UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

Escuela Académica Profesional de Ingeniería Informática

INTEGRACIÓN DE SISTEMAS HEREDADOS UTILIZANDO WEB SERVICES

TESIS Para Obtener el Título de

INGENIERO INFORMÁTICO

BACH. HERNÁN MANUEL RIVAS LEÓN

Lima - Perú

2006

- A mi Padre Juan Rivas, por su sacrificio y confianza que tuvo en mí.

- A mi Madre Olga León, por apoyarme y guiarme, me dio la fuerza necesaria para alcanzar esta meta.

- A mis Hermanos: Juan, Javier y Maritza, por su confianza y apoyo constante.

- A mi hermano Cesar, y a mis Abuelos que descansan en paz y me iluminan desde el cielo.

- A mis Tíos, Primos y Sobrinos: Rivas & León.

AGRADECIMIENTOS

Expreso mi más sincero agradecimiento al **M.Sc. Erwin Mac Dowall Reynoso**, asesor de esta tesis, quien me brindó su apoyo u orientación en todo el transcurso del presente estudio, igualmente por brindarme sus conocimientos y su experiencia para conseguir lo mejor de esta tesis.

Un sincero agradecimiento a los **docentes de la URP**, por haber compartido sus conocimientos y el haber adquirido nuevas experiencias en el desarrollo de proyectos de software, incluidos en los talleres dentro de la currícula de mi especialidad.

Por último deseo expresar mi más sincero agradecimiento a mis más grandes amistades que siempre estuvieron pendientes de mí y me apoyaron en grandes momentos de mi vida.

RESUMEN

Hoy en día, la mayoría de las organizaciones se encuentran adaptando sus procesos de negocios a los nuevos escenarios económicos y tecnológicos, a efectos de poderse mantener competitivas en estos. En ese sentido, la modernización e integración de sus Sistemas de Software son tareas indispensables para lograr ese objetivo. En algunas organizaciones, el portafolio de Sistemas de Software esta compuesto por los llamados Sistemas Heredados, los cuales en la mayoría de casos, por dar soporte a funciones de misión crítica de la organización, es necesario modernizar e integrar.

La modernización e integración de Sistemas Heredados no es una tarea fácil sino imposible de lograr, dada las características de estos sistemas, como por ejemplo tener una estructura monolítica, estar desarrollados con tecnología obsoleta, poseer poca o nula documentación, etc. Tradicionalmente, los esfuerzos de integración de estos sistemas apuntan al reemplazo total (rediseño) o gradual (migración) de estos sistemas. En muchos casos, estos esfuerzos han fracasado dado que los nuevos sistemas no tenían las mismas funcionalidades de los Sistemas Heredados, debido principalmente a que la documentación era muy pobre. Una estrategia mas simple, es la de dotar de un nuevo visual al sistema, o sea desarrollar una especie de "envoltorio" grafico (wrapping) que permita acceder a sus funcionalidades. Esta estrategia es muy ampliamente utilizada.

Con el advenimiento del concepto de Reuso de Software y dado que los Sistemas Heredados poseen funcionalidades que han demostrado su confiabilidad a lo largo de los años, seria interesante "exponer" estas para que puedan ser utilizadas en la construcción de otros sistemas. En ese sentido, la aparición de nuevas tecnologías computacionales, como por ejemplo Web Services, permite que estas funcionales puedan ser reutilizadas en el desarrollo de nuevos sistemas, con lo cual se consigue "revitalizar" a los Sistemas Heredados.

Es objetivo de esta tesis, es mostrar la utilización de la tecnología de Web Services como una nueva estrategia de integración de Sistemas Heredados.

Tabla de contenidos

CAPÍTULO 1
INTRODUCCIÓN

1.1 Objetivo

Tradicionalmente, las estrategias de integración de Sistemas Heredados están centradas en: (1) reemplazar, de forma total o gradual, el Sistema Heredado o (2) dotar de un nuevo visual (entorno gráfico) al sistema. En ambos casos, no se permite reutilizar las funcionalidades existentes, las cuales han probado ser confiables a lo largo de la vida de este tipo de sistemas. Hoy en día, los nuevos sistemas son desarrollados reutilizando funcionalidades existentes en otros sistemas. En ese contexto, y dado que los Sistemas Heredados poseen funcionalidades que serian convenientes reaprovechar, se hace necesario explorar una nueva estrategia de integración que permita esto.

Uno de los principios de integración de sistemas señala que estas no deben ser invasivas, es decir, no se debe modificar el código de estos [52]. En ese sentido, la nueva estrategia de integración de Sistemas Heredados debe incorporar ese principio. La aparición de nuevas tecnologías con especificaciones estándares, permite el desarrollo de sistemas sin tener que "atar" estos a la tecnología utilizada.

La aparición de la tecnología de Web Services, permite el desarrollo de sistemas por medio de la invocación de servicios (funcionalidades implementadas en otros sistemas) utilizando tecnología basada en la Web. Una característica interesante de la tecnología de Web Services es que permite que los sistemas puedan "exponer" sus servicios a cualquier otro sistema que lo necesite sin que sea necesaria su modificación.

El **objetivo** de la presente tesis, es el de mostrar de forma sistematizada, la utilización de la tecnología de Web Services como una nueva estrategia de integración de Sistemas Heredados, para lo cual se implementara un prototipo.

1.2 Organización de la Tesis

La presente tesis se ha estructurado de la siguiente forma:

En el **Capítulo 2**, se describe la importancia de la integración de sistemas de Software y cuáles son las dificultades inherentes en esa integración. Un aspecto central en este capítulo, es el estudio de las estrategias actuales de integración de Sistemas Heredados.

En el **Capítulo 3**, se detalla las características funcionales más importantes de la tecnología de Web Services, las cuales servirán de fundamento al capítulo 4.

En el **Capítulo 4**, se presenta la arquitectura conceptual, como propuesta para la integración de Sistemas Heredados, la cual servirá para la implementación mostrada en el capítulo 5.

En el **Capítulo 5**, se demuestra la implementación de la estrategia de Web Services, basada en la propuesta del capítulo 4.

En **Capítulo 6**, se presentan las conclusiones, recomendaciones y trabajos futuros de la presente tesis.

En el **Anexo 1**, se muestra la utilización del UML para representar en forma grafica los componentes para la implementación de integración de Sistemas Heredados utilizando Web Services.

En el **Anexo2**, se muestra las interfaces del Sistema Heredado.

En el **Anexo 3**, se muestra las interfaces del Web Services.

En el **Anexo 4**, se muestra las interfaces de la Aplicación de Consumo.

CAPÍTULO 2
INTEGRACIÓN DE SISTEMAS HEREDADOS

En el presente capítulo, se presenta el problema de integración de sistemas de Software. Una de cuyas características es la presencia de los llamados Sistemas Heredados. Se estudia en detalle las características de los Sistemas Heredados, sus problemas de integración y cuáles son sus estrategias más utilizadas para su integración.

2.1 Caracterización del Problema de Integración de Sistemas

En la mayoría de las organizaciones, los sistemas de software presentan las siguientes características: (1) autonomía y (2) heterogeneidad.

Los sistemas son autónomos porque su construcción, mantenimiento y operación se realizan de forma independiente y sin tomar en consideración la integración con otros sistemas existentes en la organización. Esta autonomía trae como consecuencia la aparición de una plataforma de sistemas altamente heterogéneos, pues favorecen a la utilización de diferentes tecnologías de hardware y de software en su desarrollo. Es decir, estos sistemas autónomos han sido construidos utilizando una amplia diversidad de tecnologías computacionales, incluyendo plataformas de hardware, sistemas operativos, tecnologías de bases de datos, formatos de representación de datos, y lenguajes de programación. [54][55]. (Ver figura 2.1).

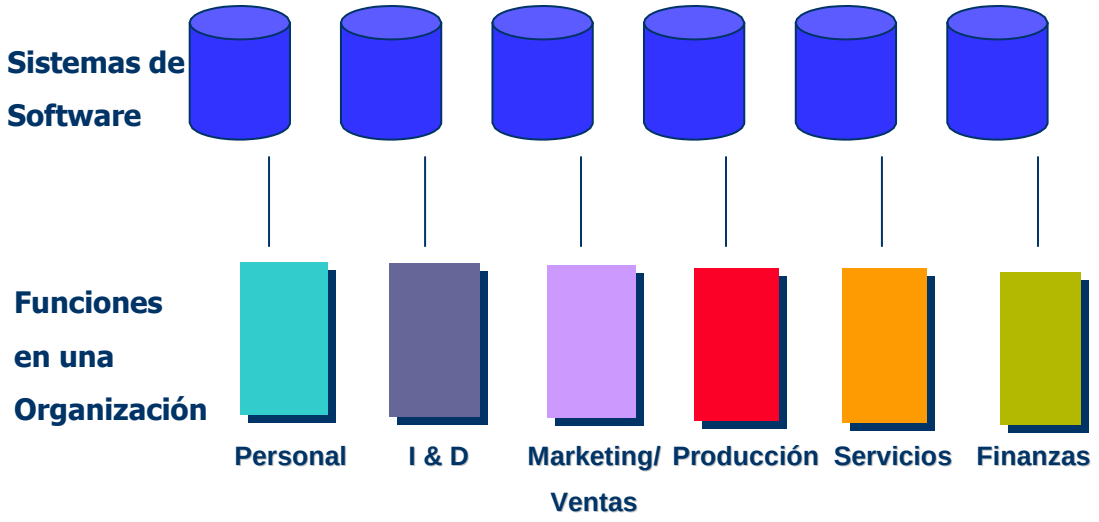


Figura 2.1 – Autonomía de Sistemas [52]

En la figura 2.1, se observa como cada función de servicio de una organización es soportada por un sistema autónomo.

Otro problema [52], que dificulta la integración de los sistemas es la existencia de la heterogeneidad, se producen debido a la incompatibilidad de las representaciones de modelos que cada sistema posee. Por decir, generalmente cada sistema posee una visión en particular del dominio de la aplicación al cual pertenece, incluye modelos de información, y también estructuras de datos.

La heterogeneidad de sistemas tiene las siguientes características:

- **Heterogeneidad Tecnológica:** Diferentes tecnologías de hardware y software utilizadas.
- **Heterogeneidad Sintáctica:** Diferentes formatos y estructuras de datos.
- **Heterogeneidad Semántica:** Diferentes representaciones de dominio.

En ese contexto, se podría decir que esta plataforma de sistemas no facilita la integración de los mismos, para proveer el soporte requerido por los procesos de negocio de una organización. Por lo tanto, estos sistemas autónomos y heterogéneos, demandan la construcción de "puentes" que puedan integrarlas.

2.2 Aspectos a considerar para la Integración de Sistemas

La importancia de la integración de sistemas, es poder permitir que los sistemas puedan interoperar, compartir informaciones y procesos de forma transparente, sin tomar en cuenta las diferencias tecnológicas ni las diferencias de representación de información [57].

En [56] es definida la interoperabilidad como la habilidad de que dos o más componentes de software puedan cooperar, sin considerar sus diferencias en lenguajes de programación, interfaces y plataformas de ejecución.

Según [57], la integración de sistemas puede ser definida como los procesos de integración para crear un sistema mayor con el objetivo de proveer servicios integrales.

Entonces, se podría decir que para una solución de integración de sistemas se deben de especificar algunas infraestructuras genéricas para poder permitir la coordinación y el intercambio de informaciones entre estos sistemas, de forma transparente [58]. Un objetivo adicional a alcanzar con una solución de integración de sistemas es el poder proporcionar una independencia de las tecnologías computacionales utilizadas en el desarrollo de los sistemas a ser integrados.

Para la integración de los sistemas existen, los componentes a ser considerados, tales como la conectividad física y la conectividad lógica, forman parte de una estructura de comunicación de sistemas.

A continuación, se definen los tipos de conectividad existentes:

- **Conectividad física:** Los sistemas son tomados en cuenta para que estos puedan operar en plataformas tecnológicas diferentes. Los aspectos físicos de conexión de sistemas caracterizan el uso de protocolos de software y mecanismos que interconecten los diferentes tipos de sistemas permitiéndoles a estos que puedan intercambiar información. Dentro de estos mecanismos de integración física podemos enumerar: gateways de Bases de Datos sobre ODBC (*Open Database Conectivity*), middlewares orientados a mensajes, interfaces de software, servicios de transporte de mensajes, y servicio de ruteamiento de mensajes.

- **Conectividad lógica:** Son los que ofrecen un mayor grado de complejidad, tienen que ver con los significados de las informaciones utilizadas por algún tipo de sistema. Es decir, describir como la información es utilizada, de forma diferente, por cada sistema.

Ya definidos algunos aspectos claves de integración, ahora mostramos una arquitectura de integración definida en dos capas básicas. (Ver figura 2.2)

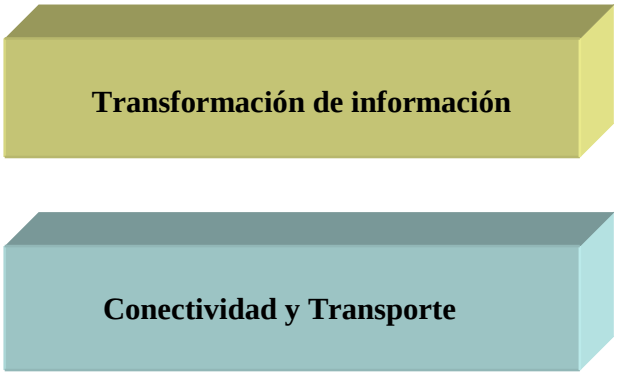


Figura 2.2 - Capas de funcionalidades para la integración de sistemas [52]

A continuación, se describe las dos capas funcionales que se mostraron en la figura 2.2:

- **Capa de Transformación de Información:** Esta capa tiene como principal función, modificar los formatos y las representaciones de las informaciones, de tal forma, que los sistemas receptores puedan utilizar las informaciones enviadas por otros sistemas.

- **Capa de Conectividad y Transporte:** Esta capa posee dos funciones básicas: (1) crear las conexiones necesarias entre los sistemas existentes, y (2) mover las informaciones de un sistema a otro, utilizando generalmente un mecanismo de intercambio de mensajes.

2.3 Estrategias de Integración de Sistemas de Software Heterogéneos

A lo largo de los años diversas estrategias están siendo propuestas para resolver el problema de integración de Sistemas Heredados [58] [59]. De las cuales se hace mención a las siguientes:

- **Un único proveedor** [59]: Esta estrategia implica construir un nuevo sistema, comprendiendo que todo sistema o subsistema sea de un único proveedor. Esta alternativa permite reducir substancialmente la heterogeneidad tecnológica, sintáctica y semántica, sin embargo presenta otros problemas. Generalmente, los proveedores utilizan tecnologías y procedimientos propietarios, que a largo plazo dificulta el mantenimiento y actualización de los sistemas. Otro problema en esta estrategia es que no siempre un único proveedor posee la experiencia y conocimiento necesario para atender a todas las necesidades de la organización.

- **Creación de Bases de Datos Centralizadas:** En esta estrategia todas las Bases de Datos de los Sistemas Heterogéneos existentes son almacenados dentro una única Base de Datos Centralizada [60]. Esta estrategia permite eliminar la duplicidad de los datos, pero aparece la existencia de modelos de datos propietarios y problemas de acceso a la Base de Datos, debido a que se incrementa el número de aplicaciones con acceso a las Bases de Datos Centralizadas, la cual se torna en un "cuello de botella". (Ver figura 2.3).

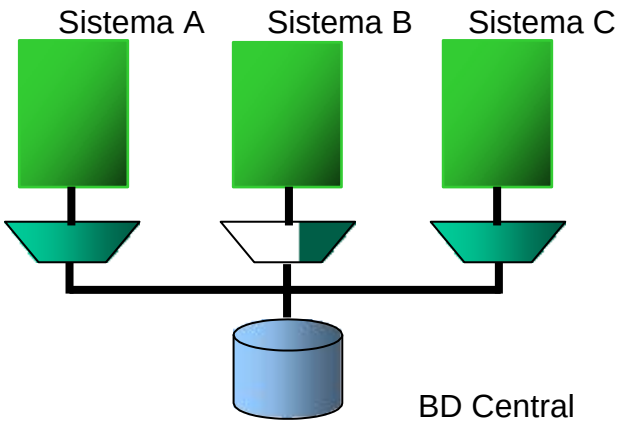


Figura 2.3 – Bases de Datos Centralizados [60]

- **Interfaces Punto a Punto:** Otra estrategia que ha sido ampliamente utilizada es la construcción de interfaces punto a punto para interconectar pares de sistemas según las necesidades de los usuarios [58] [59]. Esto permite que un sistema pueda invocar funciones de otro sistema, como si esas funciones fuesen parte del sistema que ejecuta la invocación. De forma general, una interfase es responsable por una visión interna de los servicios que un sistema opera. Las tecnologías de interfaces son diversas e incluyen adaptadores, conectores, gateways, etc. La principal diferencia entre estas tecnologías esta en la complejidad. Un conector punto a punto puede ser un bloque de programa "grande" que ejecuta traducciones básicas (manipulación de caracteres, rutinas de conversión de datos, filtrados de datos), suministra mecanismos de seguridad, ofrece acceso a las API´s de procesos, y sabe cómo usar diferentes servicios de transporte implementados. El desenvolvimiento de las interfaces punto a punto pueden ser una tarea que consume mucho tiempo, además de tener un alto costo, debido a que las interfaces precisan tener un amplio conocimiento de funcionamiento de los sistemas que se interrelacionan. (Ver figura 2.4).

