partnerLinkType name="bookSearchLinkType">
     <plnk:role name="bookSearchEngine">
          <plnk:portType name="apns:bookSearchPT"/>
     </plnk:role>
</plnk:partnerLinkType>
```

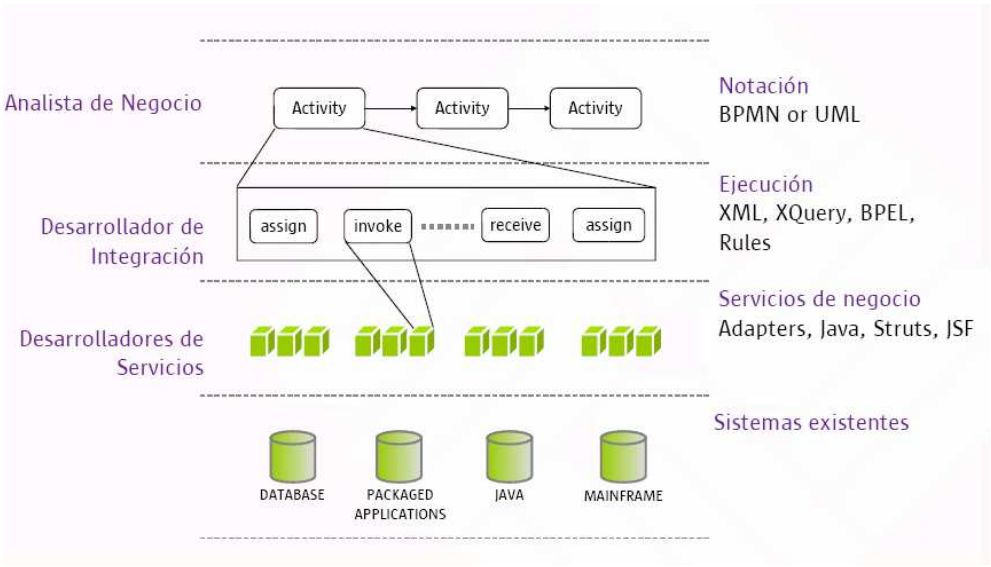


Figura 4.3 componentes de BPEL

4.3.1.1.2. Variables

Las variables se utilizan para guardar los datos utilizados dentro del proceso. Variables pueden ser:

- Mensajes completos definidos en la especificación WSDL del servicio que usa dicho mensaje.
- Un valor de un tipo básico XML (enteros, flotantes, etc.)
- Elementos compuestos XML.

El tipo de las variables se define en la especificación WSDL del servicio (o en otros WSDL externos). Ejemplos:

```
<variable name="searchRequest"
     messageType="searchdef:bookSearchMessage"/>
<variable name="searchResponse"
     messageType="searchdef:bookSearchResults"/>
<variable name="numResults"
     type="xsd:int"/>
<variable name="searchResult"
     element="searchdef:searchResult"/>
```

4.3.1.1.3. Actividades

En BPEL se pueden modelar las siguientes actividades a ser llevadas a cabo en un determinado proceso.

Actividades primitivas:

- Invoke. Invocar una operación en un servicio web.
- Receive. Esperar a la recepción de una invocación sobre una operación del sitio web.
- Reply. Generar la respuesta de una operación entrada/salida.
- Wait. Esperar un tiempo.
- Assign. Asignar valores a variables.
- Throw. Lanzar excepciones.
- Terminate. Terminar la instancia.
- Empty. No hacer nada.

Actividades Estructuradas:

Las actividades primitivas pueden ser combinadas utilizando las siguientes estructuras de control:

- Sequence. Secuencia de actividades.
- Flow. Ejecutar actividades en paralelo. Las restricciones de orden pueden especificarse mediante "links" (enlaces).
- Switch. Estructura condicional para escoger entre un conjunto de actividades.
- While. Creación de bucles.
- Pick. Ejecutar uno de varios caminos alternativos, en función de que se produzca un evento u otro.
- Scope. Define un bloque de actividades (utilizado en transacciones).
- Compensate. Define las actividades de un bloque de compensación.

4.3.1.1.4. Manejadores

- Manejadores de fallos (Fault Handlers). Se ejecutan cuando se lanza una excepción. Similar al manejo de excepciones en lenguajes de programación.
- Manejadores de compensación (Compensation handlers). Se ejecutan para deshacer una operación. Relacionados con el tratamiento de transacciones.

- Manejadores de eventos. Se ejecutan cuando se recibe un mensaje particular o se produce una determinada alerta. Ej: comprador envía un mensaje de cancelación.

Los manejadores deben asociarse a un ámbito ("scope"), concepto muy similar al de "bloque de código" en los lenguajes de programación:

> Si salta una excepción dentro de un scope determinado, se invoca el correspondiente manejador de fallo.
>
> Si hay que deshacer los efectos de un scope (e.g. transacción), se llama al manejador de compensación del bloque. Si hay bloques anidados, se llama a sus manejadores de compensación en orden inverso de ejecución.
>
> Los manejadores de eventos están activos mientras el flujo de control permanece dentro del bloque. Se ejecutan si llega un determinado mensaje o se da una cierta condición.*[www31].*

Las aplicaciones creadas con BPEL están basadas en procesos. Se divide la aplicación en dos niveles claramente separados: el nivel superior de los procesos de negocio escritos en BPEL y que representan el flujo de la lógica de aplicación, mientras que a nivel inferior, los WS representan la funcionalidad lógica de la aplicación.

Durante su tiempo de vida, la instancia del proceso de negocio mantiene conversaciones con su "partner". En este caso se requiere un mecanismo a nivel de aplicación que aparee mensajes con conversaciones. BPEL dirige los escenarios de correlación dando un mecanismo declarativo para especificar conjuntos de correlación.

Se puede tener dos visiones de BPEL: por un lado puede verse como un script XML que una máquina de procesos de negocio ejecuta. Pero por otro lado puede verse como un lenguaje de intercambio, o sea la máquina de procesos de negocio permite convertir un lenguaje propietario a BPEL y viceversa.

4.4 Principales tecnologías involucradas en los WS

Los Servicios Web engloban una serie de tecnologías XML que se encargan de solucionar problemas concretos de interoperación. La mayor parte de ellas nacieron de empresas privadas (normalmente un consorcio de varias empresas a las que se iban uniendo otras a medida que la necesidad de la tecnología era más patente).
Muchas de estas tecnologías han sido remitidas a organismos de estandarización, principalmente W3C y OASIS, que o bien las han rechazado, o bien han creado un grupo de trabajo para convertirlas en recomendaciones o estándares.

Especificaciones WS-

A continuación describo las principales especificaciones relacionadas con los Servicios Web disponibles en la actualidad. W3C y OASIS mantienen una relación exhaustiva de informes técnicos, borradores, recomendaciones y estándares relacionados con estas tecnologías.

SOAP Especifica la estructura de los mensajes que los WS intercambian. Es independiente de la plataforma, flexible y fácilmente extensible.

WSDL Permite la descripción de WS: estructura de los mensajes SOAP que intercambiará con otros WS, servicios que ofrece, negociación de los parámetros de seguridad en las comunicaciones entre WS, etc.

UDDI Permite mantener repositorios de especificaciones WSDL simplificando el descubrimiento de WS y el acceso a sus especificaciones. Hace posible que una aplicación busque dinámicamente servicios que ofrezcan una serie de características, seleccione el más adecuado (por coste, calidad, etc.) e incluso localice servicios alternativos si uno falla.

WS-Addressing. Permite incluir en un mensaje SOAP información sobre el emisor, los destinatarios, a quién se debe responder, a quién se debe informar en caso de error, etc. Con estos elementos se consigue un direccionamiento independiente de capas de transporte inferiores como HTTP.

WS-AtomicTransaction Suele emplearse en la coordinación de WS que realicen actividades de corta duración en entornos de confianza. Las acciones a realizar por cada servicio implicado se agrupan en una transacción atómica. El coordinador decide cuándo realizarla y puede abortar una transacción en curso devolviendo a los WS implicados a su estado previo.

WS-BPEL Define un lenguaje que facilita la composición de WS. Permite especificar la lógica de la composición de los servicios (envío de mensajes, sincronización, iteración, tratamiento de transacciones erróneas, etc.) independientemente de su implementación.

WS-BusinessActivity Permite la coordinación de WS compuestos (normalmente con WS-BPEL) a partir de actividades independientes que no se pueden modelar como transacciones atómicas por su duración, por requerir intervención humana o por ser incapaces de bloquear recursos.

WS-Coordination Permite crear contextos de coordinación para la sincronización de WS. Requiere protocolos complementarios como WS-AtomicTransaction o WS-BusinessActivity.

WS-DistributedManagement Permite gestionar recursos distribuidos de todo tipo (PDA, televisores, dispositivos de conexión de redes, etc.) mediante WS.

WS-Notification Permite que un WS reciba información puntual sobre determinados acontecimientos y está formado por tres especificaciones: WS-BaseNotification, WS-BrokeredNotification y WS-Topics.

WS-Policy Proporciona un medio de especificar las características que presentan y exigen los WS durante su operación. Por ejemplo, un determinado servicio puede exigir para operar que los mensajes se firmen o cifren con determinados algoritmos o que la coordinación se realice mediante un protocolo dado. Con esto se dota a los WS de la capacidad de negociar entre ellos las condiciones de interacción.

WS-Reliability Es un protocolo que permite numerar los mensajes SOAP y obtener confirmación de su recepción en destino. Con esto se puede garantizar el orden de recepción de los mensajes, evitar duplicados, comprobar la entrega, etc.

WS-ReliableMessaging Esta especificación define un protocolo que permite el intercambio de mensajes de manera fiable en presencia de fallos en el software, la red, etc.

WS-ReliableMessagingPolicyAssertion Con este protocolo se puede manejar políticas WS-Policy que expresen los requisitos de los emisores y receptores de mensajes que usen WS-ReliableMessaging.

WS-ResourceFramework Mediante este protocolo se pueden modelar y utilizar servicios con estado interno. Permite superar las limitaciones que poseen las tecnologías WS-_ respecto a REST, que permite contemplar servicios con estado interno.

WS-SecureConversation Define extensiones de WS-Security que proporcionan un marco de trabajo en el que se pueden solicitar y emitir *tokens* de seguridad, de manera que los agentes implicados puedan establecer relaciones de confianza mutua.