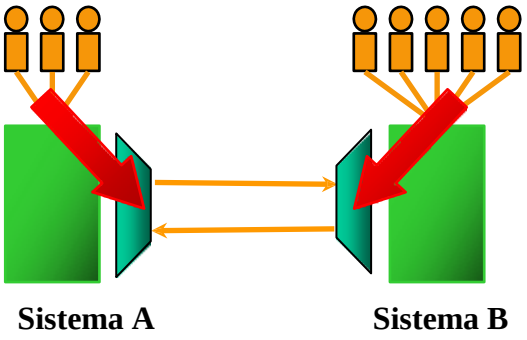


Figura 2.4 – Integración Punto a Punto [58] [59]

- **Middleware Orientado a Mensajes:** Con la integración punto a punto anteriormente mencionada, el número de interfaces necesarias crece espontáneamente. Esto puede ser reducido para un crecimiento lineal a través del uso de tecnologías de middlewares. Según [61], un middleware es un software de conectividad que pertenece a un conjunto de servicios para permitir que varios Sistemas Heterogéneos de diferentes plataformas puedan integrarse. De forma general, un middleware facilita la comunicación de pedidos de servicios entre sistemas a través de la utilización de mensajes o interfaces

definidas. La tecnología de middleware es más utilizada para propósitos de integración. El Middleware Orientado a Mensajes permite una comunicación asíncrona entre sistemas, utilizando mensajes y filas de mensajes. En esta estrategia cada sistema es conectado por un bus de mensajes a través de un adaptador o interfase. De esta forma, cada sistema tiene solamente una interfase, aquella que varía para el bus de mensajes. (Ver figura 2.5).

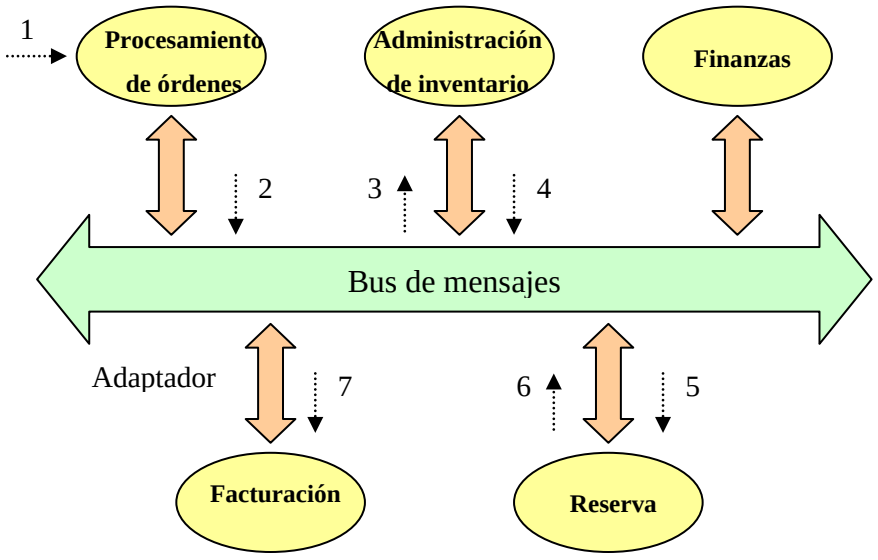


Figura 2.5 – Integración utilizando Bus de mensajes [61]

- **Message Broker:** Esta estrategia es una evolución de la anteriormente mencionada. Un controlador coordina los movimientos de información entre los sistemas integrados acorde con el flujo de procesamento definido para los sistemas [62]. En otras palabras, un controlador explícitamente coordina el flujo de comunicación entre sistemas que se interrelacionan. Un message broker es un ejemplo típico de un controlador o intermediador. Un message broker suministra la habilidad de encaminar y manipular los mensajes de los sistemas, de forma inteligente. Por ejemplo, un message broker podría recibir pedidos de compra de una aplicación Web y encaminar esos pedidos a uno o más sistemas, tomando en consideración una información de pedido. Un message broker, también, podría incluir un mecanismo de transformación

de información. De esta forma, la transformación seria hecha en un message broker y no en los adaptadores específicos de cada tipo de sistema. Un beneficio de utilización de un message broker es simplificar la construcción de interfaces adaptadores de conexión, debido a que elimina la necesidad de definir interfaces lógicas de manipulación de mensajes individuales entre cada par de sistemas independientes que están siendo integrados. De esta forma, se hace una modificación en el flujo de procesamiento de un sistema, solamente una interfase tendría que ser actualizada, aquella en el cual el sistema está siendo modificado. (Ver figura 2.6).

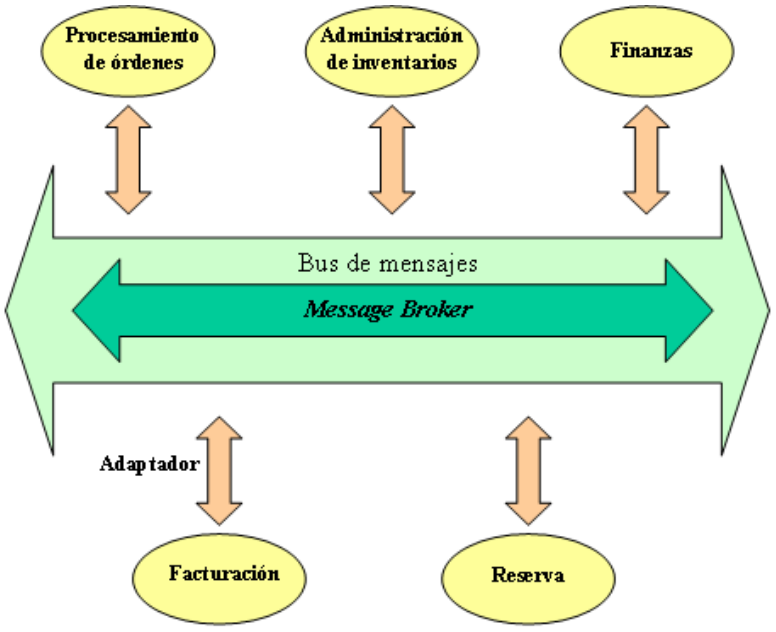


Figura 2.6 - Integración utilizando message broker [62]

2.4 Integración de Sistemas Heredados

Los **Sistemas Heredados** son aquellas aplicaciones que se caracterizan por ser antiguos debido al periodo de vida desde que estos sistemas fueron puestos en producción, son de tecnología obsoleta, es decir están siendo trabajados en una o muchas plataformas de hardware/software antiguos, no tienen técnicas de estructuración de sistemas, es decir son sistemas de tipo monolíticos al no estar constituidos entre arquitecturas de capas,

son de documentación pobre dificultando así la modificación de los mismos, y en la mayoría de los casos estos sistemas dan soporte a funciones de misión crítica dentro de una organización.

Los Sistemas Heredados [53], constituyen un activo para la organización pero, como todo activo, debe ser objeto de cuidados, mantenimiento y adecuación continua a las necesidades cambiantes de los negocios y la tecnología.

En la literatura, el término "Sistemas Heredados" se define como sigue: "Los Sistemas Heredados son aplicaciones importantes, imprescindibles para el funcionamiento normal de una organización y que están en producción desde hace cinco o más años" [53].

A los efectos de este trabajo se extiende el concepto de Sistemas Heredados incluyendo a los sistemas adquiridos o desarrollados por una organización para cubrir las necesidades de uno o más sectores críticos, con independencia del tiempo transcurrido desde su puesta en producción.

De esta manera, consideramos Sistema Heredado a todo sistema en producción cuyo funcionamiento es esencial para que una organización pueda operar normalmente en las actividades que dicho sistema atiende.

El hardware, los sistemas operativos, el software de base de datos y demás herramientas de base (a medida que se fueron extendiendo y aceptando los estándares) se han ido convirtiendo en "comodities". Mientras que los Sistemas Heredados son difícilmente sustituibles. Esto se debe a distintos factores: su alto costo de desarrollo, por ser dependientes de los fabricantes ("propietarios") o por haber requerido mucho tiempo de trabajo para adaptarlos a las necesidades de la empresa y fundamentalmente por ser los depositarios de las reglas de negocios y de los procedimientos operativos que competen a su rol [53].

De hecho, en general son sustituidos como consecuencia de una reingeniería de procesos de negocios y no por necesidades de adecuación tecnológica.

El concepto dominante de este trabajo con relación a los Sistemas Heredados, es que estos implementan funcionalidades importantes para la organización, las cuales deben ser reutilizables desde otros sistemas. Es necesario pues, estudiar la forma de realizar

esto, sin pérdida de identidad de los Sistemas Heredados y sin poner en riesgo la confiabilidad e integridad de sus procedimientos y datos.

Según [63] un Sistema Heredado puede ser definido como "un sistema de software amplio que no sabemos cómo tratarlo y que es vital para la organización". En [64] un Sistema Heredado es definido como "un sistema que significativamente resiste la modificación y evolución para atender nuevas necesidades de una organización". (Ver figura 2.7).

Figura 2.7: Sistemas Heredados
Fuente: Elaboración Propia

Para un mejor entendimiento, en la figura 2.7, se aprecia cómo están constituidos los Sistemas Heredados dentro de una organización: (Web-Help-Desk, Cliente/Servidor, Hojas de tiempo, 3270-Contratos, ASP.NET-Librería Virtual, Propietario-Contactos, etc.), estos sistemas trabajan de forma autónoma.

2.4.1 Problemática de Integración de Sistemas Heredados

Según [9] los sistemas informáticos van evolucionando a lo largo del tiempo conforme lo hacen tanto los modelos organizacionales, como los requerimientos operativos y para ello es necesario contar con herramientas que faciliten la adaptación de los sistemas conforme se van produciendo cambios.

En ese sentido, como parte de la plataforma de sistemas se encuentran los Sistemas Heredados, los cuales en la mayoría de las organizaciones, dan soporte a funciones de misión crítica dentro de la organización. Según [9], estos sistemas, en la mayoría de veces, componen el núcleo central de la infraestructura de tecnología de la información dentro de una organización. Por lo tanto, la adaptación e integración de este tipo de sistemas se vuelve necesaria.

Para la integración de Sistemas Heredados hay que tomar en consideración los siguientes aspectos [9]:

- **Distribución y heterogeneidad:** Cada una de las aplicaciones que integran un sistema pueden estar en distintos equipos, pueden operar sobre diversas arquitecturas compuestas por hardware, sistemas operativos, lenguajes de programación, herramientas para administrar datos, soporte de comunicaciones, etc.

- **Reuso:** En la medida en que las operaciones y servicios, pasen a ser reutilizables desde otros tipos de sistemas, nada impide que un servicio pueda ser utilizado desde diferentes tipos de operaciones, e incluso más de una sola vez dentro de la misma, mediante interfaces de programas bien definidos.

- **Documentación y estado del sistema:** La falta de actualización de la documentación, afecta al estado del sistema, los cuales pueden ser fuentes de problemas y errores.

- **Impacto de los cambios realizados en los Sistemas Heredados:** Hay que tener en cuenta un mecanismo de administración de versiones y notificación de nuevas funcionalidades.

- **Riesgos de alterar el funcionamiento del Sistema Heredado:** La pertenencia de un Sistema Heredado no debe afectar el funcionamiento interno de cada tipo de sistema.

- **Derechos de uso y autenticación de usuarios:** La administración de usuarios se torna algo muy complejo ya sea por la autenticación de los usuarios

como también la asignación de los derechos para cada uno de los servicios a los que estos acceden.

- **Seguridad y privacidad de los datos:** Existen ambientes operativos que requieren la utilización de transmisión de datos en entornos que pueden ser públicos, privados o mezclas de ambos.

- **Integridad transaccional:** Se deben instrumentar todo tipo de procedimientos para deshacer todas las operaciones ejecutadas satisfactoriamente antes de la ocurrencia de una falla.

- **Adaptar los Sistemas Heredados:** Para integrar Sistemas Heredados a un entorno de aplicaciones es necesario adaptar aquellas piezas de software cuyas funcionalidades sean requeridas por la organización.

- **Administración y gestión de sistemas:** Integrar sistemas es una tarea que requiere conformar equipos de trabajo, fijar metas comunes, acordar estándares, construir un equipo de dirección, instrumentar control de calidad a nivel de integración, administración de cambios y demás elementos propios del ciclo de vida de un sistema.

- **Son de gran tamaño.** Un sistema puede tener un total de millones de líneas de código fuente.

- **Técnicas y lenguajes de programación obsoletos.** Generalmente están codificadas en lenguajes Assembler, COBOL, C++, Visual Basic 6.0, y Power Builder.

- **Plataformas de hardware obsoletas.** Dan soporte a estos sistemas, factor que dificulta el poder hacer mantenimiento a estas plataformas.

- **El mantenimiento.** Es difícil y tienen un alto costo, debido principalmente a la falta de documentación y detalles de funcionamiento de estos tipos de sistemas.

- **Adición de nuevas funcionalidades.** Es una tarea muy difícil, muchas veces imposible.

- **Falta de interfaces bien definidas.** Para acceso de datos y funcionalidades de los sistemas, es uno de los principales obstáculos para la integración con otros tipos sistemas dentro de una organización.

Continuando con las definiciones de los problemas ya mencionados con anterioridad, los Sistemas Heredados son aplicaciones monolíticas que mezclan funciones de acceso a los datos, funciones de lógica de negocio y funciones de interfaces de usuario. En otras palabras, no existe una delimitación de funciones individuales. Esto no se torna difícil, si no imposible, separar o distribuir estas funciones en capas.

2.4.2 Estrategias de Integración de Sistemas Heredados

De acuerdo con [65] [66], existen tres estrategias que posibilitan la integración de Sistemas Heredados:

- **Redesarrollo.** Esta estrategia implica la concepción y desenvolvimiento de un nuevo sistema utilizando nuevas plataformas de hardware y arquitecturas modernas, para sustituir totalmente al Sistema Heredado existente.

- **Migración.** Esta estrategia propone el cambio gradual de los componentes del Sistema Heredado por otros componentes desenvueltos con nuevas tecnologías, preservando los datos y funcionalidades organizacionales del sistema.

- **Wrapping.** Esta estrategia implica la adaptación de los componentes existentes (datos, programas, interfaces) con nuevas interfaces de acceso a estos componentes. Estas nuevas interfaces son llamadas wrappers [67].

  Según la clasificación presentada por Harry Sneed [36] ordena los componentes de software del Wrapper de la siguiente forma:

  - **Nivel de Adaptación de Procesos/Jobs.** Un Job corresponde a un proceso batch, que toma uno o más archivos de input y produce un resultado sobre un archivo de output. (Ver figura 2.8).

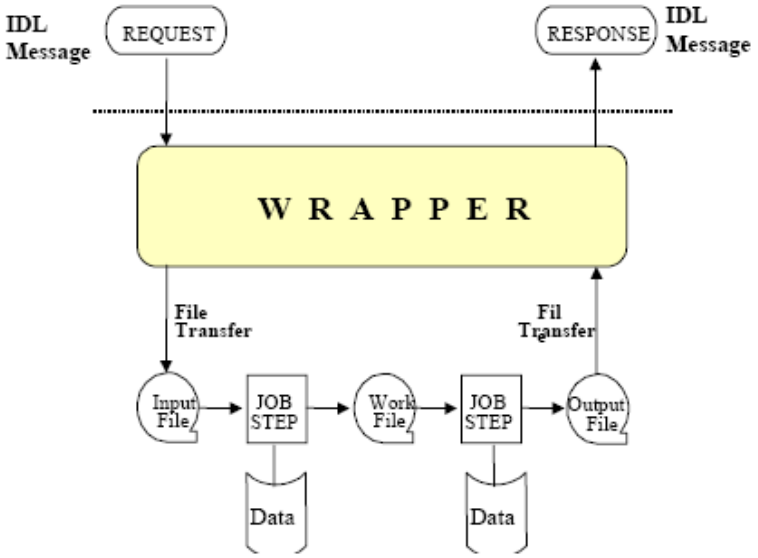


Figura 2.8: Adaptación de Jobs [67]

- **Nivel de Adaptación de Transacción.** Una transacción es una operación online invocada desde una terminal al enviar un input map y finaliza cuando la terminal recibe un output map. Un Wrapper deberá simular la terminal del usuario. (Ver figura 2.9).

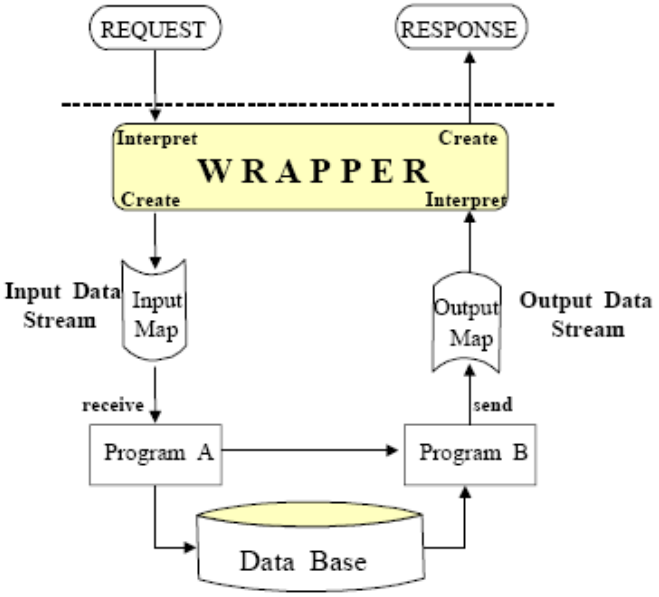


Figura 2.9: Adaptación de Transacciones [67]

- **Nivel de Adaptación de Programas.** Un programa es una unidad de proceso que se dispara mediante un procedimiento de control el cual le pasa los nombres de los archivos de input y output. Es similar al Job. No necesariamente es un proceso batch y puede interesar adaptar parte de sus funcionalidades. Este programa no tiene interfaz de usuario, ni tiene interfaz de sistemas, en consecuencia habrá que hacer algunos cambios en el código para poder adaptarlo. (Ver figura 2.10).

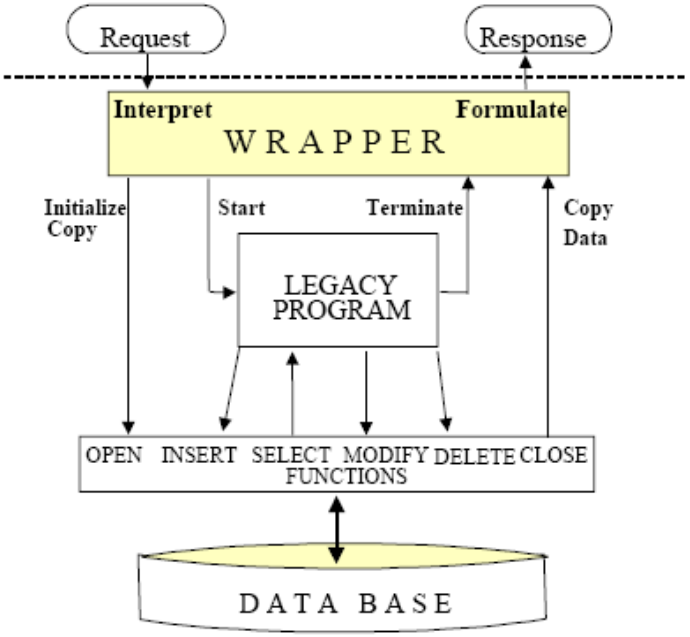


Figura 2.10: Adaptación de Programas [67]

- **Nivel de Adaptación de Módulos.** Un módulo es una pieza de software diseñada para ser invocada por otras. En consecuencia tiene su interfaz de sistemas definida. El Wrapper, adapta la interfaz a los requerimientos de arquitectura del sistema e invoca al módulo. (Ver figura 2.11).