WS-Security Proporciona integridad, confidencialidad y autenticación en las comunicaciones entre WS. Incluye varios protocolos de seguridad, como X.509 y Kerberos, de manera que los WS puedan utilizar distintas políticas de seguridad.

WS-Trust Su objetivo es facilitar el intercambio de series de mensajes seguros mediante la emisión, renovación y validación de *tokens* de diversos protocolos. *[doc01]*

CAPITULO 5

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE GESTIÓN DE HISTORIAS CLÍNICAS PARA EL DEPARTAMENTO DE BIENESTAR UNIVERSITARIO DE LA UTN

"Todo lo puedo en Cristo que me fortalece"

FIL. 4:13

5.1 INTRODUCCIÓN

"Contribuir al desarrollo educativo, científico, tecnológico, socioeconómico y cultural de la región norte del país. Formar profesionales críticos, humanistas y éticos comprometidos con el cambio social". Es la Misión Institucional que la Universidad Técnica del Norte cumple tan acertadamente para con la comunidad norteña de nuestro país.

Queriendo contribuir la confianza depositada por la comunidad y caminar a la par del avance tecnológico. La universidad se encuentra en un proceso de desarrollo y actualización tecnológica, siendo uno de sus objetivos la automatización y reingeniería de varios de los procesos que se realizan cotidianamente.

Particularmente los servicios que presta el Departamento de Bienestar Universitario, están encaminados al cuidado de la salud, a la orientación y la asistencia social de toda la comunidad universitaria.

Dichos servicios conforman procesos, que se realizan manualmente y de poca fluidez. Es así que surge el interés por parte de la Directora de este departamento y con el apoyo del personal del Departamento de Informática el proyecto de automatización de Gestión de Historias Clínicas y Odontológicas y la reingeniería del Sistema de Fichas Socioeconómicas.

Siendo mi responsabilidad el desarrollo del Sistema de Gestión de Historias Clínicas. En este capítulo presento el proceso de su desarrollo haciendo uso de la tecnología de mi investigación.

5.2 ESTUDIO DE VIABILIDAD

El Departamento de Bienestar Universitario de la Universidad Técnica del Norte es una dependencia que contribuye al desarrollo socio económico, académico y de salud del personal estudiantil, docente y administrativo de la institución.

Esta dependencia busca prestar servicios a los miembros de la comunidad universitaria con personal idóneo, para ser un punto de apoyo en el logro del mejoramiento de la calidad de vida y la búsqueda de la excelencia académica.

Coherente con el objetivo de la universidad de formar integralmente nuevos profesionales que deberán ser los protagonistas del desarrollo de nuestro país, cumple un rol fundamental. Esta acción de carácter primordial se traduce básicamente en la promoción, organización y realización de actividades de salud y formación humana.

Diariamente el Departamento de Bienestar presta sus diversos servicios, debiendo mantener información exclusiva de cada una de las personas que acuden, como son: estudiantes (7106), docentes (326), empleados (324) y familiares (112) *[Sistema Recaudación SARE]*; cuyo volumen de información representa una gran responsabilidad.

Tomando en consideración lo anterior, es necesario contar con herramientas de software que garanticen el correcto tratamiento de la información y ayuden a proporcionar un buen servicio. *[doc02].*

5.2.1 Descripción Del Problema

El departamento de bienestar universitario está conformado por cuatro áreas: Salud, Asistencia Social, Orientación Profesional, Orientación Académica.

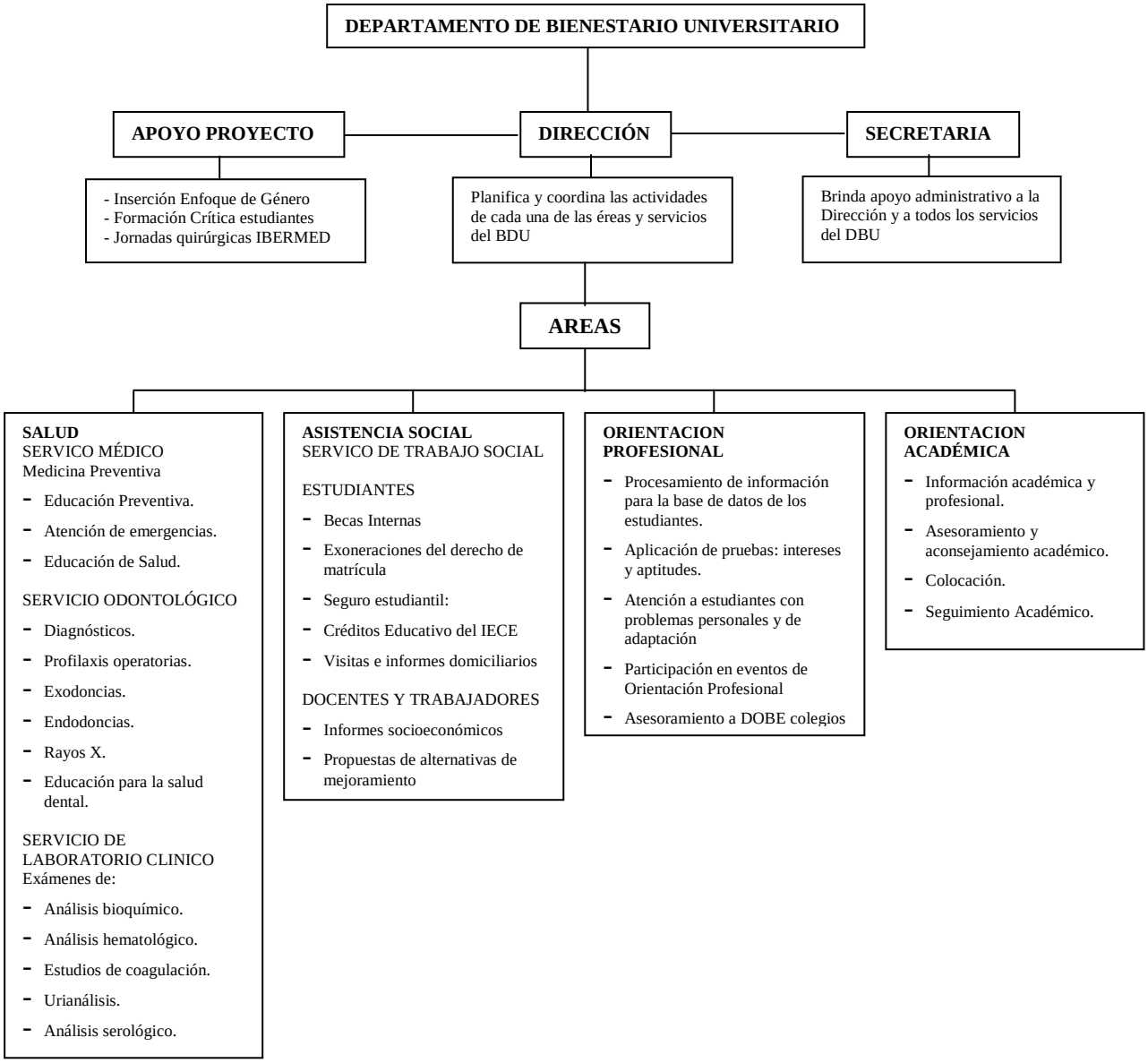


Figura 5.1 Diagrama Orgánico Estructural del DBU

ÁREA DE SALUD

El área de Salud a su vez está estructurado de la siguiente forma: Servicio Médico, Servicio Odontológico y Servicio de Laboratorio Clínico.

Esta área presta atención personalizada a estudiantes, docentes, personal administrativo y sus familiares, tanto del colegio como de la universidad.

El área de Salud maneja toda la información de sus pacientes en formularios, hojas de trabajo, registros diarios, registros mensuales y anuales manualmente; además lleva un

registro de ingresos y egresos de medicamentos y materiales mediante cartones de kardex. La inexistencia de un sistema automatizado en esta área, ha ocasionado una serie de inconvenientes tales como:

- **Información aislada y redundante.-** Al no existir un control de historias clínicas automatizado impide el acceso a la información que mantiene el servicio de Laboratorio Clínico el cual cuenta con un sistema de ingreso de resultados. Además la información existente en el sistema de Ficha Psicosocial es redundante a la que contiene la ficha historia clínica.

- **Falta de agilidad en la atención al paciente.-** La historia clínica de cada paciente está constituida por varios formularios llenados a mano y archivados en un mueble, de donde son extraídos al momento de iniciar la consulta. Este tedioso proceso requiere mayor atención y tiempo, ocasionando molestias tanto al personal del departamento como a los pacientes.

- **Dificultad en la generación de informes y estadísticas.-** Al no existir un proceso automático para registrar la atención al paciente en el departamento se lleva diariamente un registro manual que luego es contabilizado para informes mensuales, lo que requiere gran cantidad de tiempo. Además para la generación de cuadros estadísticos, es necesario que el personal del departamento obtenga los datos examinando cada uno de los formularios de historias clínicas de los pacientes existentes; siendo ésta una tarea compleja y poco precisa.

- **Pérdida de información.**- Los datos de historias clínicas y los resultados de exámenes es información confidencial y de vital importancia, sin embargo es manipulada por varias personas, y archivada en lugares poco propicios, lo cual facilita su pérdida.

- **Desconocimiento de medicamentos y materiales existentes.**- El servicio médico y odontológico manejan cierta cantidad de medicamentos de emergencia, los cuales son contabilizados manualmente mediante kárdex sin embargo no se lleva un control adecuado del mismo dado que los datos no son actualizados constantemente. Además el servicio de laboratorio y el odontológico manejan materiales de uso exclusivo los mismos que no son inventariados. *[doc02].*

5.2.2 Descripción general del sistema

Como solución a las necesidades que posee el servicio médico del departamento de bienestar universitario, planteo la realización de los siguientes módulos:

HISTORIAS CLINICAS

- Identificación del Paciente a través de un único identificador de pacientes.
- Manejo de historias clínicas a través de formularios predefinidos con datos personales, antecedentes médicos familiares y personales, signos vitales, examen físico, motivo de consulta, diagnostico clasificado por sistemas, tratamiento y observación.
- La historia clínica consultará resultados de exámenes generados por el módulo de laboratorio.
- A la Historia Clínica de cada paciente se podrá adosar una cantidad ilimitada de imágenes del mismo, resultados de exámenes realizados en otros laboratorios, ecografías, etc.
- Emisión de órdenes de Laboratorio.
- Luego de la consulta podrá digitar la prescripción e imprimirla.

INVENTARIO DE MEDICAMENTOS

- Sistema de inventario de medicamentos de emergencia, que estará a disposición del médico, proporcionando un reporte preciso de los medicamentos existentes en el momento de la consulta.
- Se descargarán automáticamente los medicamentos prescritos al paciente, sustentando el egreso mediante la impresión y firma de un comprobante.

ENFERMERÍA

- Módulo en el que se registrará actividades que desempeña la enfermera como curaciones, colocación de inyecciones, etc. adjuntándose a la historia de cada paciente.

REPORTES Y ESTADÍSTICAS

- Generación automática de reportes y estadísticas de acuerdo a las necesidades del departamento como pacientes atendidos por fechas, por tipo de paciente (estudiante, empleado y docente), por edades, por sexo, por patologías, etc.

5.2.3 RECURSOS

Para el desarrollo del sistema son necesarios los siguientes recursos:

CONCEPTO	COSTO
HARDWARE	
Equipo de Computación	1000
Procesador Intel Core 2 Duo 2.33Ghz	
Memoria 1GB	
Disco Duro 250 GB	
Tarjeta Madre Intel DG33BU	
Impresora	200
SOFTWARE	
Windows Server 2003	475
NetBean 5.5	0
NetBeans Enterprise Pack	0
Sun Java System Application Server 9	0
PostgreSQL 8.2	0
EMS PostgreSQL Manager 3 Lite	0
postgresql-8.1-408.jdbc3.jar	0
jdk1.5.0_09	0
MATERIALES	
Impresión de documentos	50
Útiles de Oficina	40
Impresión de documentos	80
Varios	60

Bibliografía	
Libros	200
Internet	500
Capacitación	500
Imprevistos	300
TOTAL	2905

Al inicio del proyecto el Departamento de Bienestar Universitario en el área de salud no contaba con computadoras ni red de datos.

Por lo cual se solicitó la colocación de puntos de red y la adquisición de equipo como muestra la siguiente figura:

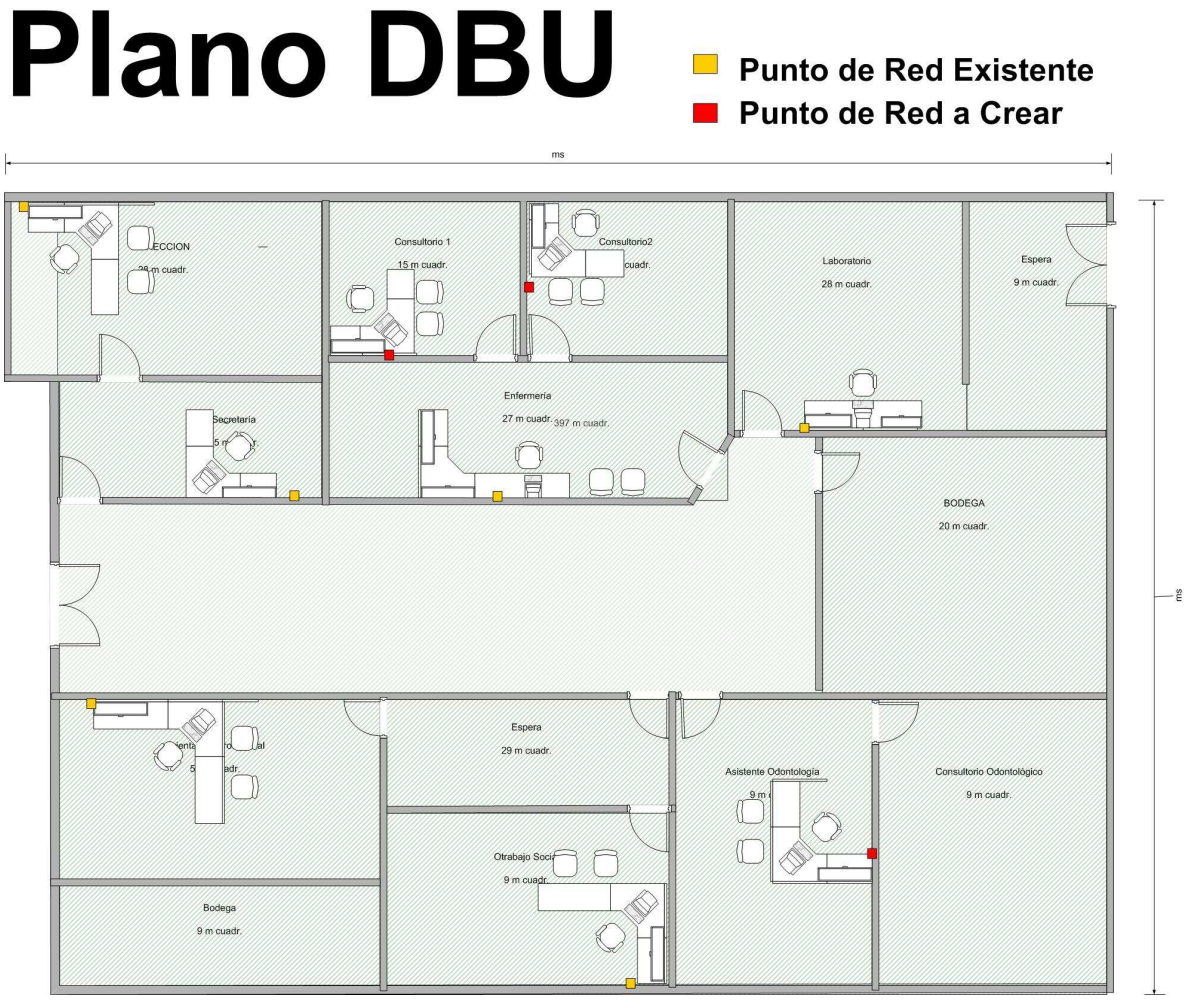


Figura 5.2 Diagrama de ubicación del cableado estructurado del DBU

Recurso	*Cantidad*	*Precio*	*Total*
Estaciones de trabajo	8	240	1920

Equipos Computador	5	1000	5000
Silla Tipo Secretaria	2	100	200
Puntos de Red	3	44	132
Impresoras Láser	7	100	700
Escáner	1	80	80
		Total:	$8032

Instalaciones y adquisiciones que se realizaron progresivamente en el transcurso del desarrollo de los sistemas. Siendo los primeros computadores, nuestras herramientas de desarrollo.

5.2.4 PLAN DE DESARROLLO

En esta etapa inicial se creo el siguiente plan de desarrollo que fue presentado a los directores departamentales como plan tentativo de inicio y culminación del proyecto.

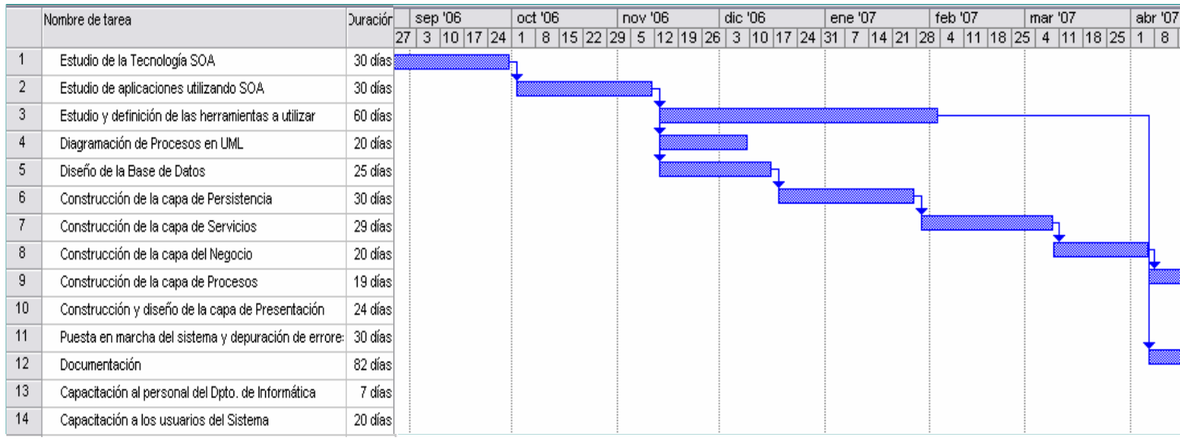


Figura 5.3 Plan de Desarrollo de la aplicación

Inicio: Agosto 2006 **Finalización:** Agosto 2007

Nota: La documentación de la Aplicación se realizará durante todo el desarrollo de la misma. Además, durante todo el tiempo que se este desarrollando el proyecto se realizará una investigación constante de los conocimientos que se necesiten para culminar la aplicación

Con la automatización de procesos complejos, y la protección de la información, el servicio del departamento de bienestar universitario será más preciso y eficiente al momento de atender a sus usuarios, aumentando en ellos la credibilidad y confianza en el uso de la atención médica.

5.3 ANÁLISIS

Durante la recopilación de la información fue posible identificar los procesos involucrados con el Servicio de Atención Médica del DBU que interactúan como muestra la figura:

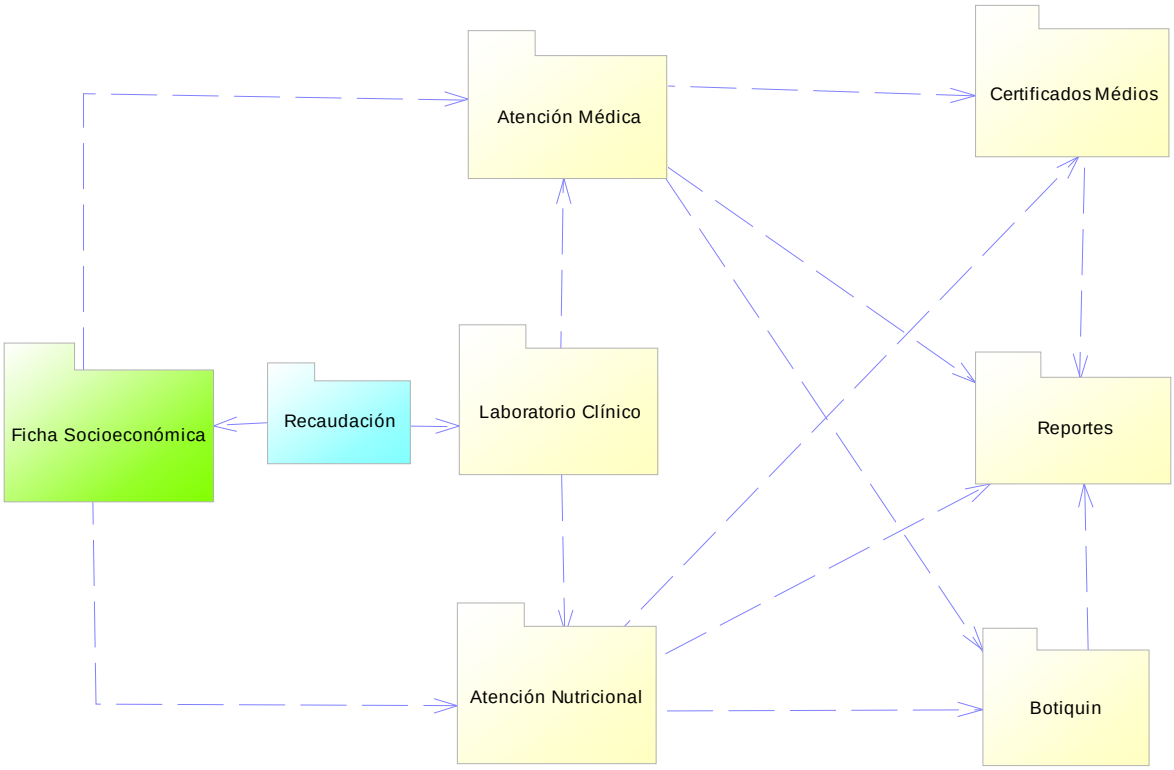


Figura 5.4 Procesos Involucrados con la Gestión Médica

A continuación describo brevemente cada uno de los procesos:
Procesos externos:

- **Recaudación**

Este sistema mantiene la información referente a todo tipo de pagos y venta de especies. De aquí se hace uso de los datos de periodos y sub períodos académicos, estudiantes matriculados, facultades, escuelas, carreras, pagos de derechos de exámenes de laboratorio, etc.

- **Ficha Socioeconómica**

Este sistema mantiene toda la información personal y socioeconómica del estudiante. De aquí se hace uso de los ingresos o actualizaciones que se realice sobre esta información.

Procesos que se realizan internamente y sus actores:

- **Certificados Médicos**

El departamento tiene la responsabilidad de emitir los siguientes certificados:

Certificados Médicos de Control.- certificado de que el estudiante recibió atención médica como requerimiento para matricularse.

Certificado Médico de Reposo.- certificado de reposo por enfermedad en formatos tanto del DBU como del IESS.

Certificado de Atención Médica.- certificado del tiempo que el paciente estuvo presente recibiendo atención médica.

Canje de Certificado Médico.- se certifica que el paciente presentó un certificado de reposo emitido por médicos ajenos al departamento.
Los actores de este proceso son:

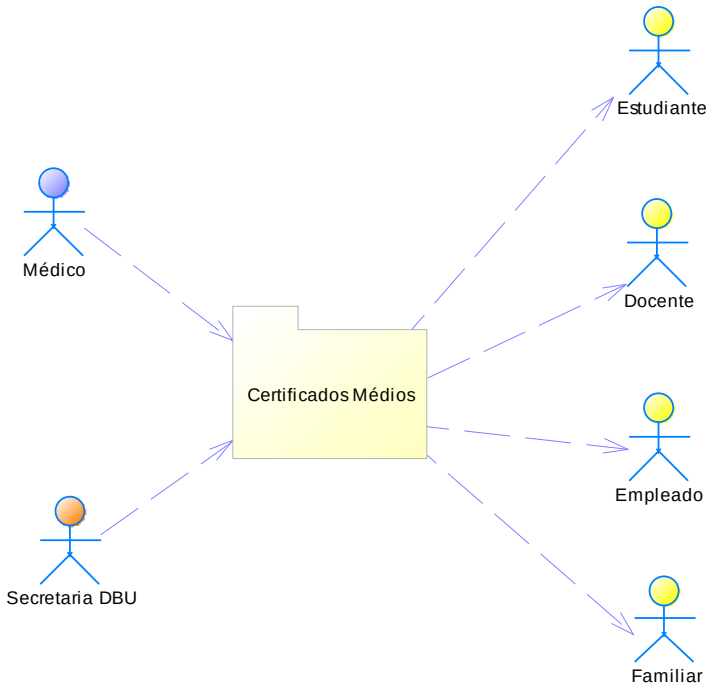


Figura 5.5 Actores del Proceso de Gestión de Certificados Médicos

- **Laboratorio Clínico**

El laboratorio clínico realiza gran variedad de exámenes de laboratorio, los mismos que son ingresados en un pequeño sistema que mantiene una base de datos y luego son impresos para ser entregados al paciente. Los cobros de estos exámenes de laboratorio los realiza el sistema de recaudación. Los actores de éste proceso los muestra la siguiente figura:

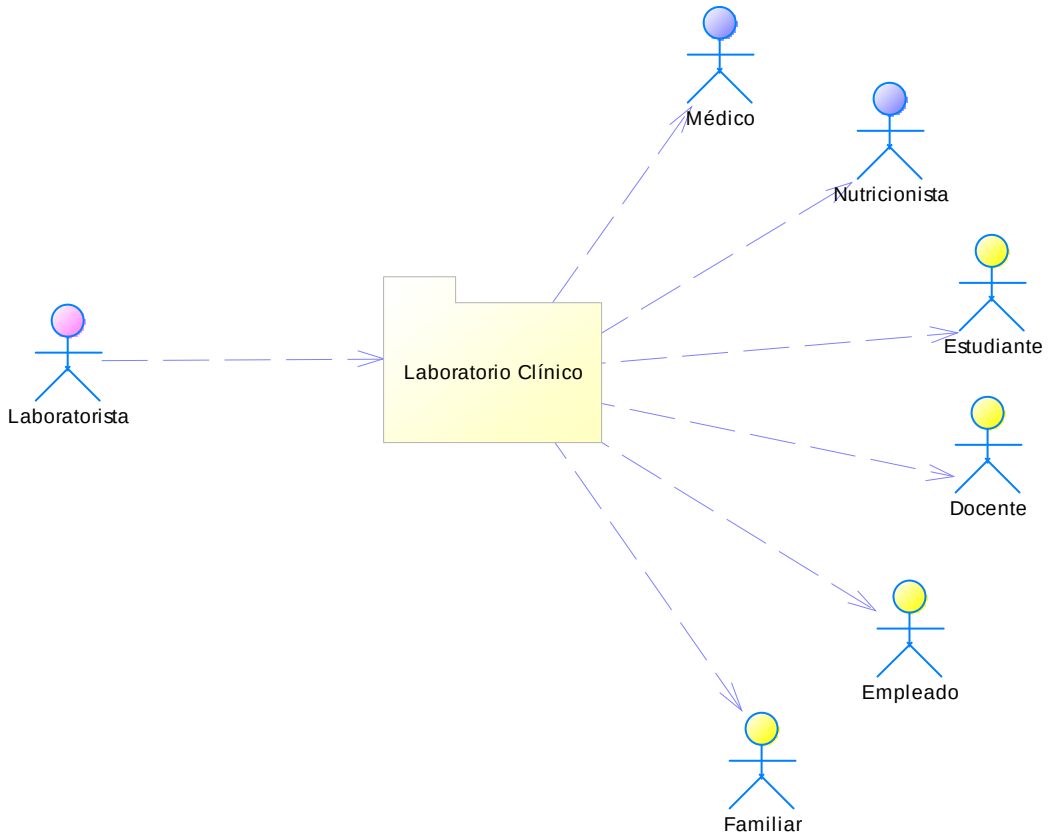


Figura 5.6 Actores del Proceso de Gestión de Laboratorio Clínico

- **Atención Nutricional**

Este proceso lo realizan doctoras Nutricionistas que dan seguimiento al estado nutricional de toda la comunidad universitaria, atendiendo dos días a la semana. Sus actores son:

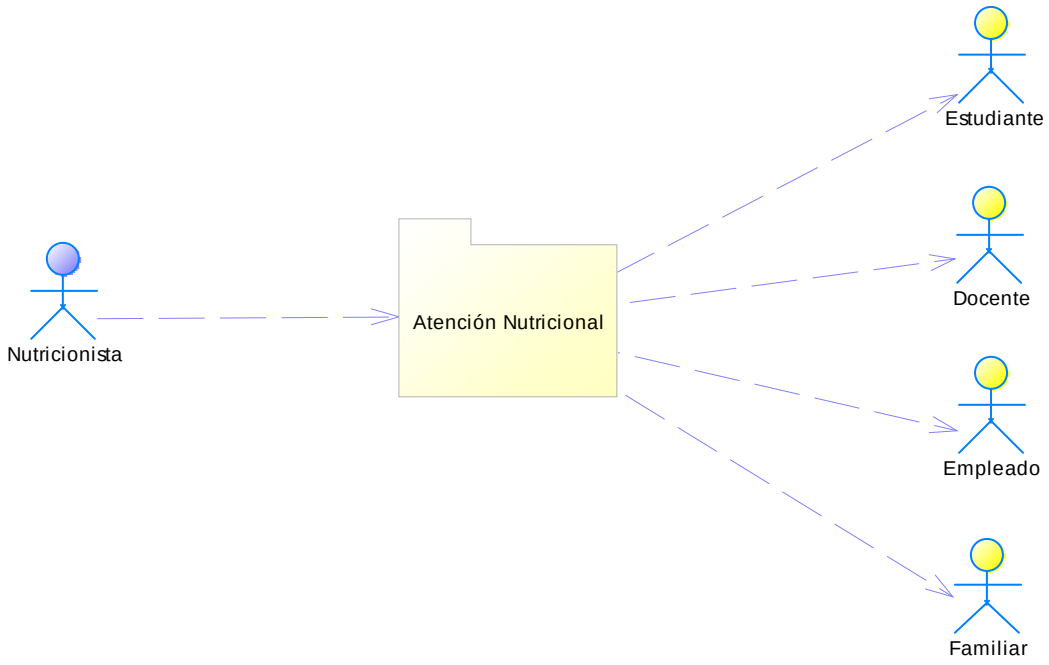


Figura 5.7 Actores del Proceso de Atención Nutricional

- **Botiquín**

El servicio médico cuenta con un conjunto de medicamentos de emergencia, que son entregados gratuitamente a sus pacientes, respaldando la entrega con la firma de un documento de recibido. Además se lleva un inventario manual de estos medicamentos mediante cartones de kardex. Los actores de este proceso son:

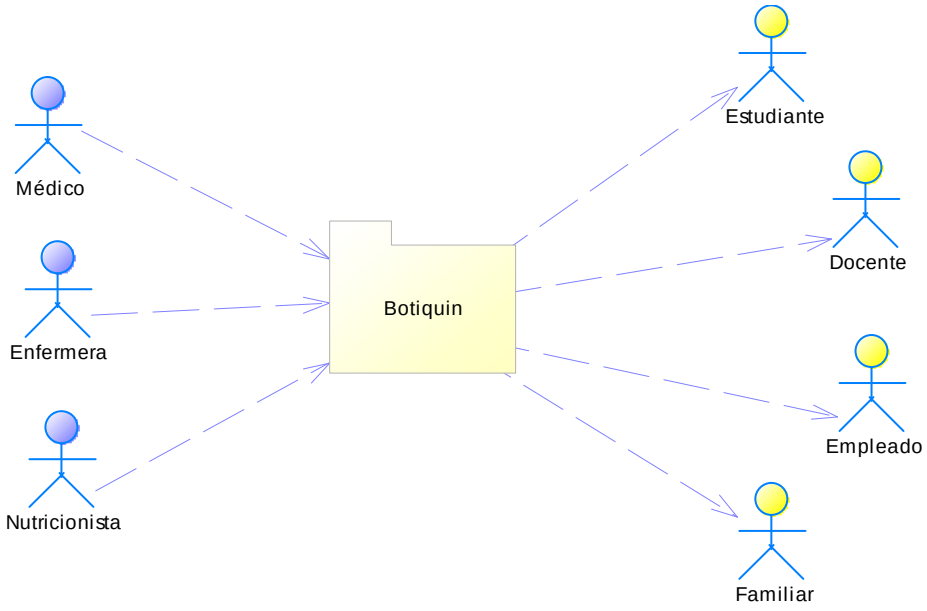


Figura 5.8 Actores del Proceso de Gestión de Botiquín

- **Reportes**

Se realizan Partes Diarios, Concentrados Mensuales y Estadísticas de Diagnósticos mediante conteos manuales de los pacientes atendidos y los diagnósticos emitidos. Los actores de este proceso son:

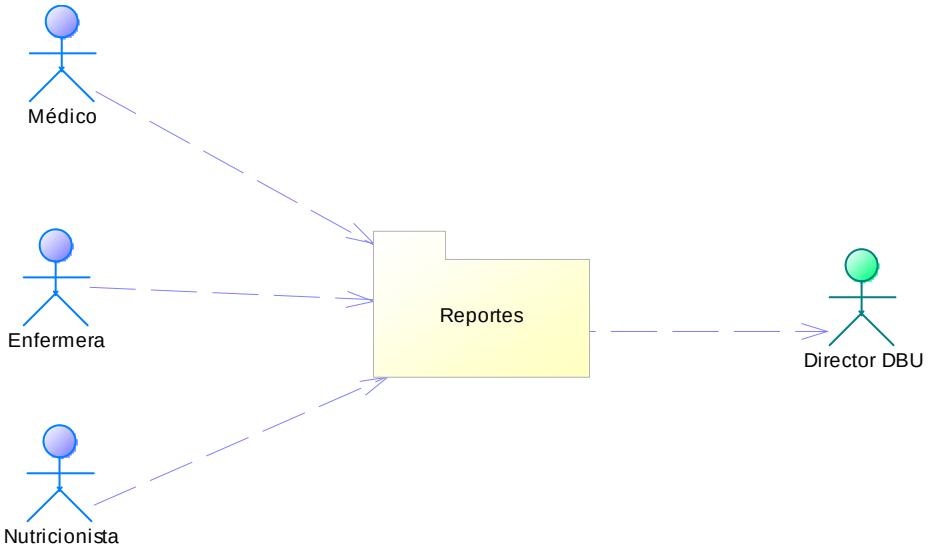


Figura 5.9 Actores del proceso de Generación de Reportes

Especificación de Casos de USO

En ingeniería del software, un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. *[www41]Y*

De forma que al ser parte del análisis nos ayudan a describir qué es lo que el sistema debe hacer. Los Casos de Uso son qué hace el sistema desde el punto de vista del usuario. Y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen *qué* es lo que debe hacer el sistema, pero no *cómo*.

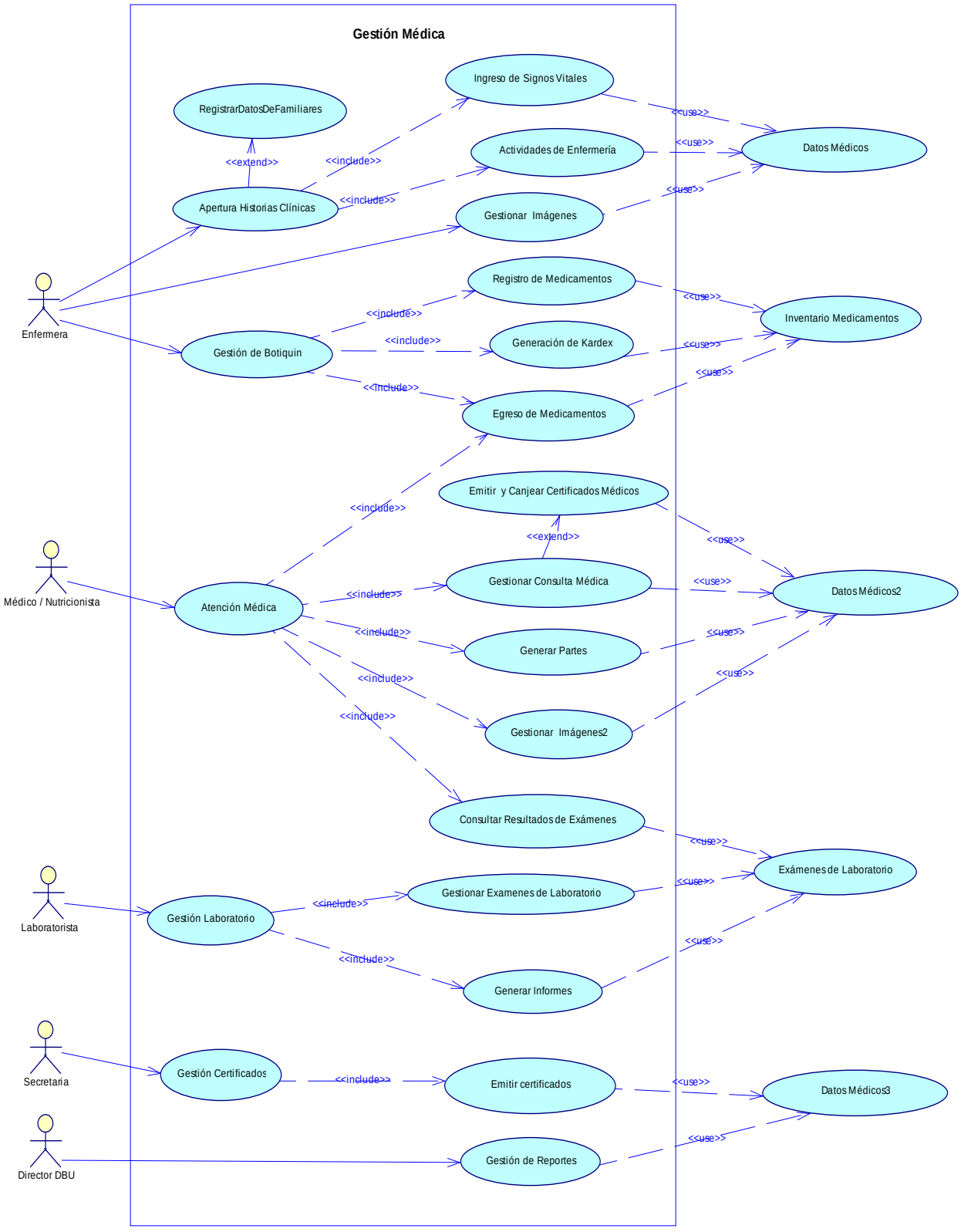


Figura 5.10 Diagrama de Casos de Uso

Realizando el análisis de los procesos a automatizarse fue factible la especificación de la información a manejarse y del flujo de esta en la aplicación.

Haciendo uso de BPM, Gestión de Procesos del Negocio a continuación presento el diagrama de Flujo de Datos de Gestión Médica:

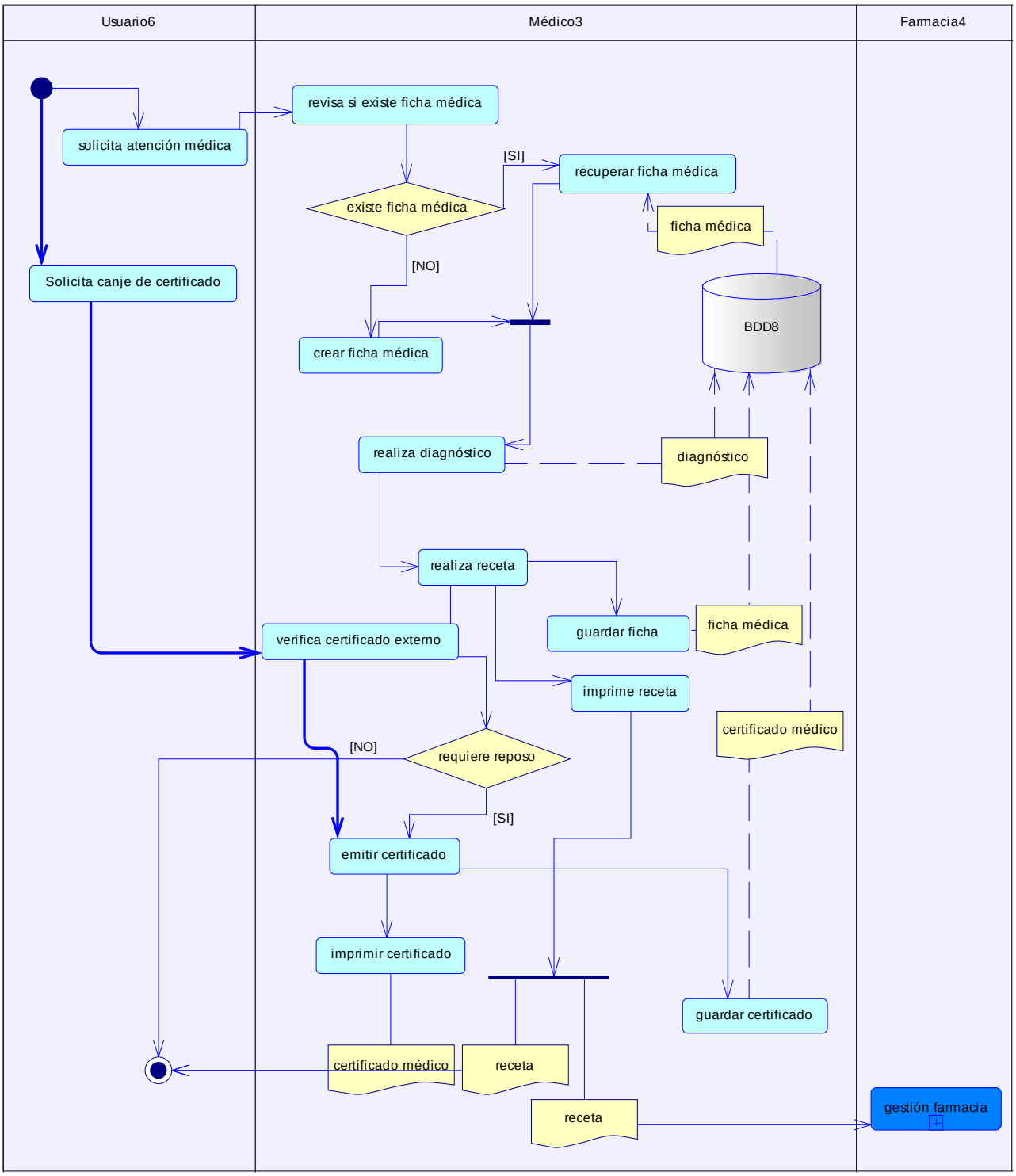


Figura 5.11 DFD Gestión Médica

5.4 DISEÑO

Modelo Físico de Datos.- la base de datos esta integrada con el sistema de gestión Odontológica, y el sistema de Gestión Socioeconómica y comparte unas tablas con el sistema de Recaudación SARE , además de manejar una tablas que forman parte del arrea Académica.

Diccionario de Datos

Como se mencionó anteriormente el sistema de gestión médica interactúa con otros sistemas que manejan información necesaria para su funcionamiento.

Para diferenciar las tablas que pertenecen a los diferentes sistemas o módulos fue necesario definir una nomenclatura, asignando prefijos a cada una de las tablas, esta definición se realizó conjuntamente con el personal del departamento de Informática, con el afán de crear un estándar para el desarrollo de aplicaciones presentes y futuras.

Nomenclatura utilizada en la Base de Datos

Proceso	*Identificativo*	*Nomenclatura*
Servicio Médico	*me*	*bme_tab*
Servcio Odontológico	*od*	*bod_ta*
Laboratorio Cínico	*la*	*blc_tab*
Salud Integral	*sa*	*bsa_tab*
Bienestar Universitario	*bie*	*bie_tab*
Gestión socioeconómica	*se*	*bse_tab*
Recaudación	*rec*	*rec_tab*
Académico	*aca*	*aca_tab*
Uso Institucional	*inst*	*inst_tab*

Descripción detallada de las tablas de la base de datos diseñada para el sistema de Gestión Médica:

5.5 IMPLEMENTACIÓN

El desarrollo de mi aplicación estuvo inicialmente encaminado al uso de software libre, usando Java y con la BDD PostgresSQL 8.2; durante el desarrollo de la misma, la universidad realizó la adquisición de Oracle, pretendiendo estandarizar el desarrollo de aplicaciones sobre esta plataforma; razón por la cual se cambio la base de datos a Oracle 10g.

La aplicación fue desarrollada en lenguaje JAVA, usando JSP, JavaScript, HTML, XML, WSDL, BPEL.

Herramientas Utilizadas:

Productos SOA de SUN:

- Herramienta de desarrollo NetBeans 5.5
- NetBeans Enterprise Pack 5.5.
- Sun Java System Application Server Plataform Edition 9.0
- JBI Runtime with BPEL (Poject Open ESB Starter Kit)

Productos Oracle:

- Base de Datos Oracle 10g.
- Oracle DevSuite Designer 10.

ARQUITECTURA DE LA APLICACIÓN

De acuerdo a las capas de la Arquitectura Orientada a Servicios mi aplicación consta de 5 capas:

Capa de Presentación:

Esta capa esta incluida en una aplicación Web denominada "MedicUTN" formada por páginas JSP y HTML.

Páginas JSPs:

```
aniadir_imagenes.jsp
antecedentes_familiares.jsp
antecedentes_personales.jsp
articulos.jsp
busca_paciente.jsp
cambiar_doctores.jsp
certificados_paciente.jsp
concentrado.jsp
diagnostico.jsp
egreso_articulo.jsp
egreso_detalle.jsp
enfermeria.jsp
```

```
ex_coproparasitario.jsp
ex_emo.jsp
ex_hematologico.jsp
ex_otros.jsp
ex_prueba_embarazo.jsp
ex_quimico.jsp
ex_secrecion_vaginal.jsp
ex_serologico.jsp
examen_fisico.jsp
habitos.jsp
index.jsp
ingreso_articulo.jsp
kardex.jsp
llenar_diagnosticos.jsp
llenar_discapacidades.jsp
llenar_examen_laboratorio.jsp
motivo_consulta.jsp
muestra_usuario_paciente.jsp
nueva_consulta.jsp
nueva_historia.jsp
orden_examen_lab_consulta.jsp
parte_diario.jsp
parte_diario_diagnostico.jsp
parte_diario_medico.jsp
parte_enfermeria.jsp
tratamiento.jsp
usuarios.jsp
ver_pacientes_enf.jsp
ver_pacientes_med.jsp
```

Páginas HTMLs:

```
banner_enfermeria.htm
bannerMedico.htm
calendar.html
indexEnfermeria.htm
indexMedicUTN.htm
inicio_enfermeria.htm
inicio_medic_utn.htm
men_consulta.htm
menu_ex_lab.html
menu_HC.html
menusito.htm
mostrar_enfermeria.html
mostrar_enfermeria_a.html
mostrar_orden_ex_lab.html
```

Capa de Procesos:

Esta capa esta formada por un proyecto BPEL denominado "BPELSalud" en donde se establece el flujo y la orquestación de los servicios web (partners) que yo he implementado.

Creando un XML Shema inicialmente, continuando por definir los wsdl para mis servicios web.

Resultado los siguientes archivos:

```
autentifica_us.wsdl
autentifica_us.xsd
bpelSalud.bpel
busca_paciente.wsdl
busca_paciente.xsd
busca_paciente.xsd…
```

Capa de Servicios:

Esta capa está constituida por un Enterprise Java Bean llamado "ServicesSalud" que contiene todos los Servicios Web, mismos que fueron desarrollados en modo *Bottom-up* es decir a partir de la clase java se generan los wsdl.

Los servicios Web están organizados de la siguiente forma:

```
Package com.dbu.ws
     autentifica_usuarios.java
     busca_paciente.java
     edad_calculada.java
     id_personal.java
     verifica_historia.java

Package com.dbu.indices
     id_consulta_paciente.java
     indice_subperiodo.java
```

Capa del Negocio:

La capa del negocio esta dentro de la aplicación web llamada "MedicUTN" conformada por servlets y clases java que contienen la lógica del negocio. Organizados de la siguiente forma:

```
package com.dbu.medic.business.botiquin
     guarda_articulo.java
     ingreso_articulo.java
     ver_recibo_botiquin.java

package com.dbu.medic.business.certificado
     guarda_certificado.java
     ver_certificado_paciente.java

package com.dbu.medic.business.common
     fecha.java
     redondear.java
     ruta_modificada.java
```

```
package com.dbu.medic.business.configuraciones
      guarda_diagnosticos.java
      guarda_discapacidades.java
      guarda_nuevo_examen_lab.java

package com.dbu.medic.business.consulta
      gestion_imagen.java
      guarda_consulta.java
      guarda_imagen.java
      guarda_prescripcion.java
      prescripcion.java
      ver_imagenes.java
      ver_prescripcion.java

package com.dbu.medic.business.enfermeria
      guarda_enfermeria.java

package com.dbu.medic.business.historia
      editar_historia.java
      guarda_historia.java
      ver_historia.java

package com.dbu.medic.business.ordenes
      guarda_orden_examen.java
      orden_examen_imagen.java
      orden_examen_laboratorio.java
      ver_orden_examen_imagen.java
      ver_ordenes_examen_paciente.java

package com.dbu.medic.business.reportes
      act_enfermeria.java
      cambio_medico.java
      de_cada_paciente.java
```

Varios de los servlets son consumidores de ServiciosWeb de esta forma:

```
//EE-->Incovacion al SW que devuelve el id_personal del usuario q esta
usando el sistema

@WebServiceRef(wsdlLocation =
"http://172.20.2.173:8080/id_personalService/id_personal?WSDL")
    private com.dbu.medic.consumersWS.id_personal.IdPersonalService
sw_id_personal;

        int id_personal=0;
        try{
            com.dbu.medic.consumersWS.id_personal.IdPersonal
port_id_personal = sw_id_personal.getIdPersonalPort();
            id_personal = port_id_personal.idPersonal(ci_usuario);
          } catch (Exception ex) {
                System.out.println("SW_nuevo_id_consulta"+ex.getMessage());
           }
```

Capa de Persistencia:

Esta capa está incluida en la aplicación web llamada "MedicUTN", conformada por dos clases: de conexión y de acceso a datos, realizando la conexión a la base de datos mediante JDBC.