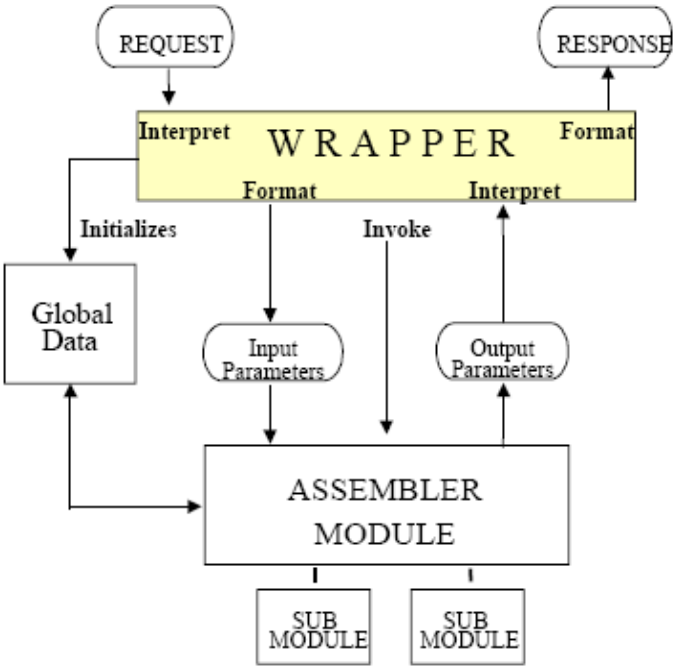


Figura 2.11: Adaptación de Módulos [67]

- **Nivel de Adaptación de Procedimientos.** Un procedimiento es una función codificada dentro de un programa a la cual se la pretende invocar en forma individual. Este tipo de funciones no tiene interfaz propia y por lo tanto hay que modificar el código fuente. Este tipo de modificación no es menor, no porque vaya a modificar la lógica del programa o de la función, sino porque implica un mayor grado de entendimiento del programa y está fuertemente afectado por la documentación existente. (Ver figura 2.12).

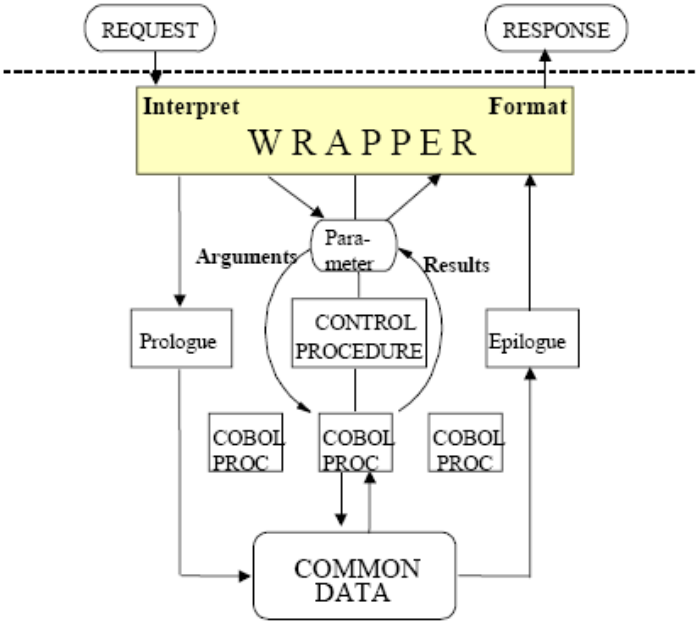


Figura 2.12: Adaptación de Procedimientos [67]

2.5 Adaptadores de Software

Básicamente la función de un adaptador de software o "adapter", es permitir adaptar una interfaz a otra a fin de que el objeto que es adaptado, pueda colaborar con otro que requiere una interfaz diferente [53]. Los adaptadores son una pieza fundamental para la interoperabilidad y el reuso de la lógica existente en una organización.

La adaptación consiste en hacer evolucionar solamente aquellas partes de un sistema que son necesarias para cumplir con requerimientos específicos, sin realizar cambios profundos y sin comprometer el comportamiento y la confiabilidad de los sistemas [53].

Los casos más típicos para aplicar técnicas de adaptación responden a necesidades de cambios [53]:

- En la presentación a los usuarios: adaptación de interfaz gráfica.
- En la forma de almacenar la información: adaptación de archivos a bases de datos.
- Necesidad de intercambiar información fluida con otros sistemas: Adaptación por requerimientos de interoperabilidad.

- Adaptación de funciones, para permitir su invocación desde otros ambientes y sistemas: Adaptación de funcionalidades mediante Wrapping (reuso).

Evidentemente, si en el marco de una política de evolución en la cual se marcan los objetivos tecnológicos a alcanzar, se establece un diseño de la arquitectura final deseada y si el estado en el que se encuentra el sistema lo permite, es posible mediante adaptaciones sucesivas evolucionar un sistema. De todas formas, es importante observar que en ninguno de los casos de adaptación que se señalaron anteriormente se menciona la realización de adaptaciones al núcleo central de los sistemas ni ha sido necesario profundizar en el entendimiento del sistema, ya que todas estas adaptaciones se realizan mediante técnicas de caja negra y caja gris valiéndose del uso de adaptadores de software [53]

En párrafos anteriores a este capítulo se analizó la técnica de adaptación, conocida como "wrapping", mientras que en capítulos posteriores se trabajarán específicamente los "wrappers" vinculados a la extracción de componentes a partir de Sistemas Heredados, concretamente adaptadores para programas y adaptadores para terminal (interfaz de usuario) [53]. (Ver figura 2.13).

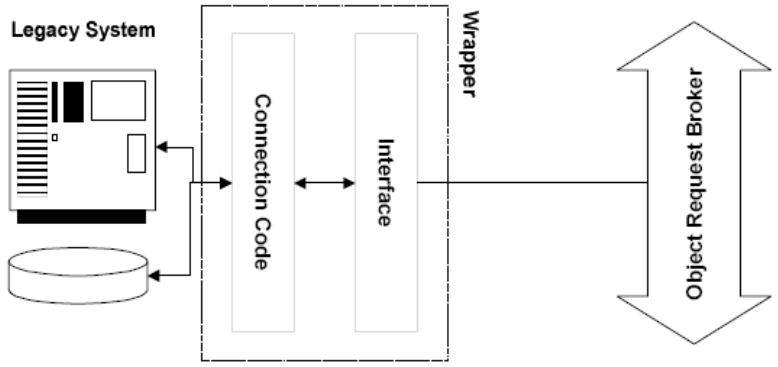


Figura 2.13. Wrapper de integración de Sistemas Heredados [67]

Este elemento Erich Gamma [31] lo presenta como "patrón" estructural según se muestra a continuación. (Ver figura 2.14).

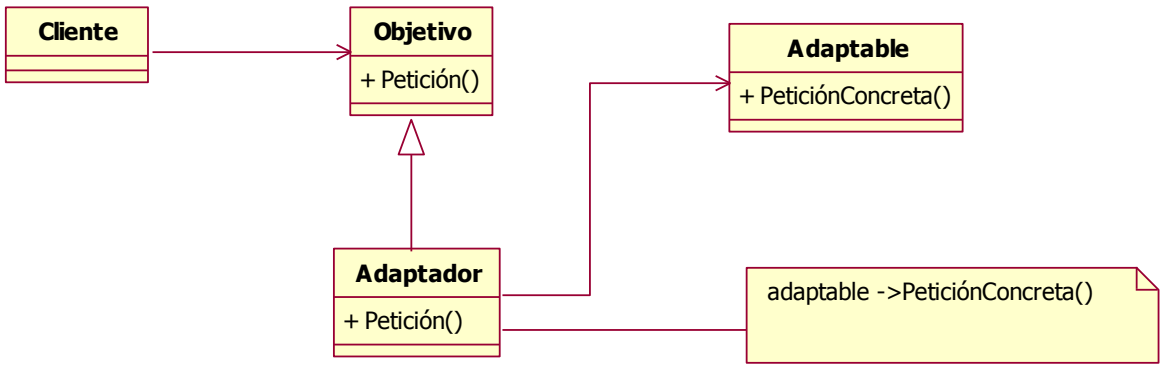


Figura 2.14. Patrón Adapter [31]

- **Objetivo**
  Define la interfaz específica del dominio que usa el Cliente.

- **Cliente**
  Colabora con objetos que se ajustan a la interfaz Objetivo.

- **Adaptable**
  Define una interfaz existente que necesita ser adaptada.

- **Adaptador**
  Adapta la interfaz de Adaptable a la interfaz Objetivo.

Esta es, sin duda, una técnica que posibilita la construcción de un "wrapper" de interfaz de usuario a partir de la aplicación de un programa que acepta la entrada/salida de otro programa, la interpreta y extrayendo los elementos importantes al intercambio de información, los pone a disposición de otras aplicaciones, mediante una interfaz de sistema adecuado y público[53].

Los distintos escenarios en los que se pueden presentar estos sistemas se establecen por combinaciones de elementos tecnológicos que vienen a conformar la arquitectura particular de cada uno. Otros elementos no menos importantes son la diversidad generacional y la calidad de la documentación [53].

Se presentan dos elementos claros que impactan fuertemente tanto sobre el diseño de la infraestructura que sustente la integración, como sobre los procesos de transformación que puedan ser necesarios para integrar los Sistemas Heredados a un entorno cooperativo [53].

- **La heterogeneidad** que requiere del diseño de soluciones particulares a cada caso, así como a la necesidad de recurrir a instrumentos adicionales de conectividad, básicamente gateways.

- **La documentación** que puede presentar un obstáculo importante y que debe ser analizada con particular rigurosidad antes de iniciar el proceso de integración y de cuyo estudio deben surgir recomendaciones claras y procedimientos concretos a fin de llevar la documentación al nivel de certeza necesario para no introducir errores respecto del funcionamiento actual. Existen procedimientos y herramientas de software que permiten combatir algunos de los inconvenientes derivados de la falta de documentación, por ejemplo mediante el uso de adaptadores de software "wrappers" de caja negra. Asociado a la documentación, pero con relativa independencia de ella, está el entendimiento del programa o sistemas, que tiene que ver con el conocimiento de lo que el sistema o programa efectivamente realiza y con qué finalidad. La asociación de estos elementos, sumado al grado de cumplimiento del sistema y sus programas respecto de las necesidades de los usuarios conforman el estado sanitario de un sistema.

2.6 Resumen del Capítulo

En esta segunda parte de la tesis, se han discutido los problemas que surgen para la integración de Sistemas Heredados. También de manera resumida, se detallaron las estrategias más utilizadas para la integración de sistemas. Las estrategias discutidas en este capítulo, se centran en los esfuerzos principalmente para la construcción de mecanismos de comunicación entre sistemas a ser integrados.

Un problema recurrente a estas estrategias, es el hecho de tener que modificar las interfaces, *message brokers*, o sistemas, cuando hay necesidad de modificar los

formatos de los mensajes, funciones de los sistemas, o el flujo de procesamiento de los sistemas. En caso de inclusión o levantamiento de sistemas, también existe la necesidad de hacer modificaciones en los componentes mencionados. Las estrategias no facilitan una operación de tipo *plug and play*.

Otro problema, es permitir que los sistemas puedan interpretar correctamente la información que estos reciben de otros sistemas, es preciso construir complejos procesos de transformación de información. La lógica de estos procesos de transformación puede estar integrada en interfaces o en un *message broker* según la estrategia a ser utilizada.

En el Capítulo 3, se dará a conocer la nueva tecnología de *Web Services*, como la principal solución al problema de integración de Sistemas Heredados.

CAPÍTULO 3
WEB SERVICES

En este tercer capítulo, se describen las características más importantes de la tecnología de *Web Services*. Los *Web Services* son servicios de informaciones para Internet. En la actualidad, esta tecnología está siendo utilizada en un amplio dominio de aplicaciones (comercio electrónico, intercambio de información, integración de aplicaciones, etc.) para la comunicación y manejo de informaciones de diversos tipos de sistemas dentro de una organización.

Los *Web Services*, se crearon con el fin de mejorar la interoperabilidad de los sistemas, haciendo uso de sus estándares, los cuales permiten un mejor funcionamiento dentro de un ambiente de Internet. Estos sistemas son desarrollados en múltiples lenguajes de programación y plataformas tecnológicas de aplicaciones. Los *Web Services*, soportan una colección de funciones que están empaquetadas como una unidad para ser publicadas en la Web, permitiendo un mejor rendimiento del mismo en las organizaciones.

3.1 ¿Qué es un Web Service?

Un *Web Service*, es una aplicación auto-contenida, auto-descrita, que puede ser publicada, localizada e invocada a través de la Web. Los Web Services, pueden ser invocados mediante el tipo de servicio dado, y estos son accedidos utilizando protocolos Web como: HTTP (*Hyper Text Transfer Protocol*) y XML (*eXtensible Markup Language*) [2].

Utilizar HTTP, como protocolo para las invocaciones en la Web facilita el uso de la infraestructura ya existente, en la actualidad prácticamente ya está siendo implementada en casi todas las organizaciones, para el intercambio de información y publicación de servicios del mismo. Asimismo, este intercambio es posible realizarlo sin tener que agregar más protocolos a los ya existentes en los *Firewalls* corporativos para poder

permitir estos flujos de información, lo cual sería necesario si se utilizara algún otro protocolo [5].

Los *Web Services*, incluyen un protocolo para el descubrimiento de los mismos [5], la cual se define en un formato estándar XML con la información de descubrimiento, para que los usuarios puedan recibir este tipo de formato, permitiendo descubrir los servicios que este presta mediante un URL (*Universal Resource Locator*). En los casos en los que los usuarios no conozcan el URL, también se incluye dentro del servicio un mecanismo UDDI de estándares para que los proveedores de servicios publiquen sus servicios y los consumidores los puedan encontrar.

A continuación, se definen algunas reseñas históricas que dieron inicio a la creación de los servicios de *Web Services* [53]:

- En octubre de 1994, se crea el *World Wide Web Consortium* (W3C), "a fin de liderar el desarrollo de la *Web* a su máximo potencial, desarrollar protocolos comunes y asegurar su interoperabilidad, velar por la estandarización mediante recomendaciones técnicas" [53]. Integrada en la actualidad por alrededor de 500 miembros, el W3C define los principios fundamentales de diseño de la Web: Interoperabilidad, Evolución y Descentralización.

- En 1998 se adopta el "*Extensible Markup Language*" (XML). El XML es la forma universal para documentos o datos estructurados sobre la *Web*. En la dirección http://www.w3.org/XML en el documento "*XML in 10 points*", la W3C expone brevemente las bases sobre las que se sustenta XML y las direcciones futuras. Hay dos factores principales que inciden en el desarrollo del XML: i) El "Business-to- Business" (B2B) e "e-Commerce" que requieren que las organizaciones que utilizan diferentes plataformas se comuniquen e intercambien datos. ii) Muchas instituciones necesitan ponerse a disposición de aplicaciones basadas en Web, información normalmente almacenada en sistemas de "back-office", como bases de datos y documentos [53].

- Gracias al respaldo de Microsoft e IBM. En Abril del 2002, se crea una nueva organización la "*Web Services Interoperability organization*" (WS-I) www.ws-i.org que se define a sí misma "como una organización abierta destinada a

promover la interoperabilidad de los *Web Services*, entre plataformas, sistemas operativos y lenguajes de programación diferentes". Esta organización trabaja con las industrias y organizaciones de estándares para satisfacer las necesidades de los usuarios como proveedores de ayudas, mejores prácticas y recursos para el desarrollo de soluciones basadas en *Web Services* [53].

La potencia de los *Web Services*, radica además de su gran interoperabilidad y extensibilidad, gracias al uso de XML, en que ellos pueden ser combinados para proveer operaciones más complejas. Varios programas proveen servicios simples que puedan interactuar, para permitir operaciones complejas. A fin de poder obtener una automatización completa de este tipo de interacciones, la arquitectura de los *Web Services* necesita mayor entendimiento y el desarrollo de algunas nuevas tecnologías [53].

Los *Web Services* [5], son un conjunto de estándares basados en: XML, SOAP, WSDL, y UDDI, que permiten la interoperabilidad entre sistemas con el HTTP como protocolo de transporte. (Ver figura 3.1).

Figura 3.1. Arquitectura de *Web Services* [5]

En la figura 3.1, se muestra la arquitectura del *Web Services*, el cual está constituido de la siguiente manera:

- **Descubrimiento:** La aplicación cliente que necesita acceder a las funcionalidades que expone un *Web Services* necesita una forma de resolver la ubicación del servicio remoto. Se logra mediante un proceso llamado normalmente descubrimiento (*discovery*). El descubrimiento se puede proporcionar mediante un directorio centralizado (UDDI).

- **Descripción:** Una vez que se ha resuelto el extremo de un *Web Services*, el cliente necesita suficiente información para interactuar adecuadamente con el mismo. La descripción de un *Web Services* implica meta datos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobre el *Web Services*, para ello describe sus servicios en un WSDL con un esquema estándar en XML.

- **Formato del mensaje:** Para el intercambio de datos, el cliente y el servidor tienen que estar de acuerdo en un mecanismo común de codificación y formato de mensaje. El uso de un mecanismo estándar de codificar los datos asegura que los datos que codifica el cliente los interpretara correctamente el servidor, utilizando para ello un servicio de mensajeria de petición y respuesta por medio de un protocolo estándar llamado SOAP.