```
package com.dbu.medic.persistence
      AccesoDatos.java
      coneccion.java
```

La integración y el consumo de servicios web se realizo mediante JBI, es un proyecto de Composición de Aplicaciones que comunica los proyectos anteriormente mencionados en cada capa de la aplicación.

FUNCIONALIDADES DEL SISTEMA

El Sistema de Gestión de Salud "MedicUTN" como he decido llamarlo está diseñado para el uso de dos usuarios potenciales:

- Enfermería - Enfermera
- Consulta Externa - Médico

Enfermería. - Para este usuario el sistema presenta las siguientes opciones:

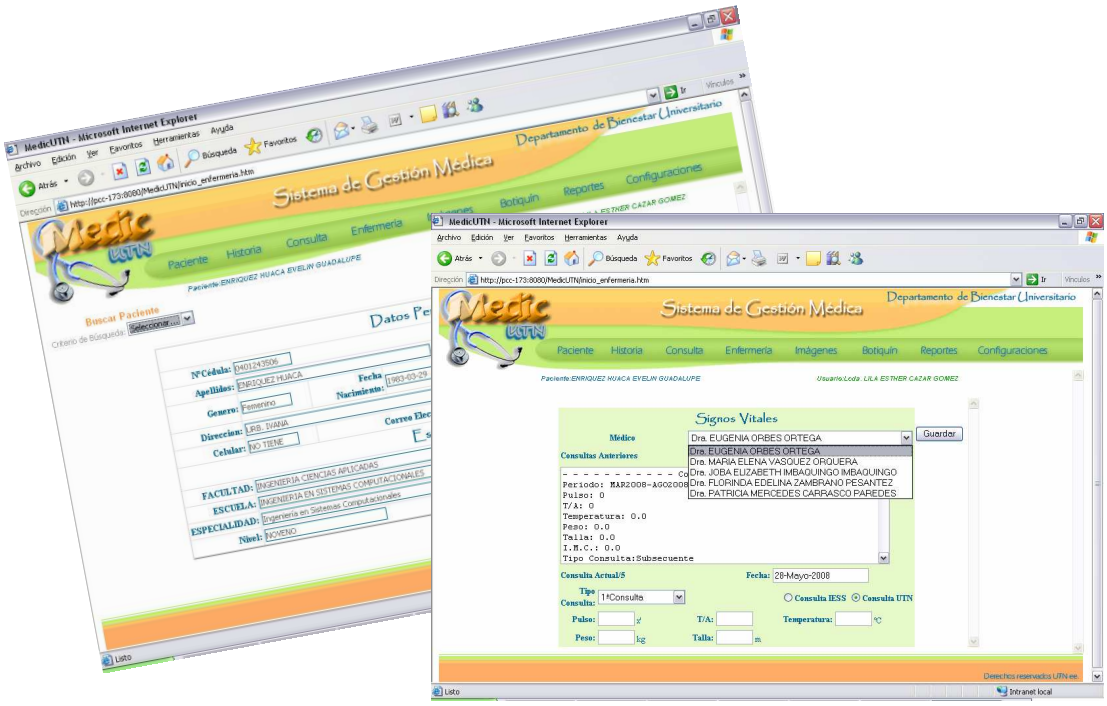


Figura 5.12 MedicUTN -Enfermera

- *Autenticación*.- mediante usuario y contraseña única por usuario, usando sesiones.

- *Búsqueda de Pacientes*.- Mediante dos criterios de búsqueda: Cédula de identidad y Nombre. La búsqueda es realizada por un servicio web, el mismo que también es consumido por el sistema de Gestión Socioeconómica.

- *Creación y Modificación de Historias*.- La Historia Clínica está formada por: Nº de Historia asignado automáticamente, Datos personales, Tipo de paciente, Tipo de sangre, Nº de Historia del IESS, Antecedentes patológicos personales y familiares, Hábitos, Alergias, Examen Visual, Anamnesis alimentaria y Datos de un familiar cercano para comunicarse en caso de emergencia.

- *Signos Vtales.*- Permite el ingreso de: Temperatura, Tensión arterial, Peso, Talla y Pulso, datos que la enfermera registra antes de que el paciente sea atendido, colocando a éste automáticamente en cola de espera para el médico que haya requerido.

- *Actividades de Enfermería.*- Permite registrar las actividades realizadas por la enfermera como: toma de muestras, colocación de inyección, curación, etc.

- *Gestión de Imágenes.*- Permite adjuntar imágenes del paciente a su historia, como radiografías, mamografías, ecografías, etc. a demás de resultados de exámenes de laboratorios particulares.

- *Botiquín.*- Es un inventario de la medicación de emergencia que maneja el área de salud, permitiendo el registro de ingresos y egresos de los medicamentos, generando kardex y alertas de faltantes.

- *Configuraciones.*- se llama así al ingreso de datos referentes a Diagnósticos, Exámenes de Laboratorio y Discapacidades.

Consulta Externa. - Para este usuario el sistema presenta las siguientes opciones:

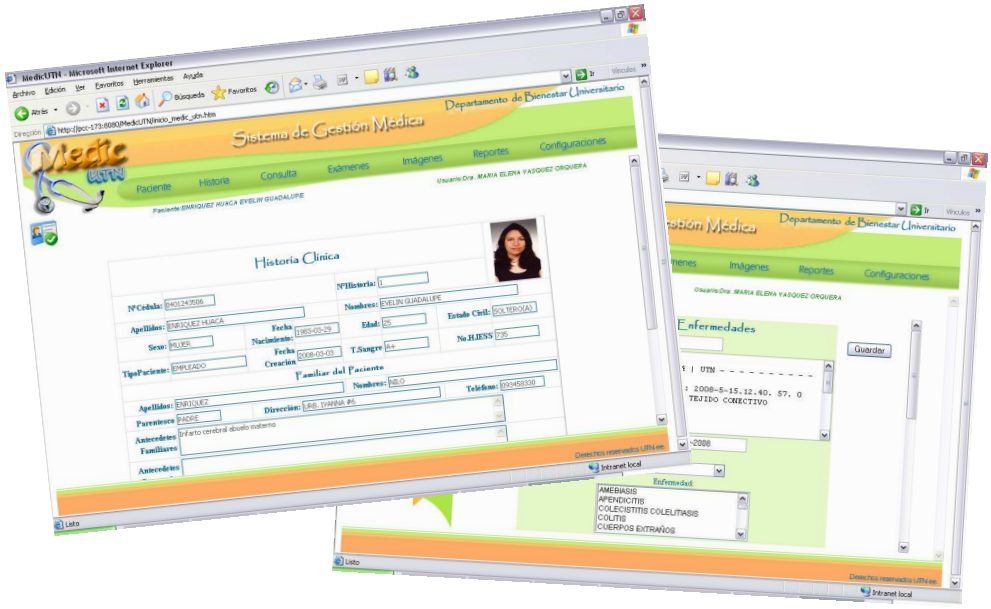


Figura 5.13 MedicUTN –Consulta Externa

- *Autenticación*.- mediante usuario y contraseña única por usuario, usando sesiones.

- *Búsqueda de Pacientes*.- Mediante dos criterios de búsqueda: Cédula de identidad y Nombre. La búsqueda es realizada por un servicio web, el mismo que también es consumido por el sistema de Gestión Socioeconómica.

- *Creación y Modificación de Historias*.- La Historia Clínica está formada por: Nº de Historia asignado automáticamente, Datos personales, Tipo de paciente, Tipo de sangre, Nº de Historia del IESS, Antecedentes patológicos personales y familiares, Hábitos, Alergias, Examen Visual, Anamnesis alimentaria y Datos de un familiar cercano para comunicarse en caso de emergencia.

- *Consulta.-* Le permite la visualización de todas las consultas anteriores, y puede registrar datos de: Motivo consulta, Examen físico, Diagnóstico (agrupado por enfermedad), Discapacidad, Tratamiento, Prescripción, Ordenes de exámenes de Laboratorio, Órdenes de exámenes por Imagen, Canjes de Certificados, Emisión de

certificados de reposo y de Atención médica; los seis últimos pueden ser impresos y entregados al paciente.

- *Exámenes.- Resultados, e*sta opción permite consultar los resultados de los exámenes de laboratorio que se realizan en el laboratorio clínico del departamento, sin que sea necesario su impresión. *Pedidos* el un listado de todo los documentos entregados al paciente para en caso necesario sea reimpresos.

- *Gestión de Imágenes.-* Permite adjuntar imágenes del paciente a su historia, como radiografías, mamografías, ecografías, etc. a demás de resultados de exámenes de laboratorios particulares.

- *Reportes.-* Permite generar reportes como: Parte diario, Parte diario con diagnósticos, Concentrado mensual o en el rango de tiempo deseado, Botiquín, que permite conocer los medicamentos existentes para que sean entregados a los pacientes con la respectiva impresión del recibo.

- *Configuraciones.-* se llama así al ingreso de datos referentes a Diagnósticos, Exámenes de Laboratorio y Discapacidades.

5.6 PRUEBAS

Una vez concluido el desarrollo se sometió a la aplicación a un periodo de pruebas que se realizaron conjuntamente con los otros sistemas que se estaban desarrollando para el departamento.

Tiempo en el cual se encontraron pocas falencias y nuevas necesidades, por ejemplo la opción de cambio de médico, que consiste en pasar un paciente del la cola de un medico a la de otro, ya sea por que se ausentó o por motivos de tiempo.

Además el sistema fue presentado en su totalidad al director y funcionarios del Departamento de Informática, quienes contribuyeron con sus observaciones y recomendaciones.

De esta forma se pulieron varias opciones, procurando ofrecer una mejor funcionalidad y buen desenvolvimiento de la aplicación.

5.7 CAPACITACIÓN A USUARIOS

Para la capacitación a los usuarios, acordamos con los desarrolladores de los Sistemas: Gestión Odontológica y Gestión Socioeconómica, realizar un cronograma conjunto; para esto en una reunión con la Directora del Departamento de Bienestar Universitario se acordó en que la capacitación sería simultánea a los usuarios de los diferentes sistemas, estableciendo horarios en la mañana y en la tarde durante 1 semana.

Para el sistema de Gestión de Salud se capacitó a los siguientes usuarios:

- Médicas (3)
- Nutricionistas (2)
- Enfermera (1)

Debido a la limitada experiencia en el manejo de computadores, la capacitación se prolongó varios días más.

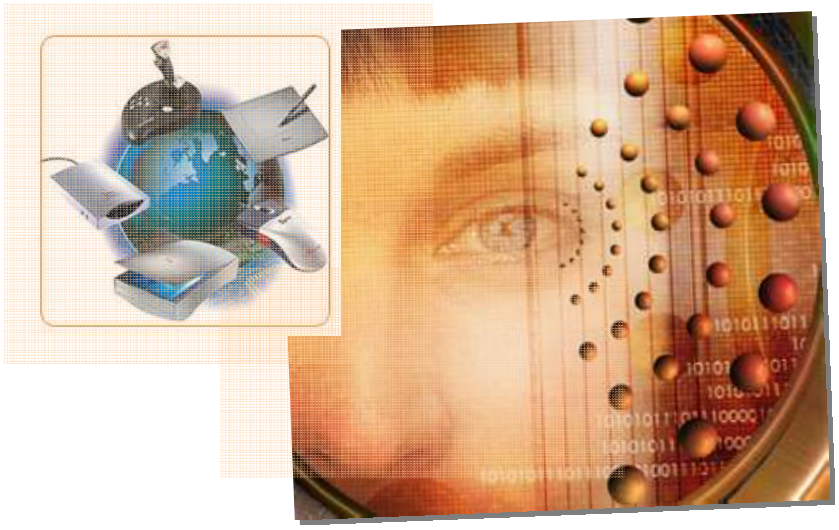
En esta etapa se contó con la mejor predisposición por parte del personal del departamento, su colaboración, su interés y su confianza hicieron que esta tarea salga adelante.

Con su alto sentido de compromiso y conscientes de las ventajas del uso del sistema fue puesto en producción el 4 de marzo del 2008.

Desde entonces la aplicación se encuentra funcionando con normalidad.

CAPITULO 6

CONCLUSIONES Y RECOMENDACIONES



"Conoceréis la verdad y la verdad os hará libres"

LUC 8:32

6.1 Verificación de la Hipótesis

Hipótesis

La aplicación de la Tecnología SOA, en el desarrollo de soluciones para la Universidad Técnica del Norte permitirá integrar eficientemente los sistemas ya existentes y proyectar una integración total con los que surjan en base a nuevas necesidades, definiendo normalización y estandarización.

Verificación

Con esta tesis se demuestra que la hipótesis planteada se cumple totalmente, ya que fue posible que el nuevo Sistema de Gestión Médica se integre con los sistemas existentes: de Laboratorio Clínico y de Recaudación, con el nuevo sistema de Gestión Socioeconómica, y el nuevo Módulo de Certificados Médicos que emite la secretaría del DBU.

6.2 Conclusiones

- La Arquitectura Orientada a Servicios representa un cambio histórico en el modelo de diseño, de programación y de despliegue.

- Si los sistemas de una organización responden al modelo de SOA se obtendrá múltiples beneficios: mejor disposición para responder al cambio tecnológico, menor dependencia de proveedores, una mayor interoperabilidad entre diferentes sistemas y una reutilización de las funcionalidades a un nivel superior, en lugar de la tradicional reutilización de código u objetos.

- En SOA un Sistema equivale a un conjunto de Servicios operando conjuntamente con algún fin específico de ahí que esta tecnología es nombrada dentro de los sistemas distribuidos.

- SOA es el paradigma actual en cuanto a arquitectura de software se refiere, como lo demuestra el apoyo de todas las casas de software y la rapidez con la que se está implantando en las empresas.

- La integración de los procesos de TI de un negocio es una inversión que aumenta el valor de sus sistemas, potencializando su uso y aumentando la capacidad, velocidad y valor global de su negocio siendo más competitivo, productivo y rentable.

- La implementación del Sistema de Gestión Médica "MedicUTN" ha constituido una gran ayuda y apoyo en el desarrollo de los procesos médicos del Departamento de Bienestar Universitario.

- El desarrollo de ésta y otras tesis diseñadas para la Universidad Técnica del Norte fue posible contando con el interés por mejorar, y la lucha contra la tan equivocada "resistencia al cambio" por parte de personas clave, lo que permitirá obtener múltiples beneficios en cuanto al manejo de la información, a nuestra institución.

- El diseño de una interfaz de usuario, amigable y llamativa es parte importante en el buen y pronto acoplamiento entre el usuario y la aplicación.

- La automatización de procesos manuales, genera rapidez y precisión

6.3 Recomendaciones

- Dentro de una empresa no es recomendable sumergirse en implementaciones de nuevas tecnologías, sólo por que estén de moda.

- La existencia de múltiples aplicaciones y sistemas dentro de la empresa que no se comuniquen e interactúen entre sí, puede generar cuellos de botella que repercuten directamente en la velocidad de respuesta global del negocio, impacto en costos al tener procesos críticos manuales o aislados, mayor tiempo de atención al cliente y finalmente una desventaja en la competencia global de su negocio. Por lo cual es importante la integración de la información en una institución de la magnitud de la UTN.

- Es importante una buena definición de los objetivos a conseguir mediante el uso de arquitectura SOA, que servicios se podrían crear, como se accederá a estos, cómo favorecerían al negocio.

- Tener seria dedicación al estudio de la calidad de los servicios implantados, entendida como un aseguramiento mediante las oportunas pruebas tanto de los aspectos funcionales como de los de rendimiento.

- Brindar un decidido apoyo por parte de las Autoridades Universitarias para que se hagan realidad Proyectos que beneficien a la Universidad y sean una oportunidad para el futuro profesional.

- Apoyar continuamente a que tesis como ésta, se realicen en pos del beneficio mutuo entre el estudiante y la universidad.

- Fomentar la investigación de tecnologías como ésta que pueden constituir soluciones óptimas a problemas críticos por los que puede estar atravesando la universidad.

GLOSARIO

API Interfaz de Programación de Aplicaciones, es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Una API representa una interfaz de comunicación entre componentes software.

APPLET Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador Web. Ejemplos comunes de applets son las Java applets y las animaciones Flash. Otro ejemplo es el Windows Media Player utilizado para desplegar archivos de video incrustados en los navegadores como el Internet Explorer.

BPCL Acrónimo de Business Process Coordination Language, es un lenguaje composicional por medio del cual la descripción de un proceso orquestado puede realizarse sin necesidad de especificar detalles de distribución de recursos. Además, cuenta con una infraestructura de coordinación que entre otras tareas será la encargada de ejecutar procesos definidos en BPCL.

Binding es una "ligadura" o referencia a otro símbolo más largo y complicado, y que se usa frecuentemente. Este otro símbolo puede ser un valor de cualquier tipo, númerico, de cadena, etc o el nombre de una variable que contiene un valor o un conjunto de valores.

CORBA Common Object Request Broker Architecture, Arquitectura de agentes de peticiones de objetos común. Es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

CRM Customer Relationship Management, Administración de relaciones al cliente

CICS ***C****ustomer* ***I****nformation* ***C****ontrol* ***S****ystem* (*Sistema de control de información de clientes*), es un gestor transaccional, o monitor de teleproceso, que se ejecuta principalmente en mainframes IBM con los sistemas operativos OS/390, z/OS o VSE. También existen versiones de CICS para otros entornos, como OS/400, OS/2, etc. La versión para entornos Unix recibe el nombre de TX Series.

DBMS Data Base Management System, sistema de administración de base de datos

DCE Distributed Computing Environment, o también OSF DCE, es un sistema de software para computación distribuida. Un conjunto de servicios que puede utilizarse separadamente o en combinación, en ambientes de desarrollo de computación distribuida

DCOM Distributed Component Object Model, modelo objetos de componentes distibuidos. Es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí.

DTD Data Type Definition, definición de tipos de datos.

EAI Enterprise Application Integration (Integración de Aplicaciones de Empresa) se define como el uso de software y principios de arquitectura de sistemas para integrar un conjunto de aplicaciones.

EJBS Los Enterprise JavaBeans son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Sun Microsystems (ahora JEE 5.0). Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor

Encapsulamiento En programación orientada a objetos se denomina al ocultamiento del estado, es decir, de los datos miembro, de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto. Cada objeto está aislado del exterior, es un módulo natural, y la aplicación entera se reduce a un agregado o rompecabezas de objetos. El aislamiento protege a los datos asociados a un objeto contra su modificación por quien no tenga derecho a acceder a ellos, eliminando efectos secundarios e interacciones.

ERP Enterprise Resource Planning, Planeación de recursos empresarial.

Escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

Flujo de control Describe el orden en que serán ejecutadas las actividades de un proceso de negocios. Dicho orden será especificado en términos de construcciones usuales de programación como la composición: secuencial, concurrente y condicional.

GUI Graphic User Interfase (Interfá Rapida de Usuario).

HTML HyperText Markup Language, lenguaje de etiquetas de hipertexto.

IDL Interface definition language. (Lenguaje de definición de interfaz) Es un lenguaje de especificación de interfaces que se utiliza en software de computación distribuida. Ofrece la sintaxis necesaria para definir los procedimientos o métodos que queremos invocar remotamente. Una vez tengamos esta interfaz creada deberemos pasarla por un compilador de interfaces que generará el proxy o stub cliente y el skeleton o stub servidor.

I**nteroperatividad** es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos.

JDBC Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

JSRs Java Specification Requests para proponer y especificar cambios en la plataforma Java.

JSR 175 (Metadatos en Java)

JVM Java Virtual Machine, Máquina virtual Java. Es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode) el cual es generado por el compilador del lenguaje Java.

Proceso Definido mediante una serie de actividades cuyo fin es la ejecución de una tarea determinada.

RPC Remote Procedure Call, llamada a procedimiento remoto. Es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

RMI Remote Method Invocation, Invocación remota de métodos.

SGML Standard Generalized Markup Language, lenguaje de etiquetas estándar generalizado.

SOAP Simple Object Access Protocol, protocolo de acceso a objetos simples.

UDDI Universal Description, Discovery and Integration, descubrimiento e integración de descripciones universales. Documentos WSDL en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

URI Uniform Resource Identifier, identificador uniforme de recursos.

URL *Uniform Resource Locator*, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

W3C World Wide Web Consortium, consorcio del www.

WDSL Web Service Description Language, lenguaje de descripción de servicios web

XHTML Extensible HyperText Markup Language, lenguaje de etiquetas de hipertexto extensible.

XML Extensible Markup Language, Lenguaje de Marcado Extensible.

BIBLIOGRAFÍA

Libros:

- Understanding SOA with Web Services: XML, WSDL, SOAP, and UDDI por Eric Newcomer, Greg Lomow.
  [lib01].

- Enterprise SOA: Service-Oriented Architecture Best Practices. DirkKrafzig, KarlBanke, DirkSlama. PrenticeHall PTR November09, 2004.
  [lib02].

- Arquitectura SOA para la integración entre software libre y software propietario en entornos Mixtos. Ginea de Salas, Alejandro; Jorrín Abellán, Sergio.
  [lib03].

- Ingeniería del Software, un enfoque práctico. Mc. Graw Hill 2da Edición, México 1998 [lib04].

Documentos

- El reto de los Servicios Web para el software libre. J. J. Domínguez Jiménez, A. Estero Botaro, I. Medina Bulo, M. Palomo Duarte y F. Palomo Lozano Depto. de Lenguajes y Sistemas Informáticos. Univ. de Cádiz- España [doc01].

- Solucion De Un Sistema De Gestion De Salud para el Departamento de bienestar Universitario de la UTN. Evelin Enríquez, Tatiana Freire. Agosto 2006 [doc02].

Internet

- Articulo sobre Sistemas Distribuidos.
  http://www.monografias.com/trabajos16/sistemas-distribuidos/sistemas-distribuidos.shtm [www01]

- Arquitectura de Sistemas Distribuidos.
  http://www.dea.icai.upco.es/jarm/Asignaturas/Doc_SistemasDistribuidos/3SDArquitectura.pdf [www02].

- Definición de RPC.
  http://es.wikipedia.org/wiki/RPC [www03].

- Computación distribuida: una historia, una idea y un programa
  http://pergaminoraspado.com/2007/02/22/computacion-distribuida-una-historia-una-idea-y-un-programa/ [www04].

- Computación distribuida
  http://theory.lcs.mit.edu/~rajsbaum/educ96r3.ps [www05].

- Definición de computación distribuida
  http://es.wikipedia.org/wiki/Computaci%C3%B3n_distribuida [www06].

- Integración de Aplicaciones
  http://www.mundoxolido.com/productos_servicios/productos_d_ing.shtml?idboletin=194&idseccion=3668&palcla=1 [www07].

- Gestión de procesos e integración de sistemas: EAI, BPM, SOA y ESBs
  http://www.afi.es/infoanalistas/comun/mostrarFichero.asp?idContenido=725191&idSeccion=156467&usuario=TIFBrewery&password=TIFBrewery
  [www08].

- Levantando el velo de SOA
  http://www.acis.org.co/index.php?id=555 [www09].

- Articulo sobre la evolución de los Servicios Web.
  http://wwww.creamoselfuturo.com/sanidad [www10].

- Servicios Web.
  http://www.microsoft.com/spanish/msdn/articulos/archivo/151102/voices/servi_web.asp [www11].

- Información sobre el Desarrollo de aplicaciones web.
  www.desarrolloweb.com [www12].

- World Wide Web Consortium.
  www.w3.org [www13].

- Portal con información teórica e implementación de Servicios web.
  www.montejava.es. [www14].

- Consosrcio UDDI.(Propone una plataforma estándar e interoperable que permite a las aplicaciones de forma sencilla y dinámica, encontrar y utilizar servicios Web sobre Internet)
  www.uddi.org [www15].

- OASIS dirige el desarrollo, la convergencia y la adopción de estándares para el comercio electrónico.
  www.oasis-open.org [www16].

- Webopedia, es una enciclopedia de términos técnicos en informática.
  www.pcwebopedia.com [www17]

- Donde empiezan los Servicios Web Por Evelio Martínez Martínez y José Martín Olguín Espinoza Publicado en la Revista RED, Mayo, Junio de 2004.

www.red.com.mx [www18].

- SOAP Protocolo Simple de Acceso a Objetos.
  http://www.w3.org/TR/soap12-part1/ [www19].
- WS-Addressing. Direccionamiento de Servicios Web.
  http://www.w3.org/TR/ws-addr-core/ [www20].
- MTOM .Descripción de la Optimización de la Transmisión del Mensaje.
  http://www.w3.org/TR/soap12-mtom/ [www21].
- WSDL Lenguaje de Descripción de Servicios Web.
  http://www.w3.org/TR/wsdl20/ [www22].
- WS-CDL Lenguaje de Descripción de la Coreografía de los Servicios Web.
  http://www.w3.org/TR/ws-cdl-10/ [www23].
- IBM WS.
  http://www-128.ibm.com/developerworks/webservices/library/ws-featuddi/ [www24].
- Apis de Google con Servicios Web.
  http://www.google.com/apis/ [www25].
- Ejemplos de servicios web.
  http://www.amazon.com [www26].
- Servicios Web en hipertexto.
  http://www.hipertexto.info/documentos/serv_web.htm [www27]
- Articulo sobre integración de la TI.
  www.mundoenlinea.cl [www28].
- SOA en las noticias del mundo.
  www.news.com [www29].
- La mejor tecnología de integración es SOA.
  www.develop.com/soa [www30].
- Sitio official de SOA. "Service Oriented Architecture", Arquitectura Orientada a Servicios.
  www.service-architecture.com [www31].
- IBM SOA.
  www-306.ibm.com/software/solutions/soa/ [www32].
- BEA SOA.
  www.bea.com/framework.jsp?CNT=index.htm&FP=/content/solutions/soa/ [www33].
- Microsoft artículos de SOA.
  http://www.microsoft.com/spanish/msdn [www34].

- Articulo de SOA.
  www.networkworld.com [www35].

- El papel de un ESB en una SOA -TIBCO
  www.tibco.com/software/soa/ [www36].

- SOA desde un enfoque JAVA
  www.onjava.com/pub/a/onjava/2005/01/26 /soa- intro.html [www37].

- Oracle SOA
  www.oracle.com/technologies/soa/index.html [www38].

- BPEL: una propuesta para el uso de Web Services Patricia Bazán Buenos Aires Argentina http://www.cacic2007.unne.edu.ar/papers/028.pdf [www39].

- Gestión de Procesos BPM
  http://www.ibermatica.com [www40].

- Casos de Uso
  http://es.wikipedia.org/wiki/Casos_de_uso [www41].

ANEXOS

a) ANTEPROYECTO DE TESIS

1. TEMA

1 "ESTUDIO DE LA TECNOLOGÍA DE INTEGRACIÓN
2 SOA (ARQUITECTURA ORIENTADA A SERVICIOS)"

APLICATIVO

"Sistema de Gestión de Historias Clínicas utilizando SOA (Arquitectura Orientada Servicios) para el Departamento de Bienestar Universitario de la UTN"

2. PROBLEMA

En la actualidad gran cantidad de empresas e instituciones cuentan con sistemas en sus diferentes departamentos, con el objetivo de agilizar el manejo de su información.

Pensando en solucionar necesidades urgentes, adquieren o desarrollan sistemas que proveen una solución específica en determinado momento. De esta forma las empresas van acumulando diversas aplicaciones, que trabajan en diferentes plataformas, con diferente base de datos, desarrolladas en diferentas lenguajes, generando resultados en diferente formato, etc. De esta forma una empresa llega a manejar información aislada, muchas veces redundante e incluso inconsistente mediante aplicaciones completamente heterogéneas.

A la vez los recursos son escasos, las empresas no pueden simplemente abandonar las aplicaciones existentes.

Esta es la situación de la Universidad Técnica del Norte, en la que sus sistemas trabajan independientemente para solucionar determinadas tareas, además no existe un estándar de tecnologías de desarrollo, ocasionando la diversidad de aplicaciones y la imposibilidad de comunicación e integración entre ellas.

JUSTIFICACIÓN

La realidad actual de TI en las empresas, es que la infraestructura es heterogénea a través de sistemas operativos, aplicaciones, software de sistema e infraestructura de aplicaciones instalados o creados en diferentes momentos y bajo diferentes paradigmas.

En una institución como la UTN que cuenta con diversos departamentos, se encuentran automatizados varios procesos, y otros en desarrollo; todos estos diseñados en diferentes tecnologías para proveer soluciones específicas.

La UTN enfrenta el problema del aislamiento de información, y redundancia de datos, tal es el caso del sistema de Recaudación, el sistema de Fichas Psico-sociales y un nuevo sistema de Gestión de Salud, los cuales manejan datos semejantes pero individualmente.

Algunas aplicaciones existentes son usadas para correr procesos de negocios claves, por lo tanto empezar desde cero a construir una nueva infraestructura no es una opción.

De allí mi interés por el estudio de la Tecnología de Integración SOA, la cual permite reutilizar aplicaciones y promete interoperabilidad entre aplicaciones y tecnologías desiguales.

3. OBJETIVOS

Objetivos Generales

- Estudiar la tecnología de Integración SOA (Arquitectura Orientada a Servicios), su funcionalidad y aplicabilidad.

- Aplicar SOA en la implementación del sistema de Gestión de Historias Clínicas para el Departamento de Bienestar Universitario de la UTN.

Objetivos Específicos

- Estudiar sobre diseño y arquitectura de Sistemas de Información Distribuidos
- Analizar sobre la Comunicación en Sistemas de Información.
- Realizar un estudio de introducción al Middleware. RPC.TP Monitors. Broker de Objetos. Middleware Orientado a Mensajes.
- Investigar las Tecnologías de Integración de Aplicaciones.
- Estudiar el funcionamiento de protocolos y los conceptos de Servicios Web.
- Documentar sobre SOA.
- Estudiar el funcionamiento y características de la Arquitectura Orientada a Servicios.
- Ventajas y desventajas del desarrollo de aplicaciones siguiendo la Arquitectura Orientada a Servicios.
- Estudio de la coordinación de Servicios Web sus protocolos, estándares, infraestructura para la coordinación (*WS-Coordination y WS-Transaction*).
- Analizar la implementación de SOA como solución a la integración de aplicaciones existentes en Planta Central de la UTN.
- Desarrollar un sistema de Gestión de Historias Clínicas para el DBU utilizando SOA para su integración con dos aplicaciones existentes en la UTN.

4. MARCO TEÓRICO

Desde siempre, desarrollar aplicaciones empresariales no ha sido una tarea fácil considerando que los avances de la tecnología han ido simplificando el manejo de varios aspectos de este problema. Sin embargo, los desarrolladores de este tipo de aplicaciones continúan enfrentándose a desafíos tales como datos complejos; la mayor parte del tiempo los requerimientos no son explícitos, usuarios simultáneos múltiples, los requerimientos del negocio cambian con frecuencia, plataformas heterogéneas, e interdependencias complejas entre aplicaciones distribuidas. [www001]

La Arquitectura Orientada a Servicios (SOA) es una arquitectura de Tecnologías de Información (TI) creciente, que intenta reconciliar la visión técnica y de negocios, basándose en estándares abiertos y promoviendo la interoperabilidad entre diversas organizaciones y plataformas, de manera eficiente y flexible a los cambios. Promoviendo la construcción de aplicaciones basadas en los servicios disponibles en una red como la Web dando soporte a los requerimientos de software del usuario.

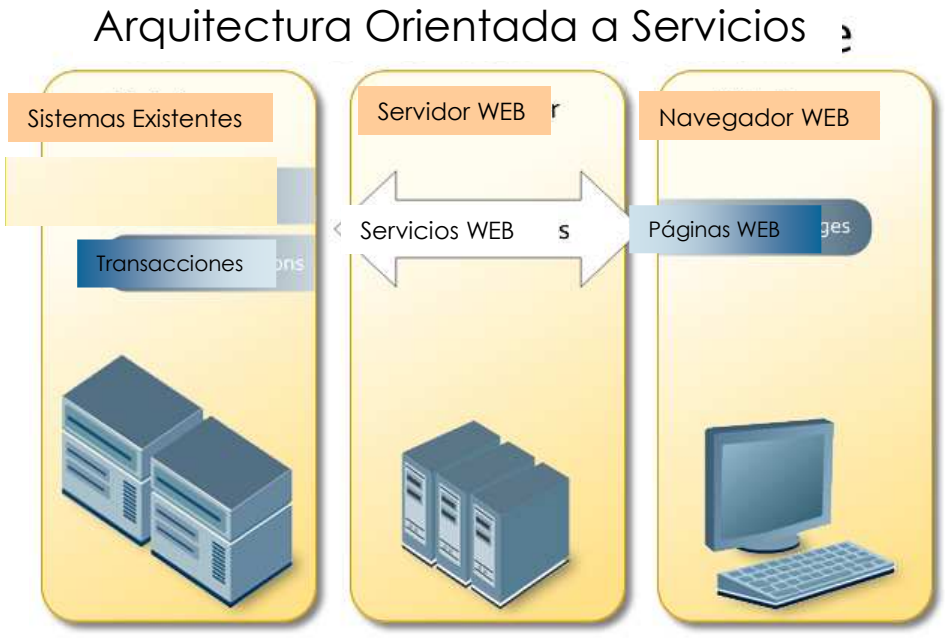


Figura 1 SOA (Arquitectura Orientada a Servicios)

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado.

La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, no obstante es posible implementar una SOA utilizando cualquier tecnología basada en servicios.

Al contrario de las arquitecturas orientadas a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación (p.ej., WSDL).

La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Java o .NET). [www02]

2.1 Definiciones SOA

Término	Definición / Comentario
servicio	Una función sin estado, auto-contenida, que acepta una(s) llamada(s) y devuelve una(s) respuesta(s) mediante una interfaz bien definida. Los servicios pueden también ejecutar unidades discretas de trabajo como serían editar y procesar una transacción. Los servicios no dependen del estado de otras funciones o procesos. La tecnología concreta utilizada para prestar el servicio no es parte de esta definición.
orquestación	Secuenciar los servicios y proveer la lógica adicional para procesar datos. No incluye la presentación de los datos. Coordinación.
sin estado	No mantiene ni depende de condición pre-existente alguna. En una SOA los servicios no son dependientes de la condición de ningún otro servicio. Reciben en la llamada toda la información que necesitan para dar una respuesta. Debido a que los servicios son "sin estado", pueden ser secuenciados (orquestados) en numerosas secuencias (algunas veces llamadas tuberías o pipelines) para realizar la lógica del negocio.

proveedor	La función que brinda un servicio en respuesta a una llamada o petición desde un consumidor.
consumidor	La función que consume el resultado del servicio provisto por un proveedor.

Cuando se habla de una arquitectura orientada a servicios se esta hablando de un juego de servicios residentes en Internet o en una intranet, usando servicios web.

Hay una lista de estándares que se hallan ligados a los servicios web. Incluyen los siguientes:

- XML
- HTTP
- SOAP
- WSDL
- UDDI

Alcance De La Investigación

La investigación está dirigida a estudiar la arquitectura y funcionamiento de SOA, las aplicaciones distribuidas, las tecnologías de integración, su interoperabilidad y acoplamiento.

Además analizar el software específico y hardware complementario, necesario para el diseño e implementación de estas aplicaciones integradas.

Incluyendo la determinación de prestaciones, funcionalidad y riesgos de las aplicaciones SOA. Al igual que las diferentes tecnologías que facilitan el diseño e implementación de SOA.

Alcance Del Aplicativo

Al realizar la investigación sobre las Tecnologías de Integración, Aplicaciones distribuidas y las prestaciones de SOA, es de mi interés encontrar el mecanismo que cumpla con los requerimientos que posee el Departamento de Bienestar Universitario y la planta central de la UTN. Y que pueda satisfacer las necesidades de los usuarios que asisten a éste día a día.

Con el objetivo de solucionar el problema de los procesos manuales en el Área de Salud, el aislamiento de información, y la redundancia de datos; desarrollaré un Sistema de Gestión de Historias Clínicas, que a su vez se integre con el Sistema de Recaudación existente.

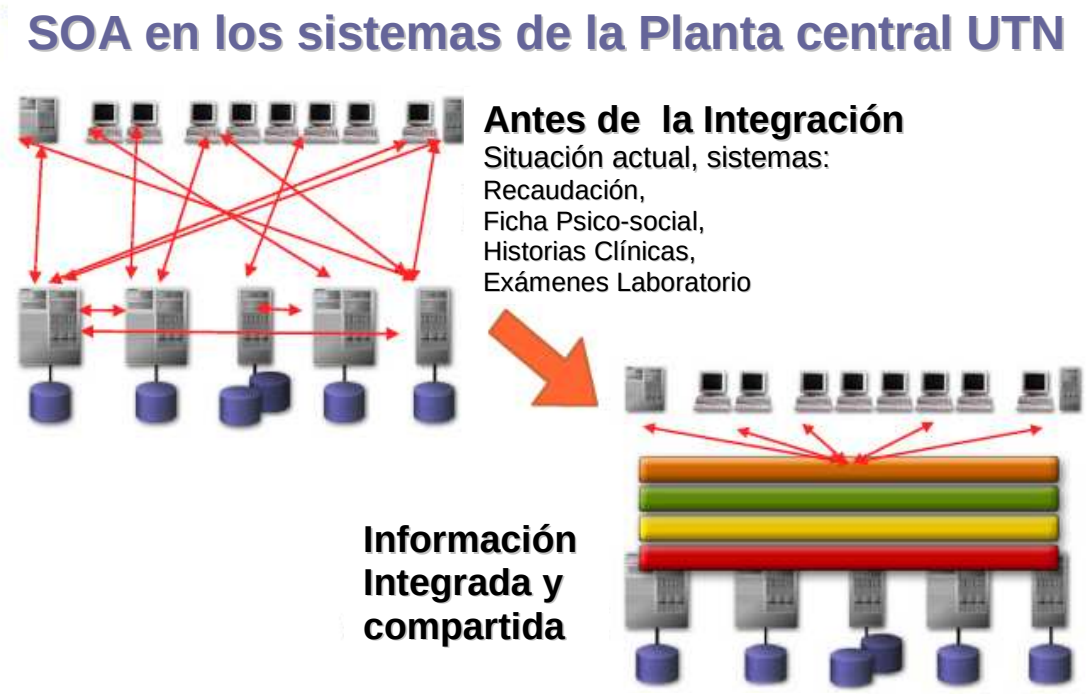


Figura 2. SOA integrando los sistemas de la UTN