- **Codificación:** Los datos que se transmiten entre el cliente y el servidor necesitan codificarse en un cuerpo de mensajes, para que estos puedan integrar aplicaciones, en múltiples lenguajes de programación y diversas plataformas tecnológicas, utilizando para ello un formato estándar XML.

- **Transporte:** Una vez que se ha dado el formato al mensaje y se han serializado los datos en el cuerpo del mensaje, se deben transferir entre el cliente y el servidor utilizando como protocolo de transporte HTTP.

3.2 Características del *Web Services*

Para poder integrar diversas aplicaciones que fueron construidas en distintas plataformas, el *Web Services* permite la integración de soluciones de negocio para una o varias organizaciones. Estos sistemas o aplicaciones que fueron desarrollados dentro

de la tecnología de *Web Services*, apuntan a formar parte de la nueva generación de aplicaciones de Sistemas Heredados.

A continuación, se presentan las principales características del *Web Services*:

- **Interoperabilidad:** Cualquier *Web Services* puede interactuar con otro *Web Services*. Como los *Web Services* pueden ser implementados en cualquier lenguaje, los desarrolladores no necesitan cambiar de ambientes de desarrollo para producir o consumir *Web Services*.

- **Comunicación:** Los *Web Services* se comunican utilizando protocolos HTTP y XML, como ya fue mencionado anteriormente. Por lo tanto cualquier plataforma que soporte estas tecnologías se pueden implementar y acceder por un *Web Services*.

- **Adaptar para reducir la complejidad:** Todos los componentes en un modelo de servicio Web son *Web Services*. Lo importante es la interfase que este servicio provee y no como se está implementado, por lo cual la complejidad se reduce.

- **Fácil de utilizar:** El concepto de los *Web Services* es fácil de entender, porque permite a los desarrolladores crear *Web Services* en forma rápida y fácil.

- **Soporte a la Organización:** Todas las empresas de software importantes utilizan SOAP, e incluso están impulsando el desarrollo de *Web Services*. Por ejemplo la nueva plataforma de Microsoft .NET está basado en *Web Services*, haciendo muy simple el desarrollo de los mismos desarrollados en ASP.NET.

Los *Web Services* presentan las siguientes ventajas:

- Las rutinas de los *Web Services*, se actualizan de forma transparente para el programador y para el encargado de mantener la aplicación.

- Mediante *Web Services*, se puede implementar al programa funciones imposibles de contemplar bajo el uso de rutinas de librerías, como por ejemplo, incorporar un buscador de páginas Web.

- La carga de CPU, que supone la ejecución de una rutina, desaparece al usar *Web Services*. La carga se reparte por Internet, sobre el servidor Web para el *Web Services*. Esto es un comienzo de "Computación Distribuida".

- Los *Web Services*, hacen uso de las mismas tecnologías que han sido atacadas en tantas ocasiones. La seguridad de una empresa puede verse comprometida. La ausencia de técnicas de seguridad estándar es un obstáculo para la adopción de la tecnología.

3.3 Estándares asociados al Web Services

A continuación, se presentan los principales estándares de apoyo a la tecnología de *Web Services*:

3.3.1 XML

El lenguaje XML (*eXtensible Markup Language*), es un lenguaje de marcas (etiquetas) usados para poder describir datos dentro de un formato en particular. Estos datos pueden leerse desde cualquier tipo de aplicación construida en cualquier plataforma tecnológica de software. El XML, permite transferir datos en un formato independiente de la plataforma. Por ello, el XML es una tecnología muy utilizada para transferir datos entre aplicaciones por la Internet. Los documentos XML, almacenan los datos en forma de textos. Esto hace que los documentos en XML se puedan leer fácilmente desde cualquier tipo de aplicación ya sean de distintos tipos de plataformas [24].

Asimismo, XML está estructurado de la siguiente manera:

- **DTD (*Document Type Definition*).** Es en general, un archivo/s que encierra una definición formal de un tipo de documento, y a la vez, especifica la estructura lógica de cada documento.

- **XSL (*eXtensible Stylesheet Language*).** Define o implementa el lenguaje de estilo de los documentos escritos para XML.

- **XLL (*eXtensible Linking Language*).** Define el modo de enlace entre diferentes enlaces.

La tecnología XML (*eXtensible Markup Language*), se ha ido desarrollando durante el transcurso de los años, según el autor Michael Morrison [51]:

- En los sesenta, la primera tecnología de información estandarizada y estructurada de cierta importancia fue SGML (Lenguaje de Marcado Generalizado Estándar) desarrollado por IBM.

- En 1986, el SGML fue aceptado como estándar ISO (Organización de Estándares Internacionales).

- En 1989, Tim Berners-Lee y Anders Berglund, crearon un lenguaje basado en etiquetas para documentos técnicos a fin de compartirlos en Internet, ampliado del sistema SGML llamado HTML.

- En 1996, la World Wide Web Consortium (W3C) propuso introducir el poder y la flexibilidad de SGML en el dominio de la Web, ofreciendo tres ventajas que faltaban en HTML: extensibilidad, estructura y validación.

- En febrero de 1998, la W3C introdujo su nueva tecnología de etiquetado llamado XML 1.0.

La tecnología XML, está compuesto por una serie de tecnologías de desarrollo en XML. (Ver figura 3.2).

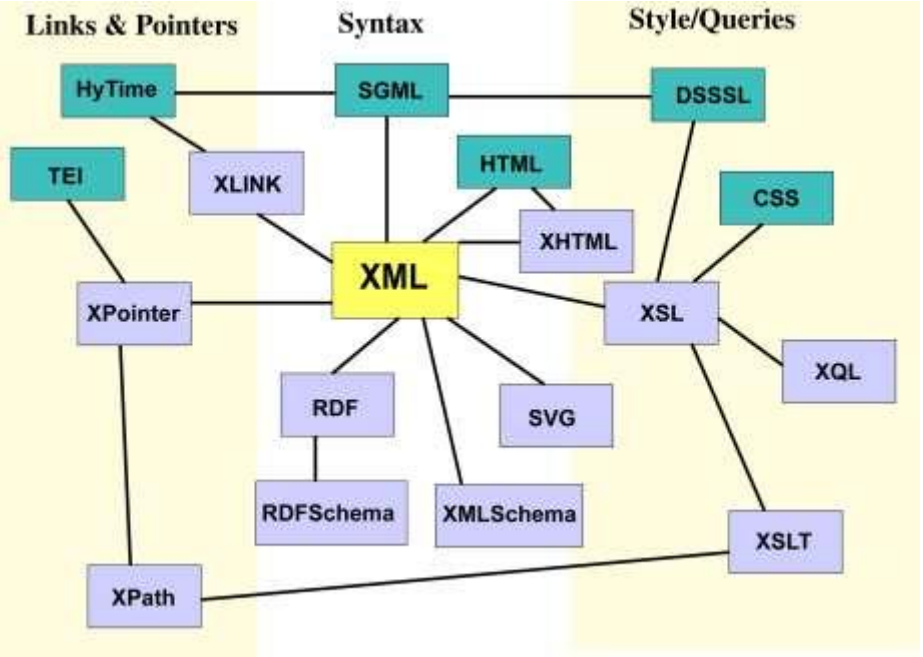


Figura 3.2: Familia de Tecnologías XML [34]

De esta manera, se puede definir que el lenguaje XML (*eXtensible Markup Language*), surge como un mecanismo para satisfacer las necesidades, para ser un formato universal de representación de datos, que permita el intercambio de información entre sistemas heterogéneos. Con el lenguaje XML, es posible representar cualquier tipo de datos estructurados o semi-estructurados, dentro de los cuales podemos enumerar: transacciones comerciales, catálogos de libros, reportes financieros, técnicos o estadísticos, datos de tipo gráfico, anuncios publicitarios, etc [47].

Un factor determinante en el suceso de la tecnología XML, es que más allá de pertenecer a un formato universal para representar datos, permite describirlos. Es decir, los datos son auto-descriptivos y pueden ofrecer todas las informaciones que los usuarios o aplicaciones necesitan para interpretar y procesar correctamente estos datos. Además, la tecnología XML pertenece a una amplia disponibilidad de herramientas de software que facilitan el rápido desenvolvimiento de las aplicaciones, haciéndose más atractiva su utilización [47].

El XML, es un lenguaje de marcación basado en texto, es un lenguaje que contiene elementos especiales (también llamadas de marcaciones o etiquetas) utilizados para describir la estructura y contenido de diferentes tipos de documentos, que serán utilizados principalmente por aplicaciones computacionales. Las marcaciones

proporcionan significado sobre el contenido de documentos. De esta forma, documentos escritos en un lenguaje de marcación consisten de texto que representan el contenido de documentos y marcaciones que adicionan estructura e información sobre ese contenido. Ejemplos de lenguajes de marcación son el HTML (*Hiper Text Markup Language*), RTF (*Rich Text Format*) y *PostScript* [47].

Los lenguajes de marcación en HTML, son solo de tipo presentación, esto quiere decir que en HTML las fuentes de datos son mostradas en el Internet de manera navegable y visual, más no como una estructura de fuentes de datos de registros [51].

Los lenguajes de marcación en XML son de tipo estructurado, esto quiere decir, que en la interfaz de aplicaciones y en el navegador de Internet se puede mostrar, ingresar, y también intercambiar fuentes de datos e información de sistemas [51].

Para ilustrar, el concepto de marcaciones y la diferencia que existe entre los lenguajes XML y HTML, a continuación se presenta un sencillo ejemplo de identificación de datos de una persona (Ver código 3.1).

- HTML:

```
<HTML>
<BODY>
<H3>Identificación </H3>
<UL>
<LI>Nombre: Lolo Fernandez</LI>
<LI>Sexo: Masculino</LI>
<LI>Fecha Nacimiento: 21/12/2000</LI>
<LI>Teléfono: 4777777</LI>
<LI>Dirección: Mayorasgo, Molina</LI>
</UL>
</BODY>
</HTML>
```

- XML:

```
< Identificación >
<Nombre> Lolo Fernandez </Nombre>
<Sexo>Masculino</Sexo>
<FecNac>21/12/2000</FecNac>
<Telefono>4777777</Telefono >
<Dirección> Mayorasgo, Molina </Dirección>
</Identificación>
```

Código 3.1. Ejemplo de Libreta direcciones (HTML-XML)

A primera vista, el lenguaje XML es similar en estilo al HTML, actualmente el XML es un súper conjunto de HTML y la mayoría de documentos HTML pueden ser considerados como documentos XML, con algunas excepciones. A XML, se le fue desenvolvida como respuesta a necesidades de una generalización de lenguajes HTML. Cabe mencionar, que tanto XML y HTML son derivados del mismo lenguaje "madre" SGML (*Standard Generalized Markup Language*). Sin embargo, el objetivo principal de HTML es describir como los datos tienen que ser mostrados en una pantalla. Además, HTML posee un conjunto fijo de marcaciones, pero no permite una fácil extensión de este lenguaje para atender necesidades específicas de usuarios o aplicaciones. En resumen, el lenguaje HTML, no está orientado a los datos, es solo una herramienta para presentación visual de páginas *Web*.

3.3.2 WSDL

Los *Web Services* [5], utilizan el WSDL (*Web Services Description Language*), para especificar los tipos de contratos de cada tipo de servicio que estas realicen dentro de una sintaxis que está basada en XML. Esta especificación permite conocer por otra parte que los consumidores utilizan métodos exportados, así como los parámetros y datos retornados.

El WSDL (*Web Services Description Language*), es un lenguaje de marcas (etiquetas), que definen a un *Web Services*. El WSDL, es un archivo de tipo estándar XML, que contiene la información sobre un *Web Services*. Esta información incluye los servicios *Web* llamados desde un *Web Services*, los métodos incluidos en cada uno de los *Web Services* y los parámetros que debemos agregarle a cada método. Además, el WSDL incluye información sobre los resultados devueltos al momento de procesar una petición de un *Web Services*. Por ejemplo, el WSDL define los tipos de valores devueltos por un método *Web*. Por ello, WSDL, es un vocabulario definido para crear un *Web Services* [24].

Además, cuando se llama a un método *Web*, tenemos que pasarle el nombre de usuario y la contraseña como parámetros. La información sobre el tipo y formato de los parámetros se almacena en un archivo WSDL. Cuando se procesa la petición y se devuelve el resultado, el WSDL almacena el formato y otra información sobre el resultado devuelto [24].

Al momento de almacenar información sobre los métodos *Web*, el WSDL, almacena la información sobre el tipo de formato empleado por el usuario para poder acceder a un *Web Services* y especificar la ubicación en la que se encuentra disponible ese *Web Services*. Por ello, el WSDL describe todo el mecanismo implicado en la transferencia de datos desde un cliente de *Web Services* hasta el servicio de *Web Services* y viceversa [35]. (Ver figura 3.3).

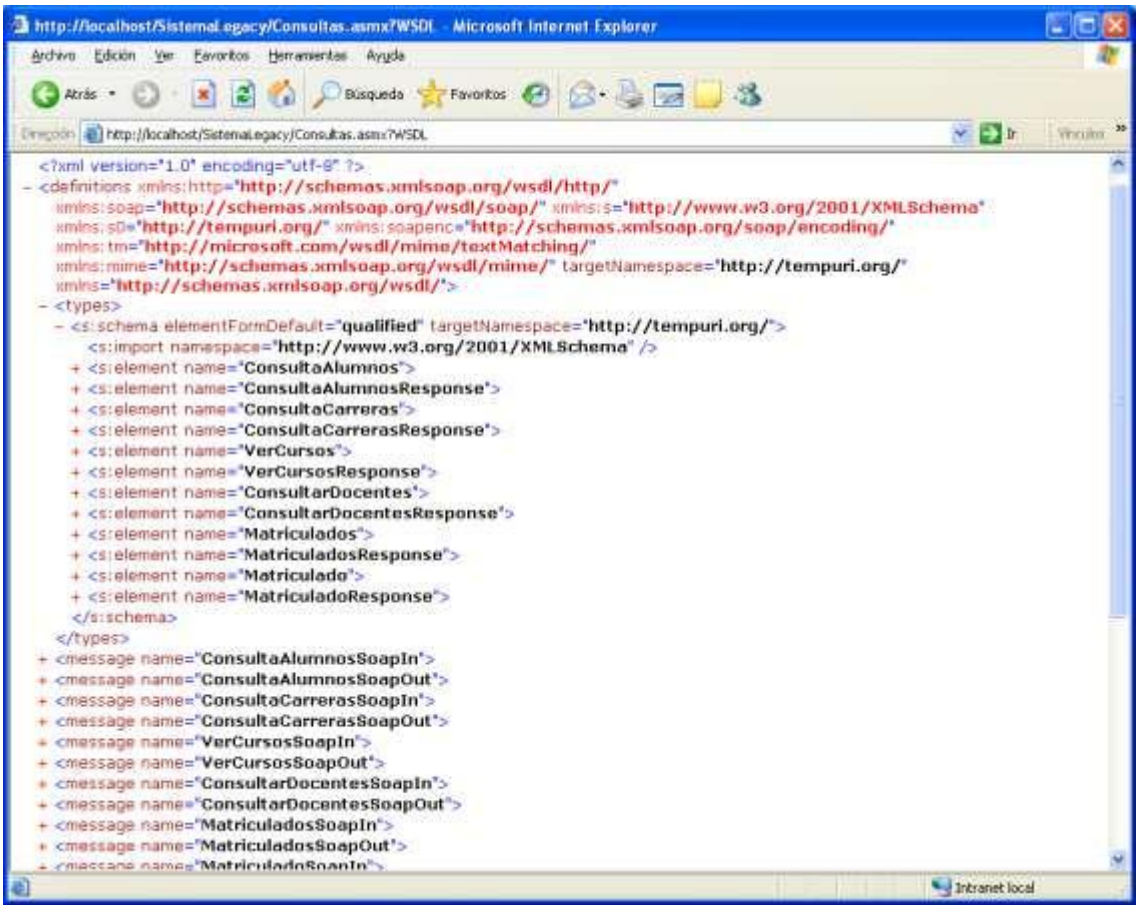


Figura 3.3. WSDL de servicios de Web Services

Fuente: Elaboración Propia

En la figura 3.3, se observa la descripción de los servicios *Web* de implementación de la presente tesis "Integración de Sistemas Heredados utilizando *Web Services*", la cual se mostrara en capítulos posteriores. Esta información incluye los servicios *Web* llamados por un *Web Services* a modo de prueba de servicios y presentados en un formato estándar XML.

3.3.3 SOAP

La comunicación de los *Web Services*, se basan en el intercambio de paquetes a través del SOAP (*Simple Object Access Protocol*) [5]. El SOAP, es un protocolo basado en XML, permite el intercambio de información. El SOAP, está compuesto por cuatro componentes básicos: i) un "envoltorio" que define cómo describir los mensajes y cómo procesarlos, ii) reglas para serializar (codificar) instancias de datos, iii) una

especificación de cómo representar invocaciones remotas y sus respuestas, y iv) una conversión para relacionar mensajes con un protocolo de transporte (habitualmente HTTP).

Para transferir datos entre un cliente de *Web Services* y un servidor de *Web Services* y viceversa, se emplea el protocolo de transferencia SOAP. El SOAP, es un protocolo basado en estándares XML, que utiliza una aplicación de tipo cliente para acceder a un *Web Services*. Además del XML, el SOAP utiliza el HTTP para transferir datos. Cuando un cliente envía una petición se realiza por medio de un mensaje SOAP. El mensaje SOAP, también incluye los parámetros y las sentencias de llamada al método. Basándose en esta información del mensaje SOAP, se llama al método Web apropiado [24]. (Ver figura 3.4).

Figura 3.4. Funcionalidad del SOAP [24]

En la figura 3.4, se muestra la funcionalidad del SOAP y sus procesos de petición y respuesta de mensajes de servicios del *Web Services* a través de la *Internet*.

El SOAP tiene las siguientes ventajas:

- No está asociado con ningún lenguaje.
- No se encuentra fuertemente asociado a ningún protocolo de transporte.
- No está atado a ninguna infraestructura de objeto distribuido.
- Aprovecha los estándares existentes en la industria.
- Permite la interoperabilidad entre múltiples entornos.