Es también parte del alcance de mi aplicativo, capacitar a las personas, que administrarán el Sistema de Gestión de Historias Clínicas y a los usuarios del mismo.

5. HIPÓTESIS

La aplicación de la Tecnología SOA, en el desarrollo de soluciones para la Universidad Técnica del Norte permitirá integrar eficientemente los sistemas ya existentes y proyectar una integración total con los que surjan en base a nuevas necesidades, definiendo normalización y estandarización.

6. METODOLOGÍA

El proceso metodológico a seguir para el logro de los objetivos y de la demostración de la hipótesis se sintetiza en:

Recolectar información a cerca de Tecnologías de Integración, Aplicaciones Distribuidas y SOA, mediante consultas, investigaciones de todos los medios bibliográficos existentes.

Clasificación y organización de la información obtenida utilizando la técnica de Lectura Científica. Estudiar la tecnología y arquitectura SOA su funcionamiento diseño e implementación en aplicaciones existentes. Estudio de los lenguajes y herramientas que permiten su desarrollo.

Analizar los requerimientos y necesidades del aplicativo a desarrollarse; mediante la observación de campo, entrevistas a profesionales, etc.

Diseñar e implementar el sistema de Gestión de Historias Clínicas para el Departamento de Bienestar Universitario de la UTN.

7. TABLA DE CONTENIDOS

6. Construcción del Sistema de Gestión de Historias Clínicas

6.1 Análisis de requerimientos

6.2 Análisis y diseño del Sistema

6.3 Desarrollo de implementación

6.4 Realización de pruebas.

6.5 Capacitación a usuarios

7. Conclusiones y Recomendaciones

7.1 Verificación de la Hipótesis

7.2 Conclusiones

7.3 Recomendaciones

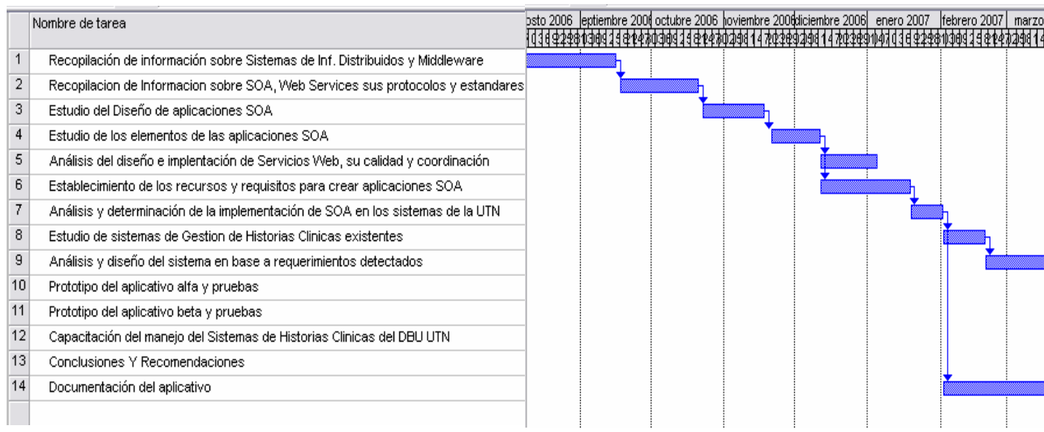
7.4 Posibles Temas de Tesis

8. MARCO ADMINISTRATIVO

RECURSOS

CONCEPTO	COSTO ACTUAL	COSTO REAL
HARDWARE		
Equipo de Computación	**700**	**0**
Intel Pentium IV		
Procesador 2.4 GHz		
Memoria 512 MB		
Disco Duro 80 GB		
Mainboard Intel 845		
Impresora Canon BJC1000		
SOFTWARE		
Windows Server 2003	**100**	**0**
Java Enterprise System NetBeans Enterprise Pack 5.5 PostgreSQL	**0**	**0**
MATERIALES		
Reproducción de documentos	**90**	**90**
Útiles de oficina	**50**	**50**
Impresión de documentos	**80**	**80**
Varios	**200**	**200**
BIBLIOGRAFIA		
Libros, Revistas	**600**	**600**
Internet	**240**	**240**
SUBTOTAL	**2060**	**1260**
IMPREVISTOS	**850**	**282**
TOTAL	**2910**	**1542**

9. CRONOGRAMA DE ACTIVIDADES



Inicio: Agosto 2006

Finalización: Agosto 2007

Nota:

La documentación de la Aplicación se realizará durante todo el desarrollo de la misma. Además, durante todo el tiempo que se este desarrollando el proyecto se realizará una investigación constante de los conocimientos que se necesiten para culminar la aplicación

b) SOLUCIÓN Y PROPUESTA DE DESARROLLO SSISTEMA INTEGRAL DE SALUD

PROYECTO

DESARROLLO DE UN SISTEMA DE GESTION DE SALUD PARA EL DEPARTAMENTO DE BIENESTAR UNIVERSITARIO DE LA UTN

ANTECEDENTES

El Departamento de Bienestar Universitario de la Universidad Técnica del Norte es una dependencia que contribuye al desarrollo socio económico, académico y de salud del personal estudiantil, docente y administrativo de la institución.

Esta dependencia busca prestar servicios a los miembros de la comunidad universitaria con personal idóneo, para ser un punto de apoyo en el logro del mejoramiento de la calidad de vida y la búsqueda de la excelencia académica.

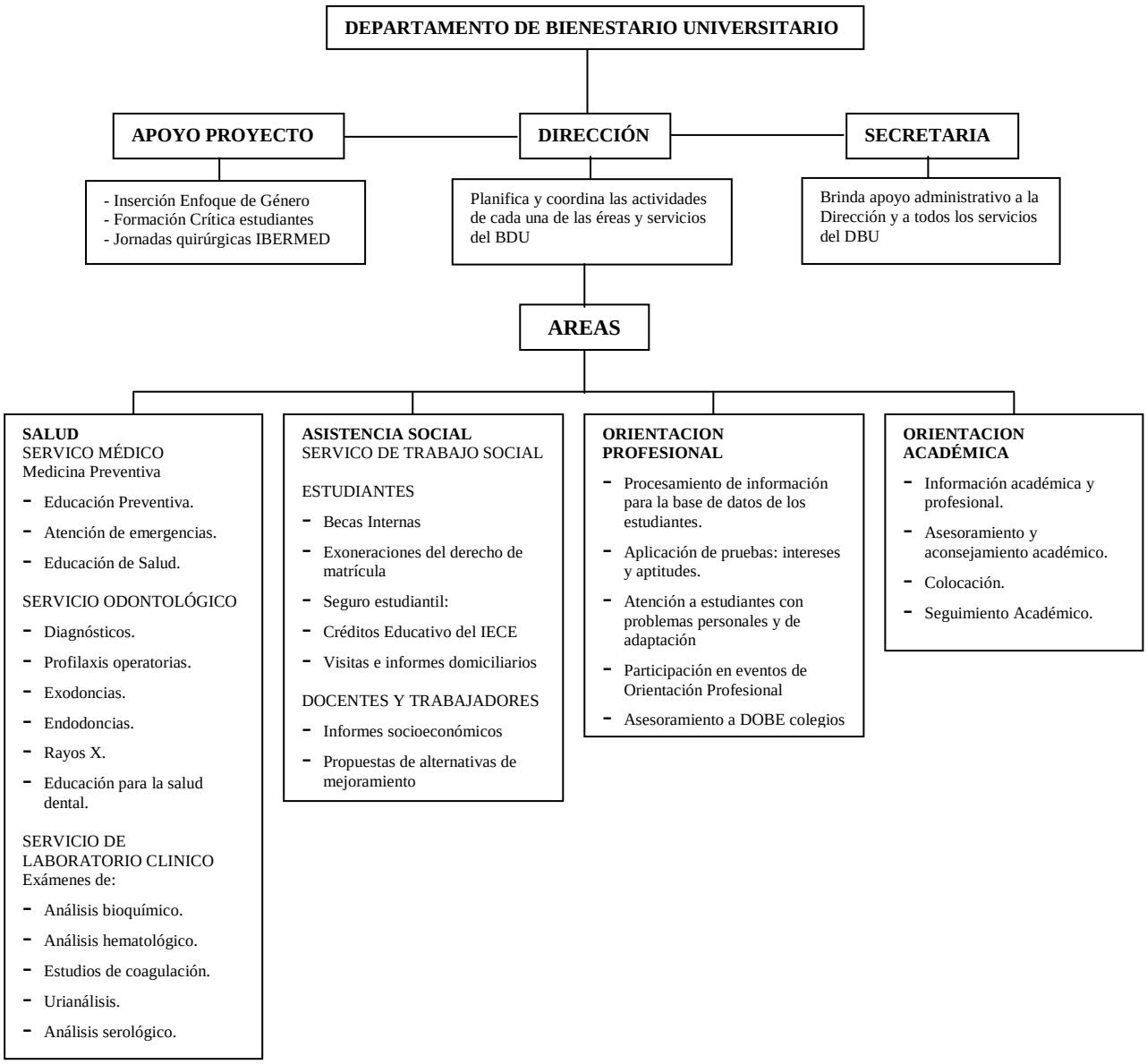
Coherente con el objetivo de la universidad de formar integralmente nuevos profesionales que deberán ser los protagonistas del desarrollo de nuestro país, cumple un rol fundamental. Esta acción de carácter primordial se traduce básicamente en la promoción, organización y realización de actividades de salud y formación humana.

Diariamente el Departamento de Bienestar presta sus diversos servicios, debiendo mantener información exclusiva de cada una de las personas que acuden, como son: estudiantes (7106), docentes (326), empleados (324) y familiares (112) *[Sistema Recaudación SARE]*; cuyo volumen de información representa una gran responsabilidad

Tomando en consideración lo anterior, es necesario contar con herramientas de software que garanticen el correcto tratamiento de la información y ayuden a proporcionar un buen servicio.

DESCRIPCIÓN DEL PROBLEMA

El departamento de bienestar universitario está conformado por cuatro áreas: Salud, Asistencia Social, Orientación Profesional, Orientación Académica.



ÁREA DE SALUD

El área de Salud a su vez está estructurado de la siguiente forma: Servicio Médico, Servicio Odontológico y Servicio de Laboratorio Clínico.

Esta área presta atención personalizada a estudiantes, docentes, personal administrativo y sus familiares, tanto del colegio como de la universidad.

El área de Salud maneja toda la información de sus pacientes en formularios, hojas de trabajo, registros diarios, registros mensuales y anuales manualmente; además lleva un registro de ingresos y

egresos de medicamentos y materiales mediante cartones de kardex. La inexistencia de un sistema automatizado en esta área, ha ocasionado una serie de inconvenientes tales como:

- **Información aislada y redundante.-** Al no existir un control de historias clínicas automatizado impide el acceso a la información que mantiene el servicio de Laboratorio Clínico el cual cuenta con un sistema de ingreso de resultados. Además la información existente en el sistema de Ficha Psicosocial es redundante a la que contiene la ficha historia clínica.

- **Falta de agilidad en la atención al paciente.-** La historia clínica de cada paciente está constituida por varios formularios llenados a mano y archivados en un mueble, de donde son extraídos al momento de iniciar la consulta. Este tedioso proceso requiere mayor atención y tiempo, ocasionando molestias tanto al personal del departamento como a los pacientes.

- **Dificultad en la generación de informes y estadísticas.-** Al no existir un proceso automático para registrar la atención al paciente en el departamento se lleva diariamente un registro manual que luego es contabilizado para informes mensuales, lo que requiere gran cantidad de tiempo. Además para la generación de cuadros estadísticos, es necesario que el personal del departamento obtenga los datos examinando cada uno de los formularios de historias clínicas de los pacientes existentes; siendo ésta una tarea compleja y poco precisa.

- **Pérdida de información.**- Los datos de historias clínicas y los resultados de exámenes es información confidencial y de vital importancia, sin embargo es manipulada por varias personas, y archivada en lugares poco propicios, lo cual facilita su pérdida.

- **Desconocimiento de medicamentos y materiales existentes.**- El servicio médico y odontológico manejan cierta cantidad de medicamentos de emergencia, los cuales son contabilizados manualmente mediante kárdex sin embargo no se lleva un control adecuado del mismo dado que los datos no son actualizados constantemente. Además el servicio de laboratorio y el odontológico manejan materiales de uso exclusivo los mismos que no son inventariados.

ÁREA DE ASISTENCIA SOCIAL

Esta área desempeña varias actividades referentes a trabajo social, mantiene información de becas, exoneraciones, créditos educativos del IECE, seguro estudiantil y Ficha Psicosocial, siendo esta última la única que cuenta con un sistema implementado desde el año 2000. Esta área tiene los siguientes problemas:

- **Dificultad en la generación de reportes y estadísticas.** El sistema existente proporciona varios reportes, muchos de los cuales presentan información errónea o no cumplen con los requerimientos actuales del departamento, ocasionando que el personal los realice manualmente.
  Al momento de generar estadísticas se reflejan los datos erróneos, obteniendo resultados inconsistentes.

- **Pérdida de información.-** Además se maneja información de becas, exoneraciones y créditos educativos mediante registros manuales en cuadernos y archivos de Word. En el caso de becas el personal lleva un seguimiento detallado del rendimiento de cada estudiante becado. Estos documentos son fácil objeto de pérdida.

- **Desconocimiento de Beneficiarios**.-Se incluye en esta área la información referente al seguro estudiantil, la cual consta de un listado de los estudiantes que han cancelado el derecho de seguro en la matrícula. Dicho listado lo proporciona el sistema de recaudación en hojas de Excel. Además el sistema de ficha Psicosocial no permite registrar a los estudiantes y las razones por las que utilizan los servicios de becas, exoneraciones o créditos ocasionando desconocimiento.

ÁREAS DE ORIENTACIÓN PROFESIONAL Y DE ORIENTACIÓN ACADÉMICA

Estas áreas se encargan de actividades como: aplicación de pruebas, atención a estudiantes con problemas, información académica y profesional, entre otras; las mismas que son desempeñadas personalmente.

También se encargan de realizar seguimiento académico, el cual resulta complejo al no existir un sistema académico integrado que proporcione información de todos los estudiantes.

PROPUESTA DE DESARROLLO EN BASE A REQUERIMIENTOS

JUSTIFICACION

La integración de la salud, con las tecnologías de la información y las telecomunicaciones permitirá la arquitectura y la construcción de una nueva forma de organización y funcionamiento de los servicios de salud, donde la innovación sea factor clave para un cambio con sentido y rumbo. Un cambio que permita transformar la administración tradicional en salud, mediante procesos más eficientes orientados a la atención del paciente y al logro de un sistema de salud mejor comunicado e integrando con los procesos de los sistemas de gestión informática de la universidad.

ÁREA DE SALUD

SERVICIO MÉDICO

Para el servicio médico se desarrollará los siguientes módulos:

HISTORIAS CLINICAS

- Identificación del Paciente a través de un único identificador de pacientes.
- Manejo de historias clínicas a través de formularios predefinidos con datos personales, una amplia lista de antecedentes médicos familiares y personales, signos vitales, examen físico clasificado por sistemas, motivo de consulta, diagnostico, tratamiento y observación.
- La historia clínica consultará resultados de exámenes generados por el módulo de laboratorio.
- A la Historia Clínica de cada paciente se podrá adosar una cantidad ilimitada de imágenes del mismo, resultados de exámenes realizados en otros laboratorios, ecografías, etc.
- Emisión de órdenes de Laboratorio.
- Luego de la consulta podrá digitar la prescripción e imprimirla.

INVENTARIO DE MEDICAMENTOS

- Sistema de inventario de medicamentos de emergencia, que estará a disposición del médico, proporcionando un reporte preciso de los medicamentos existentes en el momento de la consulta.

- Se descargarán automáticamente los medicamentos prescritos al paciente, sustentando el egreso mediante la impresión y firma de un comprobante.

ENFERMERÍA

- Módulo en el que se registrará actividades que desempeña la enfermera como curaciones, colocación de inyecciones, etc. adjuntándose a la historia de cada paciente.

REPORTES Y ESTADÍSTICAS

- Generación automática de reportes y estadísticas de acuerdo a las necesidades del departamento como pacientes atendidos por fechas, por tipo de paciente (estudiante, empleado y docente), por edades, por sexo, por patologías, etc.

SERVICIO ODONTOLÓGICO

Para el servicio odontológico se desarrollará los siguientes módulos:

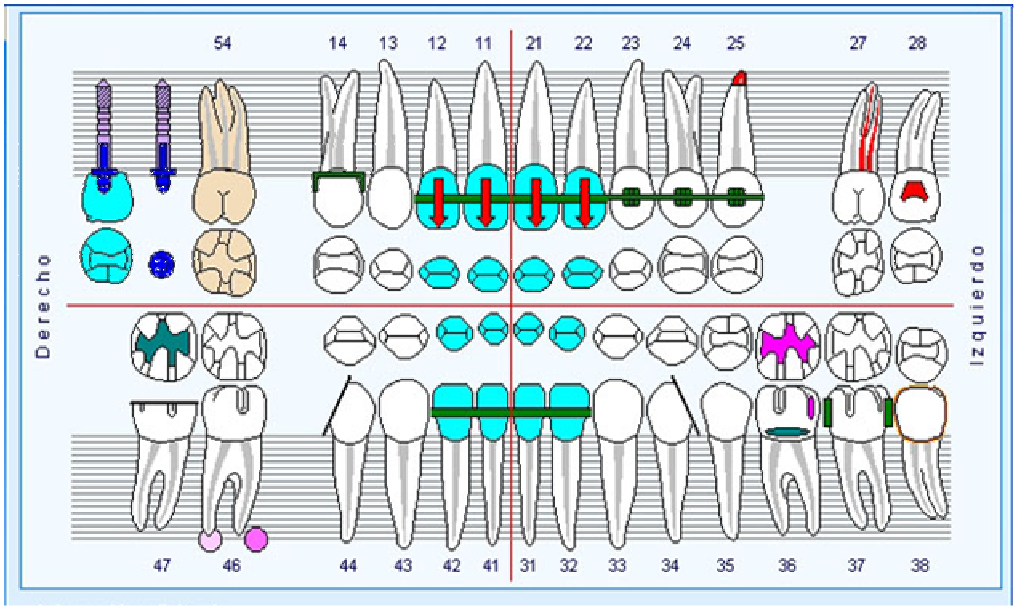
HISTORIAS CLINICAS ODONTOLÓGICAS

- Obtendrá datos del módulo de Servicio Médico, como datos personales, antecedentes patológicos, signos vitales.

- Registro del examen bucal.

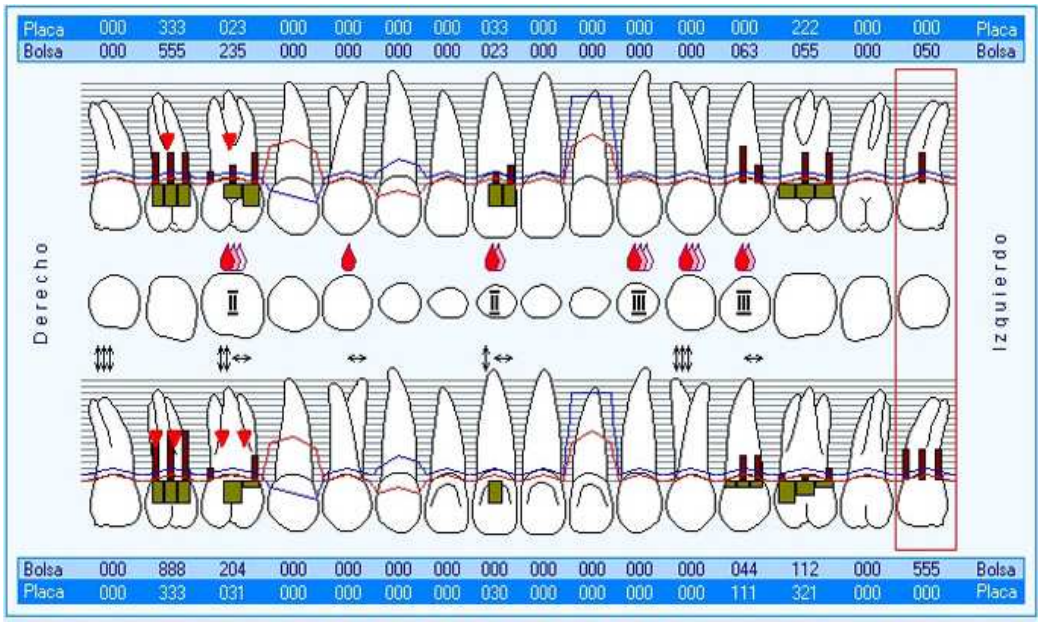
- Creación del odontograma inicial al momento de la primera consulta y modificaciones para las subsecuentes.

Odontograma



- A los pacientes se le podrá registrar datos periodontales, de endodoncia y oclusión.

Periodontograma



- A la Historia Clínica de cada paciente se podrá adosar una cantidad ilimitada de imágenes del mismo

- Luego de la consulta contará con una ayuda visual de Vademécum para emitir la prescripción e imprimirla.

INVENTARIO DE MEDICAMENTOS

- Sistema de inventario de medicamentos de emergencia, que estará a disposición del médico, proporcionando un reporte preciso de los medicamentos existentes en el momento de la consulta.

- Se descargarán automáticamente los medicamentos prescritos al paciente, sustentando el egreso mediante la impresión y firma de un comprobante.

INVENTARIO DE MATERIALES

- Sistema de inventario de materiales.

REPORTES Y ESTADÍSTICAS

- Generación automática de reportes y estadísticas de acuerdo a las necesidades del departamento como pacientes atendidos por fechas, por tipo de paciente (estudiante, empleado y docente), por edades, por sexo, por patologías, etc.

AGENDA DE CITAS

- Inicialmente se definen los horarios de atención de cada doctor y la duración de las citas. De esta manera, la agenda de citas es totalmente realista**.**

- La agenda de citas se puede visualizar de varias maneras en la pantalla: 1 día, 2 días, la semana de lunes a viernes o la semana de lunes a domingo, pudiendo indicar para cada paciente si asistió o no.

LABORATORIO CLÍNICO

EXAMENES DE LABORATORIO

- Rediseño del actual sistema para la integración de este servicio con las historias clínicas.
- Registro de todos los trabajos enviados al laboratorio, registrando, entre otros datos, las fechas de envío, recepción.

INVENTARIO DE MATERIALES

- Sistema de inventario de materiales.

REPORTES Y ESTADÍSTICAS

- Generación automática de reportes y estadísticas de acuerdo a las necesidades del servicio.

SEGURIDAD

- Se puede definir una cantidad ilimitada de usuarios que pueden acceder al sistema, cada uno con su correspondiente clave secreta, que puede ser modificada en cualquier momento.
- Los usuarios autorizados pueden configurar a qué módulos o funciones pueden acceder los demás.

ÁREA DE ASISTENCIA SOCIAL

FICHA PSICOSOCIAL

MODIFICACIONES AL SISTEMA

- Modificación del sistema actual para cumplir con nuevos requerimientos del departamento en cuanto a reportes y rangos de selección.
- Establecimiento de nuevas opciones para el manejo de los datos y generación de reportes.

- Integración al sistema la información de becas, exoneraciones, créditos y seguro estudiantil lo que permitirá la obtención de reportes y el eficaz manejo de la misma.

- Modificación a la estructura de la base de datos.

REQUERIMIENTOS DE HARDWARE Y SOFTWARE

- 5 PC's
- 1 impresora blanco y negro por médico
- 1 scanner
- Power Builder 9.0 /10.0
- Postgress 8.1

LIMITACIONES

- Tiempo reducido.
- Desconocimiento de términos médicos.
- Usuarios exigentes.
- Dependencia de terceros.
- Falta de recursos económicos.