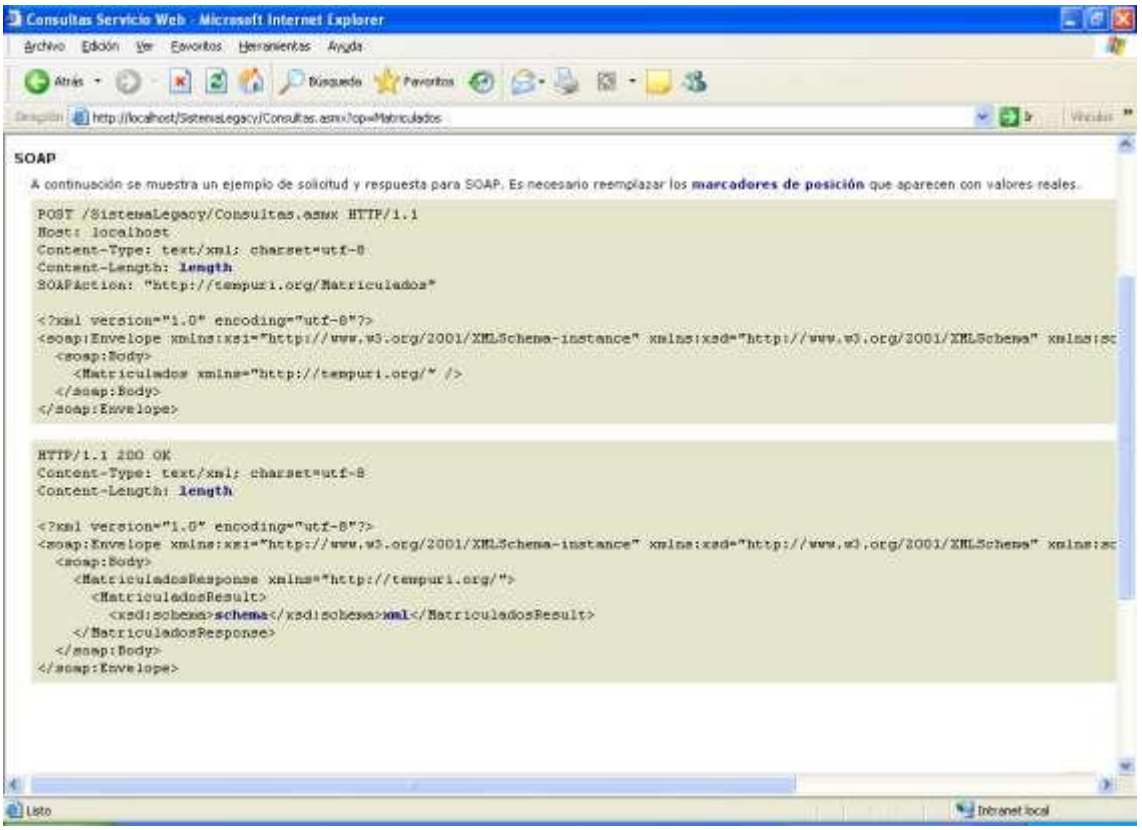


Figura 3.5. SOAP para servicios de invocación de Matriculados en el Web Services

Fuente: Elaboración Propia

En la figura 3.5, se observa la funcionalidad del SOAP, para uno de los servicios de la implementación de la presente tesis "Integración de Sistemas Heredados utilizando *Web Services*", la cual se mostrara en capítulos posteriores. Esta información incluye los procesos de petición y respuesta de invocación de servicios de Matriculados llamados por un *Web Services*.

3.3.4 UDDI

El *Web Services*, utiliza el UDDI (*Universal Description, Discovery and Integration*) [5], para proveer un mecanismo de páginas amarillas que permitan la localización de los servicios a través de la Internet.

Cuando se desarrolla un Web Services, se tienen que registrar estos servicios dentro de en un directorio llamado UDDI. El UDDI proporciona un mecanismo para que los proveedores registren sus Web Services. Cuando se registra un Web Services, en un directorio UDDI, este crea una entrada para el Web Services. Un directorio UDDI, mantiene un archivo estándar XML por cada Web Services registrado. Este archivo XML, contiene un puntero al Web Services registrado en el directorio. Además, el directorio UDDI, también contiene punteros al documento WSDL de un Web Services. Para ello, el proveedor del Web Services tiene que describir antes el Web Services en un documento WSDL. Una vez creado este documento, se puede registrar el Web Services en el directorio UDDI [35].

El directorio UDDI, devuelve la lista de Web Services que se están registrando en él. El usuario podrá seleccionar entonces el método Web requerido de la lista de Web Services disponibles [35].

El UDDI, contiene tres tipos de páginas necesarias para la información de un Web Services. i) Las páginas blancas, contienen información sobre la organización que proporciona el Web Services. Esta información incluye el nombre, dirección y otros datos de contacto de la empresa que proporciona el Web Services, ii) Las páginas **amarillas**, de un directorio UDDI contienen la información sobre las empresas organizadas en forma geográfica. iii) Las páginas verdes, proporcionan la interfaz de servicio para las aplicaciones cliente que accedan al Web Services. [35]. (Ver figura 3.6).

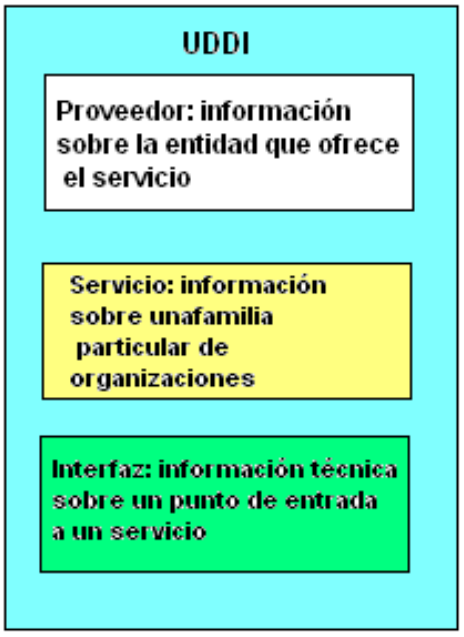


Figura 3.6. Arquitectura tradicional de un UDDI Fuente: Elaboración Propia

El UDDI, provee de un mecanismo para que los negocios se "describan" a si mismos y los tipos de servicios que proporcionan y luego se puedan registrar y publicar en un registro UDDI. El registro del UDDI, trabaja de la manera siguiente:

Definir la entrada de UDDI

- Determina los Modelos (archivos WSDL), que utilizan las implementaciones del Web Services.
- Determina el nombre de la empresa y una breve descripción de la misma en varios idiomas, si es necesario, así como los contactos principales para los que el Web Services ofrece.
- Determina las categorías e identificaciones adecuadas para la empresa.
- Determina los Web Services, que la empresa ofrece a través de UDDI.
- Determina las categorías adecuadas para los servicios.

Registrar la entrada de UDDI

- Una vez finalizada la tarea de definición, el siguiente paso consiste en registrar la empresa. Deberá obtener una cuenta con un registro UDDI.

Buscar la entrada en UDDI

- Es recomendable realizar tres comprobaciones una vez registrada la entrada en UDDI.
- Buscar la empresa por su nombre y categorizaciones.
- Asegurarse que este referenciada a la dirección Web.
- Luego de 24h, busca el servicio en Internet.

3.4 Resumen del Capítulo

En este capítulo, se han discutido las características más importantes del Web Services. La creación de un Web Services es una solución ideal para cualquier tipo de organización, ya que ofrece una serie de servicios a diversos usuarios y organizaciones que mantienen sus fuentes de información de manera dispersa. Estos servicios de Web Services, se realizan por medio del acceso a la Web. En este capítulo, se ha visto el papel que desempeñan: XML, WSDL, SOAP y UDDI en un Web Services.

En el Capítulo 4, se presentará la propuesta de una arquitectura a utilizar para la estrategia de integración de Sistemas Heredados utilizando Web Services.

CAPÍTULO 4

INTEGRACIÓN DE SISTEMAS HEREDADOS UTILIZANDO WEB SERVICES

4.1 Introducción

En el capítulo 2, se definieron las estrategias existentes para la integración de Sistemas Heredados. Debido a que las estrategias, no tuvieron los resultados deseados, dado que los nuevos sistemas no tenían las mismas funcionalidades de los Sistemas Heredados. En el presente capítulo, se describe la motivación para la utilización de una nueva estrategia, como solución al problema de integración de Sistemas Heredados. También, se mostrara la arquitectura propuesta, basada en el uso de la tecnología de Web Services.

4.2 Motivación

En la actualidad, muchas organizaciones se encuentran adaptando sus procesos de negocios hacia los nuevos escenarios económicos y tecnológicos. En ese sentido, la modernización e integración de sus sistemas de software son tareas indispensables para lograr ese objetivo. Un problema a solucionar es la presencia de los llamados Sistemas Heredados, como parte del portafolio de sistemas que las organizaciones poseen.

En el capítulo 2, se ha descrito la existencia de tres estrategias para la integración de los Sistemas Heredados. Una característica común de estas tres estrategias, es que no permiten reutilizar las funcionalidades existentes en otros tipos de sistemas. En ese contexto, y dado que los Sistemas Heredados poseen funcionalidades que serían convenientes reutilizar, se hace necesario explorar a una nueva estrategia de integración que permita esto.

En ese sentido, la aparición de nuevas tecnologías computacionales, como por ejemplo Web Services, permiten que las funcionalidades de los sistemas existentes puedan ser reutilizadas en el desarrollo de nuevos sistemas.

Una característica interesante de la tecnología de Web Services, es que permite que los sistemas puedan "exponer" sus servicios, utilizando la tecnología estándar de Internet, para cualquier otro tipo de sistema que lo necesite sin que sea necesaria su modificación. (Ver figura 4.1).

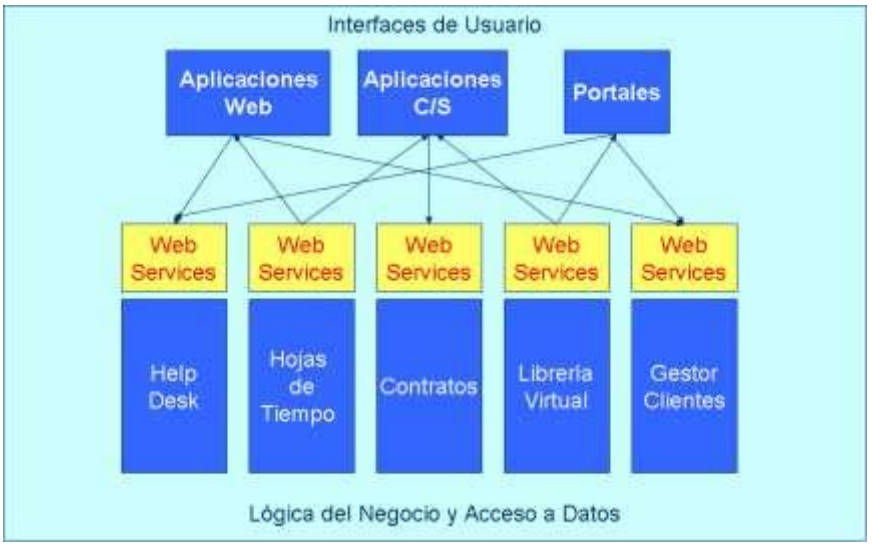


Figura 4.1: Integración de Sistemas Heredados usando Web Services

Fuente: Elaboración Propia

En la figura 4.1, se muestra un proyecto de integración, en la cual los desarrolladores crean interfaces para cada tipo de aplicación, para poder permitir una mejor comunicación entre sus sistemas. Debido a que cada interfase, para cada tipo de aplicación es autónoma, se integran grandes números de aplicaciones, y se genera una compleja red que requiere de niveles especializados para su desarrollo, incrementando los costos de integración y también incrementando la complejidad de los procesos de los negocios en las organizaciones. Si a este problema, se reduciera el número de interfaces, se podría reducir la complejidad de integración de las aplicaciones. En ese sentido, los Web Services, permiten que los sistemas puedan "exponer" sus servicios a cualquier otro sistema, a través de una interfaz única que pueda ser usada por cualquier tipo de aplicación. Estos sistemas, pueden ser invocados directamente como una interfaz tradicional.

4.3 Arquitectura conceptual de Integración de Sistemas Heredados Utilizando Web Services

A continuación, se muestra la arquitectura conceptual, propuesta para la utilización de la nueva estrategia de integración de sistemas utilizando Web Services.

(Ver figura 4.2).

Figura 4.2. Arquitectura Conceptual de Integración de Sistemas Heredados utilizando Web Services

Fuente: Elaboración Propia

La arquitectura conceptual mostrada, es la propuesta para la implementación de la nueva estrategia de integración de Sistemas Heredados utilizando Web Services, la cual se describirá en el capítulo 5. Esta estrategia permite reutilizar las funcionalidades del Sistema Heredado existente, permitiendo realizar llamadas desde el Web Services hacia una llamada de tipo convencional. El Web Services, se encargara de adaptar las funcionalidades para poder exponerlas en entornos de tipo Internet, utilizando para ello un componente llamado Wrapper, permitiéndose a los servicios poder ser integrados y consumidos por nuevos tipos de sistemas.

A continuación, se muestra la funcionalidad de la estrategia de Web Services. (Ver figura 4.3).

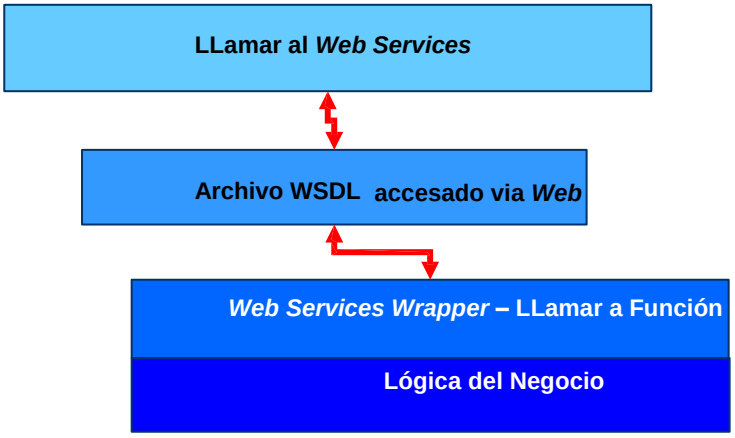


Figura 4.3: Arquitectura de Funcionalidad de la estrategia de integración de Sistemas Heredados usando Web Services

En la figura 4.3, se observa la funcionalidad de la estrategia de Web Services, el cual es iniciado desde el envió de una petición por parte del cliente para obtener un servicio, es decir llamar al Web Services. Esta petición se realiza por medio de mensajes a través del SOAP entre el cliente y el servidor del Web Services. Los servicios del Web Services, se encuentran dentro de un repositorio llamado UDDI, los cuales son definidos en interfaces especificados en WSDL dentro del estándar XML. El Web Services (Wrapper) se encargara de realizar llamadas a los servicios adaptados (Funcionalidades del Sistema Heredado), permitiendo la integración de los mismos, para poder ser consumidos por nuevos tipos de sistemas.

4.3.1 Uso de la Técnica Wrapping para la Integración de los Sistemas Heredados

Una de las técnicas para migrar Sistemas Heredados, a nuevos tipos de sistemas, es realizando llamadas a las funcionalidades del Sistema Heredado existente, mediante el uso del Wrapper. La idea de wrappear, las funcionalidades del Sistema Heredado para ser reutilizado, es un concepto que fue introducido por primera vez por W. Dietrich en

1989 [53] y desde entonces se ha ido ampliando y difundiendo. Actualmente, la necesidad de incorporar a Internet servicios proporcionados por Sistemas Heredados, conjuntamente con la mayor difusión de Corba, J2EE, COM y .NET, ha impulsado notoriamente esta idea.

Mediante wrapper, es posible adaptar al Sistema Heredado, para que sea parte de una nueva generación de sistemas, sin los riesgos inherentes a la reescritura y los altos costos de nuevos desarrollos. Es evidente, que para que esto no introduzca problemas a futuro, el estado de salud del sistema debe ser "sano" o debe ser factible pasarlo a dicho estado. Otro de los aspectos importantes de esta técnica, es que permite adaptar partes de un sistema según las necesidades y las urgencias, manteniendo el resto inalterado.

El Wrapping, es una estrategia parecida a la de Message Broker, debido a que utiliza un controlador que coordina los movimientos de información entre los Sistemas Heredados a ser integrados mediante mensajes.

4.4 Resumen del Capítulo

En este capítulo, se ha conocido la estrategia de Web Services, como nueva solución al problema de integración de Sistema Heredados. Por ello, los Sistemas Heredados, incluyen aplicaciones creadas en distintas plataformas y escritas en distintos lenguajes. Para crear soluciones, es esencial integrar estos sistemas. La integración de Sistemas Heredados, creados en distintas plataformas es más fácil con el uso de Web Services.

Un Web Services, es un componente reutilizable, como un método, que puede ser utilizado por cualquier aplicación Web que se ejecute en Internet. Estas aplicaciones, se llaman aplicaciones cliente de Web Services. La aplicación que aloja al Web Services se llama proveedor de servicio Web.

En el capítulo cinco, se presenta la implementación de la estrategia de Web Services.

CAPÍTULO 5

IMPLEMENTACIÓN DE INTEGRACIÓN DE SISTEMAS HEREDADOS UTILIZANDO WEB SERVICES

5.1 Introducción

En el capítulo anterior, se dio a conocer la arquitectura propuesta para la implementación de la estrategia de integración de Sistemas Heredados utilizando Web Services. En este capítulo, se detallan los factores más importantes a tomar en cuenta, para la implementación de la estrategia de Web Services. Este capítulo, se estructura de la siguiente manera: En la sección 5.2, se describe el problema. En la sección 5.3, se describe la solución al problema. En la Sección 5.4, se muestra la arquitectura física. En la sección 5.5, se detalla la implementación de integración de Sistemas Heredados utilizando Web Services. En la sección 5.6, se describe el resumen del capítulo.

5.2 Caracterización del Problema

El problema radica en la existencia de un Sistema Heredado no integrado, en la cual se requiere de la reutilización de sus funcionalidades. Este sistema tiene algunas funcionalidades (alumnos, docentes, cursos, carreras, matricular alumnos), para un Sistema Administrativo de Escuela Académica. La aplicación esta desarrollada bajo la plataforma Visual Basic 6.0 y Base de Datos en SQL Server 2000. Ante la necesidad de reutilizar las funcionalidades del Sistema Heredado existente, se requiere de la utilización de una nueva estrategia que permita integrarlo, para poder ser consumido en el desarrollo de nuevos sistemas.

5.3 Descripción de la Solución

Como solución al problema anteriormente descrito, se requiere de la utilización de la tecnología de Web Services, como nueva estrategia de integración de sistemas. El Web Services, gracias al uso de sus estándares (WSDL, UDDI, XML y SOAP), permite la adaptación de diferentes tipos aplicaciones (Ejemplo: funcionalidades del Sistema Heredado) hacia un entorno de servicios tipo Internet, para que estas a su vez puedan ser publicadas y consumidas por nuevos tipos de plataformas tecnológicas.

5.4 Arquitectura Física de Integración de Sistemas Heredados utilizando Web Services

A continuación, se muestra la arquitectura física propuesta para la integración de Sistemas Heredados utilizando Web Services. (Ver figura 5.1).

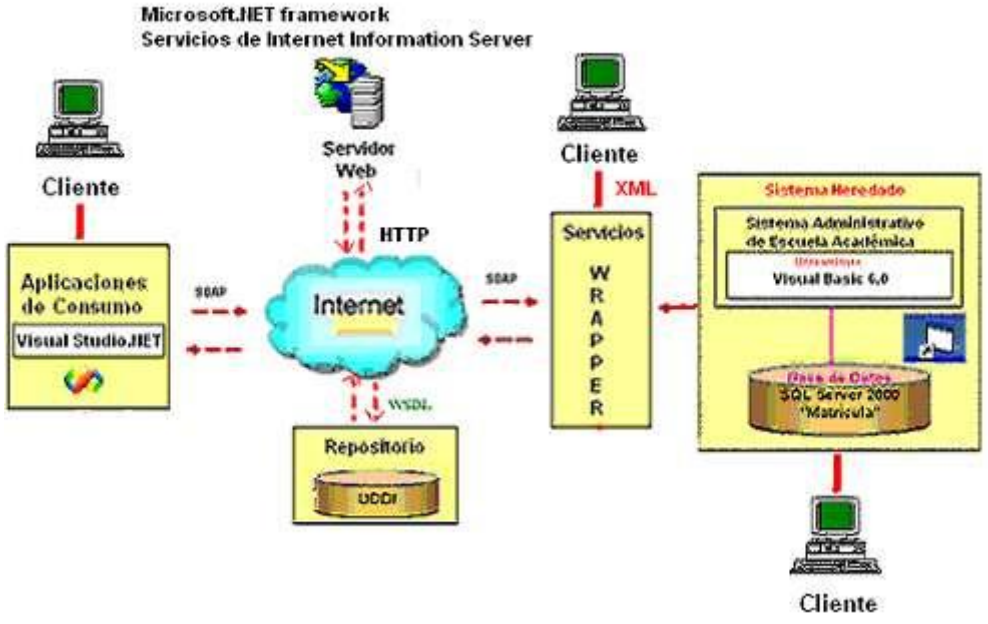


Figura 5.1. Arquitectura Física de Integración de Sistema Heredados Usando Web Services

La arquitectura física está compuesto de la siguiente manera: (1) **Sistema Heredado,** es un Sistema Heterogéneo, tiene algunas funcionalidades (alumnos, carreras, docentes y cursos, matricular alumnos) para un Sistema Académico Administrativo. Este sistema está desarrollado bajo la plataforma de Visual Basic 6.0 y un servidor de Base de Datos en SQL Server 2000 ("Matricula"), con sus tablas relacionales y Stored Procedure. (2) **Wrapper,** es un componente que permite extraer aquellos elementos importantes para el intercambio de información desde las funcionalidades del Sistema Heredado existente, este componente está incluido dentro de la lógica del negocio del **Web Services**, la cual permite realizar llamadas desde un Web Services hacia llamadas de tipo convencional, permitiendo adaptar y exponer las funcionalidades (alumnos, docentes, cursos, y carreras, matricular alumnos), para luego poder ser integrados y consumidos en el desarrollo de nuevos sistemas. El Web Services**,** esta implementado bajo la plataforma tecnológica de Microsoft. NET Framework y cuyo servidor Web, trabaja con directorios virtuales gracias al uso del Internet Information Server (IIS). (3) **Aplicaciones de Consumo**, es la implementación de una nueva aplicación, que permita consumir las funcionalidades adaptadas por el Web Services, a manera de consulta de servicios. Esta aplicación esta implementado en Visual C#.NET con interfaces de consumo.

5.5 Implementación de Integración de Sistemas Heredados utilizando Web Services

5.5.1 Objetivo de Implementación

El **objetivo** de la presente tesis, es la de poder demostrar de manera sistematizada el uso de la nueva estrategia de **Web Services,** como una solución al problema de integración de sistemas. Por lo cual, se pretende reutilizar las funcionalidades del Sistema Heredado existente (alumnos, carreras, docentes y cursos, matricular alumnos), realizando llamadas desde un Web Services hacia llamadas de tipo convencional, para poder adaptar las funcionalidades existentes hacia una nueva plataforma tecnología para su posterior consumo. (Ver figura 5.2).

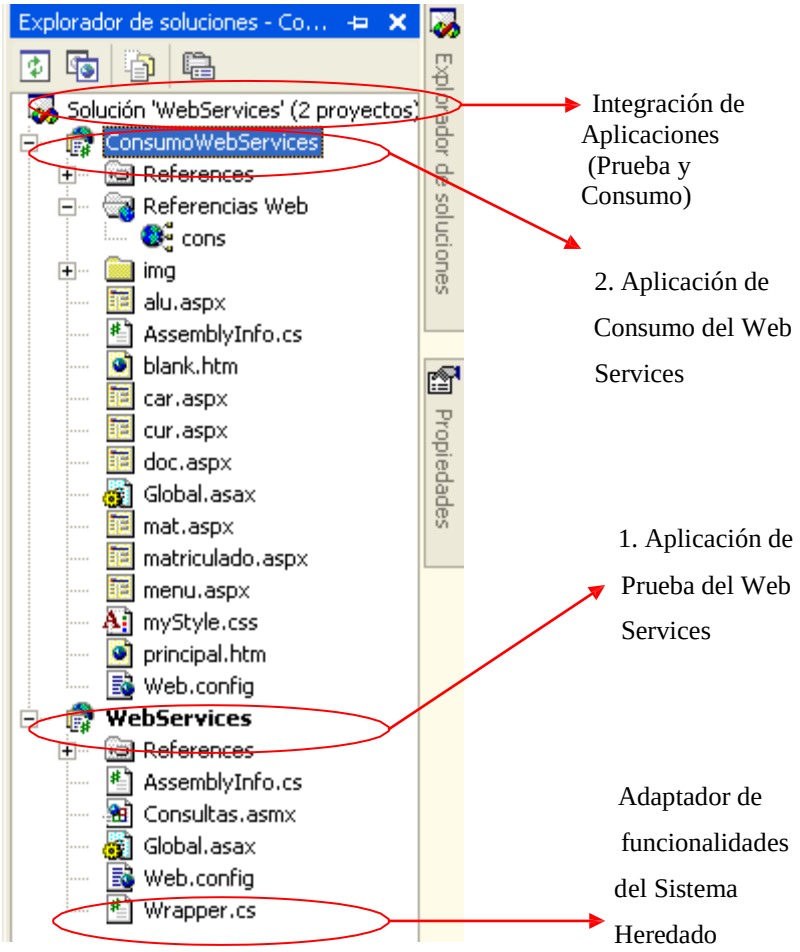


Figura 5.2. Integración de dos aplicaciones: Prueba (WebServices) y Consumo (ConsumoWebServices).

En el Anexo 1 (Página 73) se muestra el Modelamiento de la implementación utilizando UML. A continuación, se muestra el diagrama de flujo, con los procesos requeridos para la implementación de la estrategia de Web Services. (Ver figura 5.3).

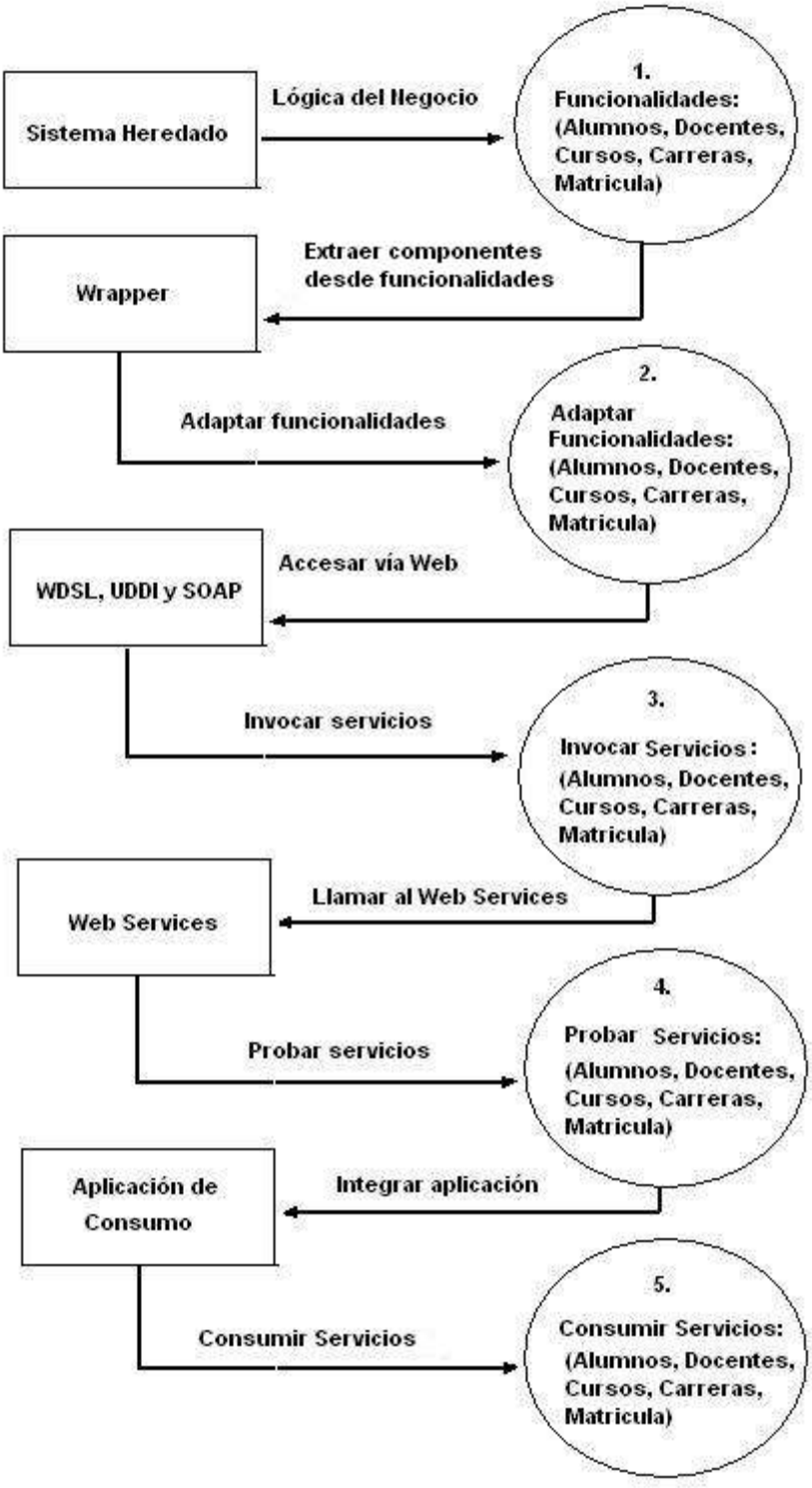


Figura 5.3. Diagrama de Flujo de la estrategia de Integración de sistemas Heredados utilizando Web Services

5.5.2 Sistema Heredado

Las interfaces, aplicaciones y servicios de Base de Datos de un Sistema Heredado, pueden ser considerados como componentes diferentes con interfaces bien definidas. Consisten en un conjunto de aplicaciones independientes, cada una de los cuales interactúa con los servicios de Base de Datos, y potencialmente con sus propias interfaces de usuario y de sistemas. Estas aplicaciones, son consideradas como aplicaciones Heterogéneas, ya que pueden estar creadas en distintas plataformas tecnológicas y escritos en distintos lenguajes de programación.

El Sistema Heredado, que se muestra en esta sección esta desarrollado bajo la plataforma de Visual Basic 6.0 y un servidor de Base de Datos en SQL 2000. La aplicación cumple con las siguientes funcionalidades:

- Registrar matricula de alumnos.
- Realizar un mantenimiento de registros de: alumnos, docentes, carreras y cursos.
- Generar reportes de: alumnos, docentes, carreras y cursos.

A continuación, se muestra un diagrama de flujo con las funcionalidades del Sistema Heredado. (Ver figura 5.4).

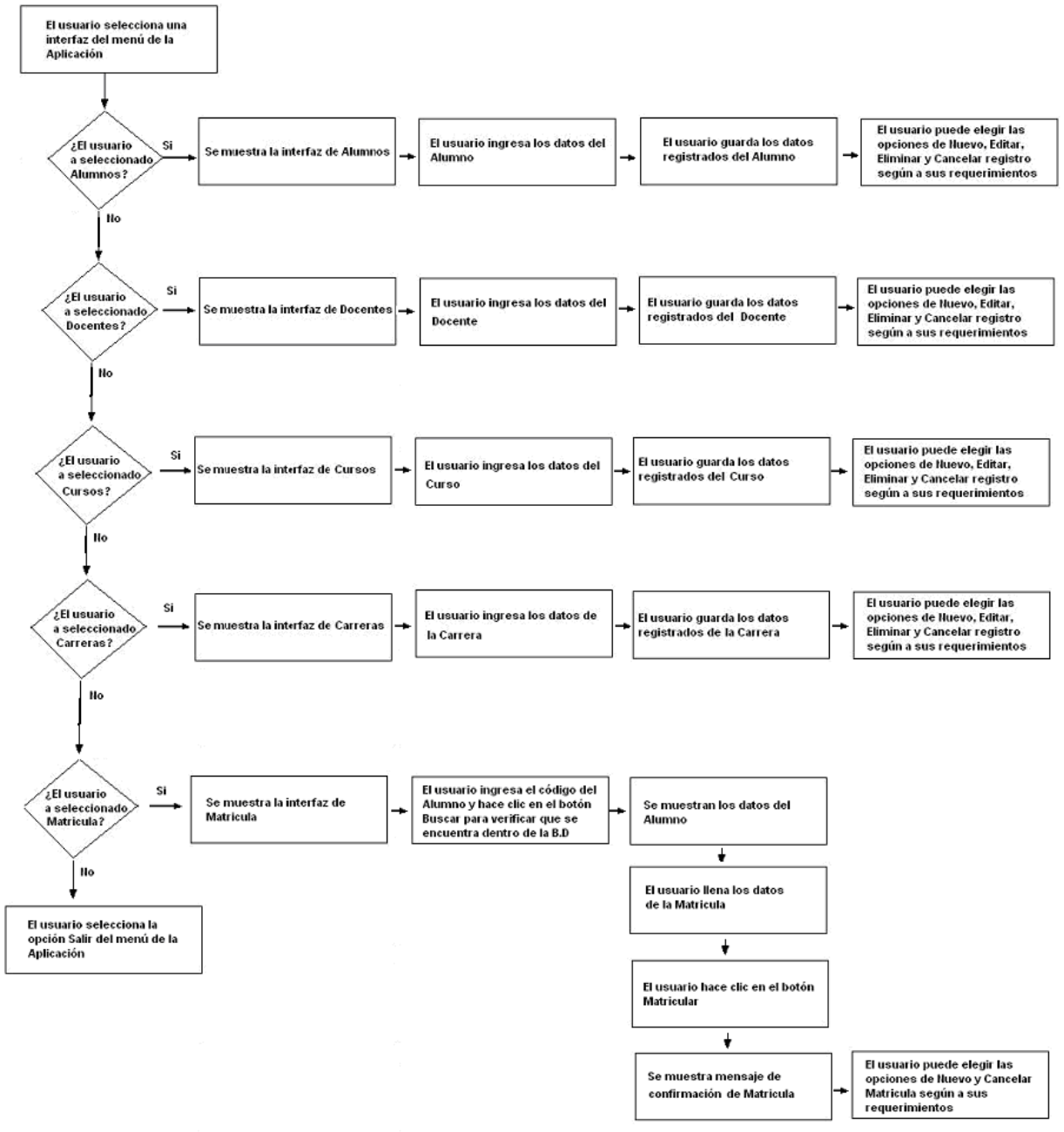


Figura 5.4. Diagrama de Flujo de funcionalidades del Sistema Heredado

Ya estando familiarizado con la importancia en cuanto a la funcionalidad del Sistema Heredado, el problema radica en poder reutilizar sus funcionalidades desde sus fuentes de datos, para que estas pudieran ser reutilizadas y expuestas en nuevas plataformas tecnológicas. Las interfaces del Sistema Heredado, están incluidas en el Anexo 2 (Página 108). A continuación, se muestra los componentes del Sistema Heredado. (Ver tabla 5.1):

SistemaHeredado	Ejecutable (SistemaHeredado.exe) desarrollado en Visual Basic 6.0, contiene las funcionalidades del Sistema Heredado.
MATRICULA	Data (Matricula) del Sistema Heredado, almacenada en un servidor de Base de Datos en SQL Server 2000, con tablas relacionales y stored procedures.

Tabla 5.1: Componentes del Sistema Heredado.

Desde el punto de vista tecnológico el camino más sencillo y para el cual existe una mayor cantidad de herramientas y trabajos realizados, es el acceder directamente a los datos del Sistema Heredado, para no perder la lógica del negocio del mismo.

A continuación, se presenta el diseño de la Base de Datos (Matricula) creada en SQL Server 2000, con sus respectivas tablas relacionales (Ver figura 5.5).

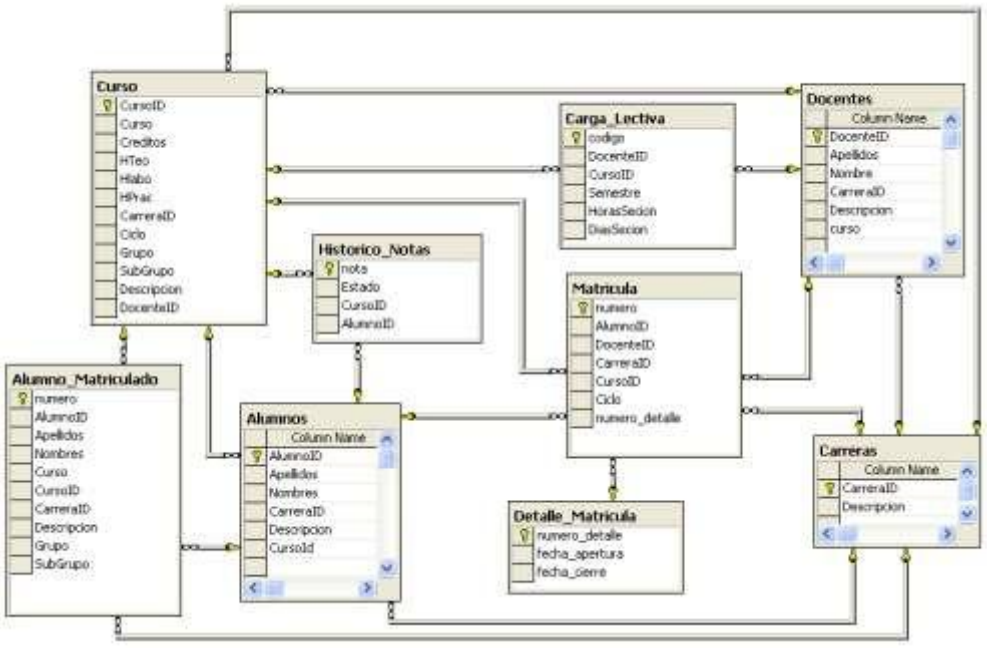


Figura 5.5. Diagrama de E-R de la Base de datos “Matricula” en SQL Server 2000

5.5.3 Web Services (Wrapper)

Los Web Services cuentan con la ventaja de que están soportados en forma nativa por cualquier plataforma, lo cual permite una solución de bajo costo y fácil de implementar. Por otro lado los Web Services tienen la desventaja que aún no permiten manejar transacciones entre plataformas, lo cual lleva a que la integración sea factible sólo para consultar información, no para la actualización. Una solución basada en Web Services permite la integración de cualquier plataforma.

La solución de implementar un Web Services, consiste en poder reutilizar las funcionalidades de un Sistema Heredado existente, extrayendo aquellos elementos importantes para la integración de la información, adaptando los mismos para permitir su exploración en el desarrollo de nuevos sistemas. Este prototipo, se encuentra implementado dentro del Servidor Web y bajo la plataforma de Visual Studio.NET, el cual gracias al uso del IIS, permite su navegación en entornos de tipo Internet (localhost).

El procedimiento de la implementación de la estrategia de Web Services es la siguiente:

1. Crear una clase Wrapper dentro de la lógica del negocio de la implementación del Web Services, la cual permitirá extraer aquellos elementos importantes dentro de las funcionalidades del Sistema Heredado existente, para poder adaptarlos, a través de sus fuentes de información de datos y poder llamarlos dentro de un "dataset" para establecer la conexión.

```
public class Wrapper //Creación de la clase Wrapper
{
private SqlDataAdapter da;
private DataSet ds = new DataSet();
private SqlConnection con;
...
...
```

Dentro del adaptador Wrapper, se procede a incluir los elementos a adaptar, es decir desde las fuentes de datos por medio de un dataset. El modificador de clase new, permitirá ocultar la clase heredada con el mismo nombre de la clase base.

```
...
...
public DataSet ConsultaAlumnos () //Adaptando funcionalidades de Alumnos
{
da = new SqlDataAdapter("spConsultaAlumno", con);
//oculta la clase heredada con el mismo nombre de la clase base.
...
...
}
...
...
}
```

2. Ya adaptados los elementos de las funcionalidades del Sistema Heredado, se procede a realizar llamadas desde un Web Services hacia llamadas de tipo convencional, por medio de sus métodos, permitiendo que los mismos puedan ser expuestos por el Web Services, y poder ser reutilizados en el desarrollo de nuevos sistemas para su posterior consumo.

```
[WebMethod] // Llamada de Web Services a llamada de tipo convencional.
public DataSet ConsultaAlumnos ()
{
Wrapper w=new Wrapper();
return w.ConsultaAlumnos();
}
```

3. La aplicación cliente que necesita acceder a las funcionalidades adaptadas que expone el Web Services necesita una forma de resolver la ubicación del servicio remoto. Se logra mediante un proceso llamado UDDI. El sitio Web en la cual se expondrán los servicios del Web Services es la siguiente:

   http://localhost/SistemaLegacy/Consultas.asmx

4. Una vez que se ha resuelto el extremo de un Web Services, el cliente necesita suficiente información para interactuar adecuadamente con el mismo. La descripción de un Web Services implica meta datos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobre el Web Services, para ello describe sus servicios en un WSDL con un esquema estándar en XML.

5. Para el intercambio de datos, el cliente y el servidor tienen que estar de acuerdo en un mecanismo común de codificación y formato de mensaje. El uso de un mecanismo estándar de codificar los datos asegura que los datos que codifica el cliente los interpretara correctamente el servidor, utilizando para ello un servicio de mensajería de petición y respuesta por medio de un protocolo estándar llamado SOAP.

6. Los datos que se transmiten entre el cliente y el servidor necesitan codificarse en un cuerpo de mensajes, para que estos puedan integrar aplicaciones, en múltiples lenguajes de programación y diversas plataformas tecnológicas, utilizando para ello un formato estándar XML, lográndose así la prueba de servicios del Web Services.

7. Una vez que se ha dado el formato al mensaje y se han serializado los datos en el cuerpo del mensaje, se deben transferir entre el cliente y el servidor utilizando como protocolo de transporte HTTP.

8. Ya habiendo definido y publicado los servicios del Web Services dentro de un Repositorio llamado UDDI, estos podrán ser encontrados por nuevas aplicaciones de consumo, para poder lograr esto, se hizo una referencia Web a las interfaces de prueba del Web Services.

En el Anexo 3, se encuentran incluidos las interfaces y códigos representativos del Web Services. (Página 115). A continuación, se muestra los componentes del Web Services. (Ver tabla 5.2):

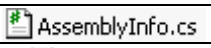
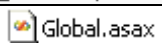
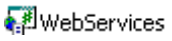
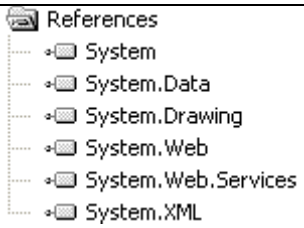
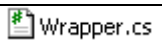
 AssemblyInfo.cs	Contiene los metadatos del ensamblado del proyecto.
 Global.asax	Contiene el código para gestionar los eventos generados en la aplicación.
 Web.config	Contiene información sobre los ajustes de los recursos ASP.NET.
 WebServices.csproj.webinfo	Contiene información sobre la ubicación del proyecto en el servidor de desarrollo.
 WebServices	Este archivo contiene los metadatos de la solución. Si el servidor de desarrollo es máquina local, este archivo existe en el servidor local.
 References System System.Data System.Drawing System.Web System.Web.Services System.XML	La carpeta de Referentes, contiene referencias a los espacios de nombres empleados para desarrollar el servicio de Web Services. Por ejemplo, la carpeta Referentes, contiene los ficheros System, System.Data, System.Drawing, System.Web,System.Web.Services, y System.XML que contienen referencias a los respectivos espacios de nombres.
 Wrapper.cs	Clase que contiene el código Wrapper, para realizar llamadas desde un Web Services, esta clase permite adaptar las funcionalidades del Sistema Heredado.

Tabla 5.2: Componentes del Web Services (Wrapper)

5.5.4 Aplicaciones de Consumo

Gracias a la utilizaron del Web Services, en cuanto a la adaptación de las funcionalidades del Sistema Heredado, se ha logrado poder reutilizar las funcionalidades de la aplicación antigua y exponerlas de manera de consulta hacia una nueva plataforma tecnológica como son las Aplicaciones de Consumo, referenciando de manera Web las interfaces del adaptador Web Services, para que estas puedan ser expuestas para su posterior consumo. Esta aplicación esta implementado en Visual C#.NET con interfaces de consumo.

Las interfaces de consumo, se encuentran incluidas en el Anexo 4 (Página 130). A continuación, se muestra los componentes de la Aplicación de Consumo.

(Ver tabla 5.3):

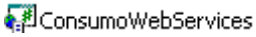
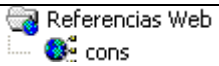
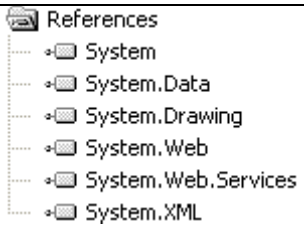
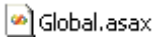
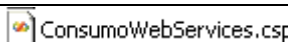
	Este archivo contiene los metadatos de la solución. Si el servidor de desarrollo es máquina local, este archivo existe en el servidor local.
	Esta carpeta contiene una referencia Web para exponer servicios (llama a los servicios de la aplicación Cliente del Web Services) desde interfaces de otros tipos de aplicaciones construidas en cualquier lenguaje de programación .NET de manera integrada
	La carpeta de Referentes, contiene referencias a los espacios de nombres empleados para desarrollar el servicio de Web Services. Por ejemplo, la carpeta Referentes, contiene los ficheros System, System.Data, System.Drawing, System.Web,System.Web.Services, y System.XML que contienen referencias a los respectivos espacios de nombres.
	Contiene los metadatos del ensamblado del proyecto.
	Contiene el código para gestionar los eventos generados en la aplicación.
	Contiene información sobre los ajustes de los recursos ASP.NET.
	Contiene información sobre la ubicación del proyecto en el servidor de desarrollo.

Tabla 5.3: Componentes de la Aplicación de Consumo

5.6 Resumen del Capítulo

En este capítulo, se ha demostrado de manera sistematizada la aplicabilidad del uso de la nueva estrategia de Web Services, para el problema de integración de Sistema Heredados. Esta estrategia demuestra como poder integrar un Sistema Heredado existente y adaptarlo hacia una nueva plataforma tecnológica como es el Web Services, para su posterior consumo. También, se demuestra la utilización del servicio Wrapper incluido en la lógica del negocio del servicio de Web Services.

En el Capítulo 6, se describen las conclusiones, recomendaciones y trabajos futuros de la presente tesis.

CAPÍTULO 6

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

6.1 Conclusiones

- La aparición de nuevas tecnologías, obligan a las organizaciones a integrar sus Sistemas Heredados hacia nuevas plataformas tecnológicas, adaptando las funcionalidades de sus sistemas, para poder ser reutilizados en el desarrollo de nuevos sistemas.
- Existen estrategias de integración, que apuntan al reemplazo total (rediseño) o gradual (migración) de estos sistemas. En muchos casos, estos esfuerzos han fracasado dado que los nuevos sistemas no tenían las mismas funcionalidades de los Sistemas Heredados, debido principalmente a que la documentación era muy pobre. Una estrategia más simple, es la de dotar de un nuevo visual al sistema, es decir, desarrollar una especie de "envoltorio" (*wrapping*) que permita acceder a sus funcionalidades.
- Con el advenimiento del concepto de Reuso de Software y dado que los Sistemas Heredados poseen funcionalidades que han demostrado su confiabilidad a lo largo de los años, sería interesante "exponer" estas para que puedan ser utilizadas en la construcción de otros sistemas. En ese sentido, la aparición de nuevas tecnologías computacionales, como por ejemplo *Web Services*, permite que estas funcionales puedan ser reutilizadas en el desarrollo de nuevos sistemas
- Con el desarrollo de la presente tesis, se ha logrado obtener como resultados: i) Extraer aquellos elementos utilizados por las funcionalidades del Sistema Heredado existente, por medio de un componente incluido dentro de la lógica del *Web Services* llamado *Wrapper*, para poder realizar llamadas desde un *Web Services* hacia llamadas de tipo convencional, permitiendo la adaptación de los mismos. ii) Utilizando los estándares (WSDL, UDDI, XML y SOAP) se realizó la invocación de las pruebas de publicación de los servicios adaptados por el *Web Services*,

permitiendo que los elementos adaptados puedan ser expuestos a través de la *Web*. iii) Con la utilización del XML, se procedió a reutilizar e integrar los servicios adaptados, para poder ser consumidos en el desarrollo de nuevos sistemas de manera de consultas.

- La experiencia recogida en integrar Sistemas Heredados, y la poca bibliografía existente en la materia, han impulsado al desarrollo de esta tesis, como una nueva visión de conjunto a esta problemática. Se entiende, que este enfoque constituye un aporte interesante a la comunidad informática, ya que presenta elementos aplicables tanto para la integración de sistemas, como para reconstruir componentes a partir de Sistemas Heredados.

6.2 Recomendaciones

- Se deben migrar hacia nuevas plataformas tecnológicas, para el desarrollo de las organizaciones.
- Utilizar medios tecnológicos, para facilitar la búsqueda de información para la toma de dediciones.
- Utilizar la tecnología de *Web Services*, para accesar documentos y fuentes de información por la *Internet*.
- Utilizar la presente Tesis, como un aporte de documentación referente a la utilización de la tecnología de *Web Services* como estrategia de integración de Sistemas Heredados.

6.3 Trabajos Futuros

En todo trabajo de tesis, van surgiendo una serie de elementos que no se pueden incluir en el mismo, algunos de ellos, fueron mencionados en texto y ejemplos, trabajarlos constituye un compromiso con este proyecto. Hay otros aspectos vinculados al tema, que surgen como consecuencia de los estudios realizados, que no deberían quedar sin profundizar. Esta tesis ha puesto énfasis en el uso de la tecnología de *Web Services*, como una nueva estrategia de integración de sistemas, para dar soporte a procesos

críticos de las organizaciones. Es por ello que esta estrategia se mantendrá como un plan piloto, para que esta idea pueda ser aprovechada para nuevas investigaciones y puesta en marcha para su implementación en mejora de las organizaciones e instituciones académicas.

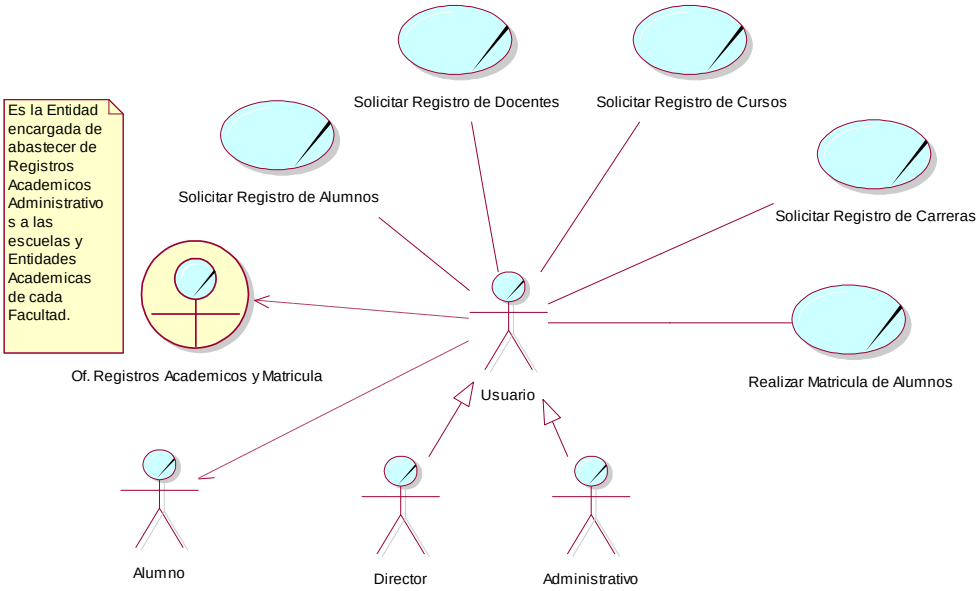
ANEXOS

ANEXO 1

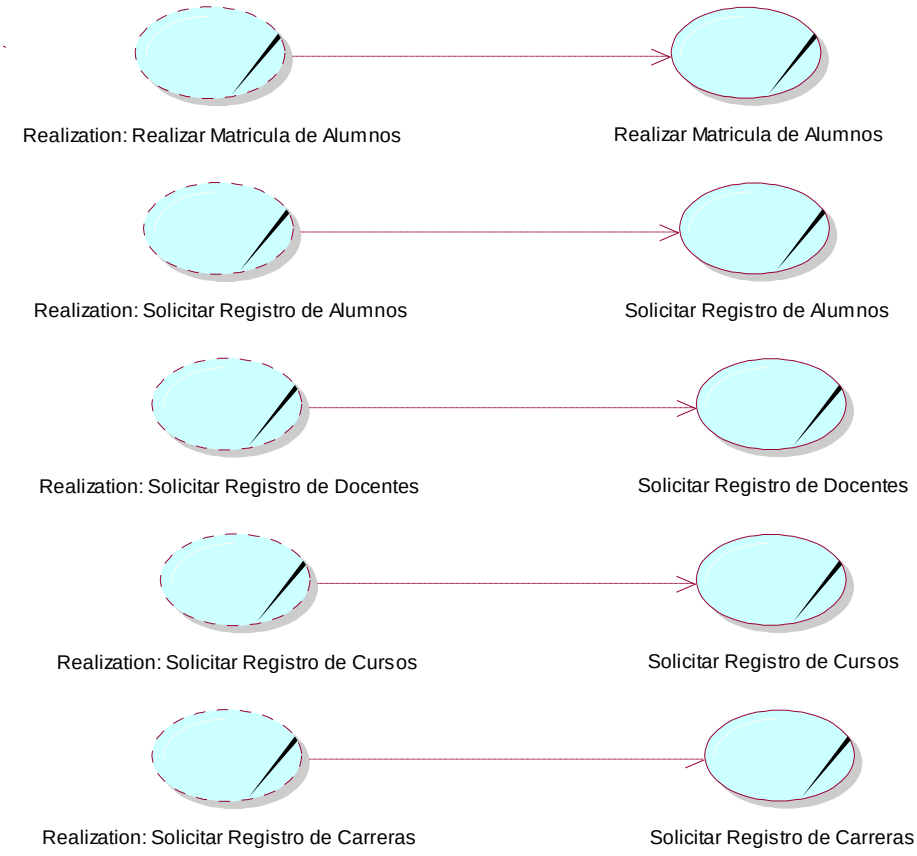
MODELADO DE LA IMPLEMTACIÓN DE INTEGRACION DE SISTEMAS HEREDADOS UTILIZANDO WEB SERVICES

Modelo del Negocio

a) Vista de Casos de Uso del Negocio

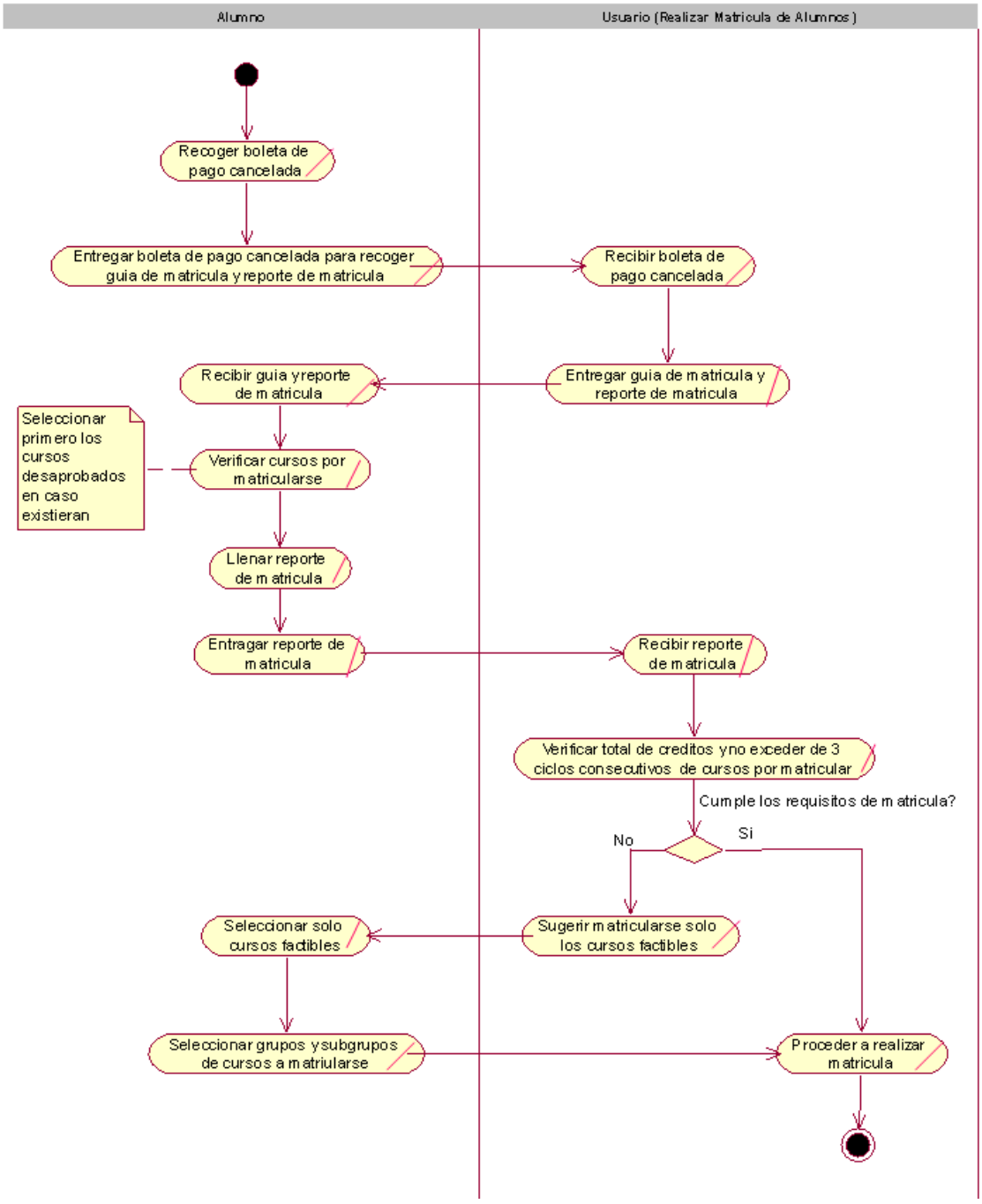


b) Vista de Realizaciones de Casos de Uso del Negocio

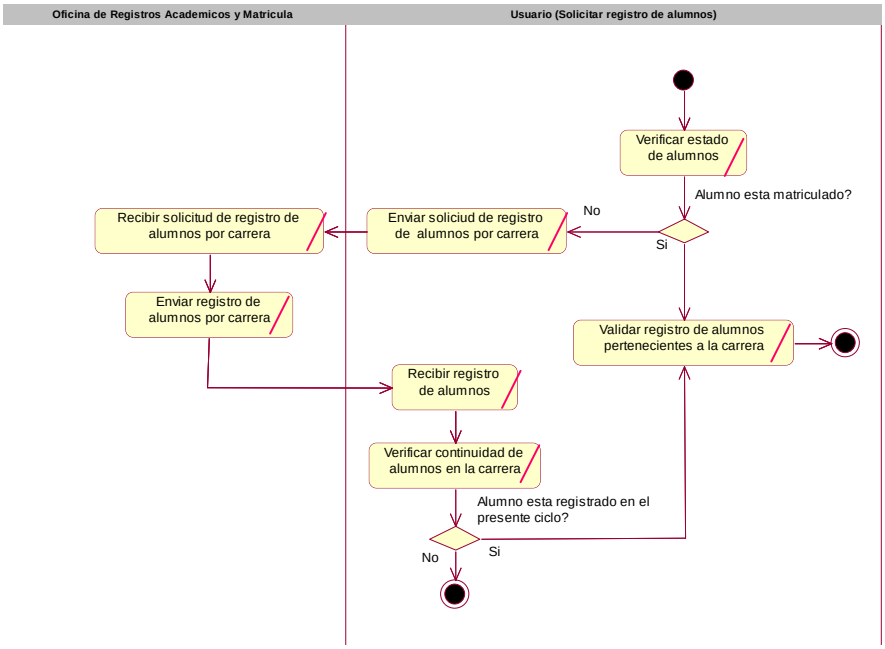


c) Vista de Diagrama de Actividades de Casos de Uso del Negocio

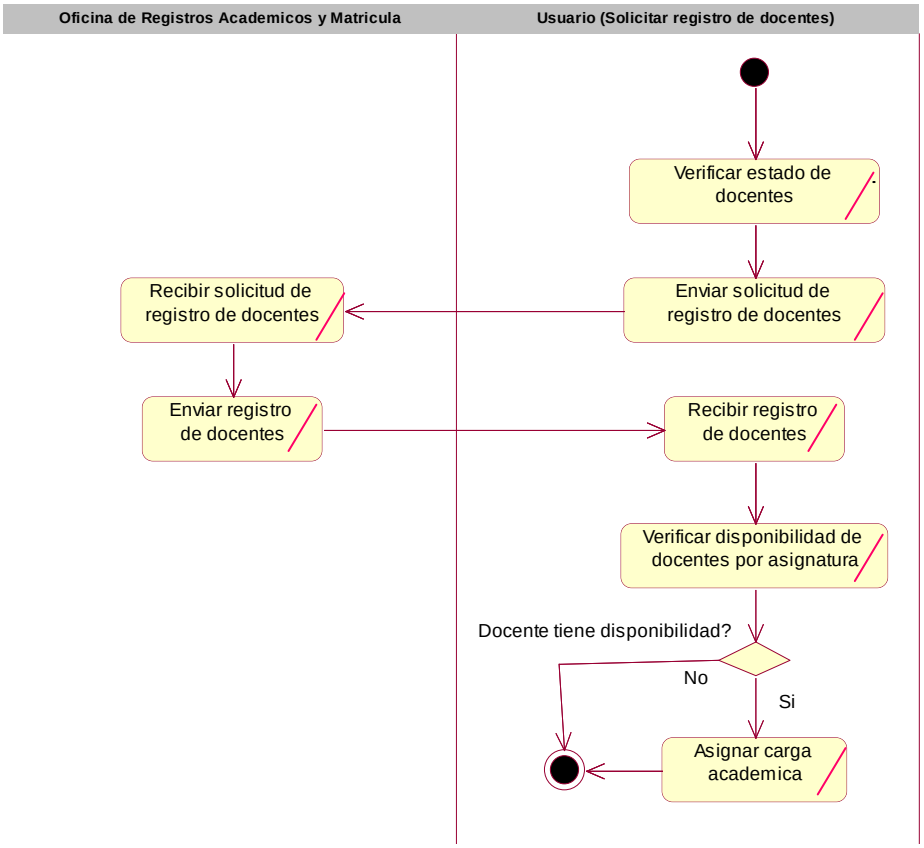
a. Realizar Matricula de Alumnos



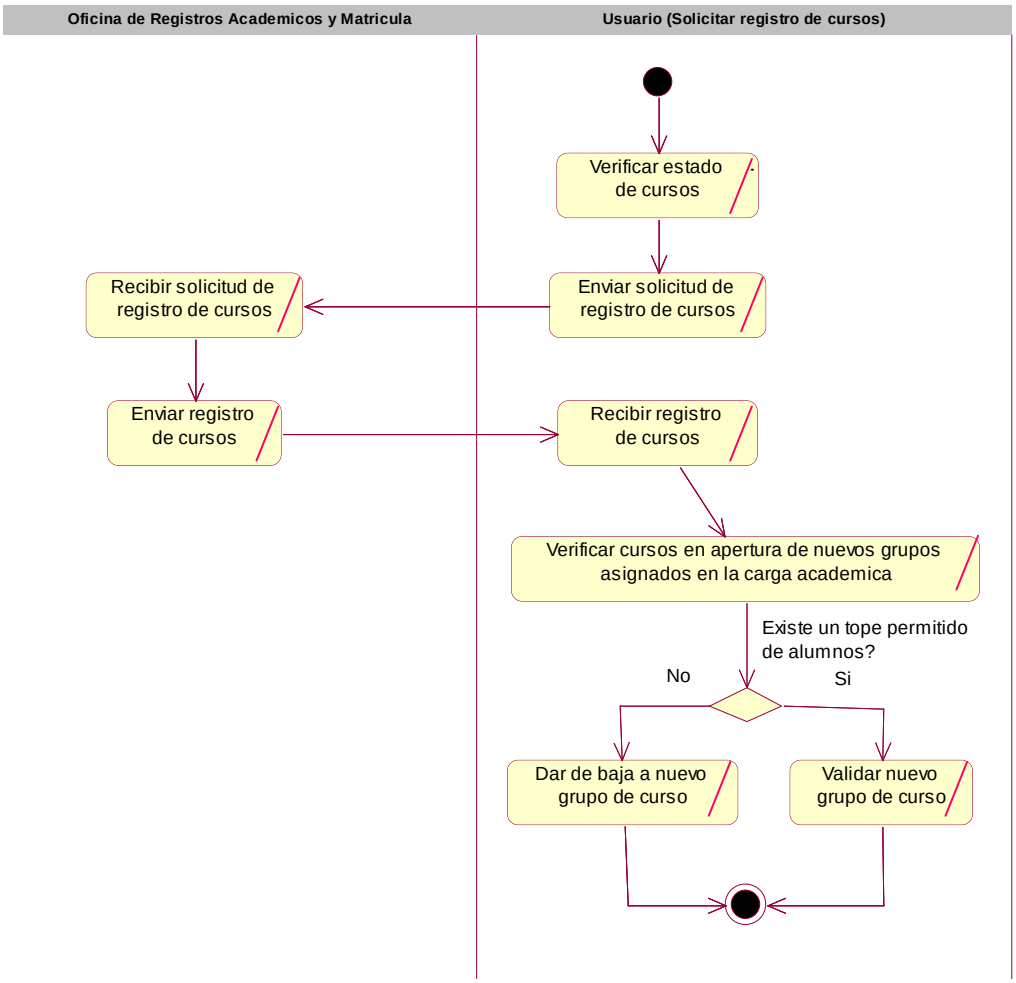
b. Solicitar Registro de alumnos



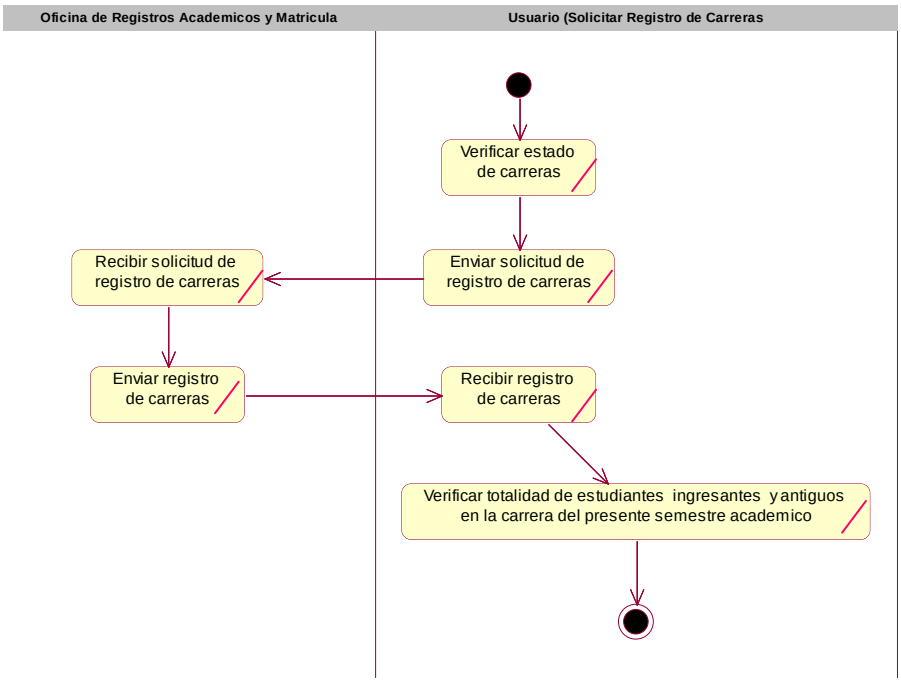
c. Solicitar Registro de Docentes



d. Solicitar Registro de Cursos



e. Solicitar Registro de Carreras



d) Especificaciones de Casos de Uso del Negocio

a. Realizar Matricula de Alumnos

Este caso de uso, permite al usuario autorizado poder realizar los registros de matrícula de los alumnos.

b. Solicitar registro de alumnos

Este caso de uso, permite al usuario autorizado poder solicitar los registros de los alumnos, para los procesos académicos administrativos.

c. Solicitar registro de docentes

Este caso de uso, permite al usuario autorizado poder solicitar los registros de los docentes, para los procesos académicos administrativos.

d. Solicitar registro de cursos

Este caso de uso, permite al usuario autorizado poder solicitar los registros de los cursos, para los procesos académicos administrativos.

e. Solicitar registro de carreras

Este caso de uso, permite al usuario autorizado poder solicitar los registros de las carreras, para los procesos académicos administrativos.

f. Abastecer registros de datos académicos

Este caso de uso, permite al proveedor administrativo (Oficina de Registros Académicos y Matricula) poder abastecer de información de datos (alumnos, docentes, cursos y carreras) a las entidades académicas.

Modelo de Análisis

1. Visión del Sistema

1.1. Posicionamiento

1.1.1. Problema

El problema de	De la existencia de un Sistema Heredado, para administrar procesos académicos (alumnos, docentes, cursos, carreras, matricula alumnos).
Que afecta	Al personal administrativo, encargado de los procesos académicos.
y su impacto	Necesidad de migrar y reutilizar las funcionalidades de la aplicación existente (alumnos, docentes, cursos, carreras, matricula alumnos), hacia nuevos sistemas tecnológicos.
Una buena solución debería	Utilizar la tecnología de *Web Services*, para reutilizar las funcionalidades del Sistema Heredado (alumnos, docentes, cursos, carreras, matricula alumnos), integrando aplicaciones y adaptándolos, para ser expuestos en nuevos sistemas.

1.1.2. Posicionamiento del Producto

Para	Para el personal administrativo, encargado de los procesos académicos en las instituciones.
Que necesitan	Brindar un servicio de consulta de información (alumnos, docentes, cursos, carreras, matricula alumnos), para gestiones administrativas realizadas.
El (nombre del producto)	Servicios *Web* para Sistemas Académicos Administrativos.
Que	Permitirá por medio de estándares de *Internet* poder realizar consultas de servicios (alumnos, docentes, cursos, carreras, matricula alumnos).
A diferencia	De la aplicación cliente/servidor actual (Sistema Heredado).
Nuestro Producto	Permitirá adaptar las funcionalidades existentes del Sistema Heredado (alumnos, docentes, cursos, carreras, matricula alumnos), para poder ser invocadas y reutilizadas en el servicio de nuevos sistemas.

1.1.3. Descripciones de los Roles

1.1.3.1. Resumen

Nombre	Descripción	Responsabilidades
Director	Encargado de administrar la Escuela Académico Profesional al cual fue asignado.	Encargado de aprobar los trámites y gestiones administrativos de los estudiantes, así como también para los docentes.
Administrativo	Encargado de la atención de los estudiantes y docentes, como también formar parte de los procesos académicos que se realicen.	Asegurar que el sistema cumpla con las necesidades y las expectativas requeridas.

1.1.4. Producto

1.1.4.1. Necesidades y Características

Necesidad	Prioridad	Característica	Versión del Sistema
Director: Información actualizada sobre los procesos administrativos (alumnos, docentes, cursos, carreras, matricula alumnos).	Alta	El servicio de *Web Services* proporcionará una consulta de información (alumnos, docentes, cursos, carreras, matricula alumnos).	1
Administrativo: Una reducción significativa del tiempo de acceso a las consultas realizadas.	Alta	El servicio de *Web Services* permitirá de manera ágil acceder a los servicios de consulta de infamación (alumnos, docentes, cursos, carreras, matricula alumnos).	1

1.1.5. Otros requerimientos del producto

1. Interfaz de Usuario

Todas las interfaces, deberán ser expuestas en estándares de *Internet*, para los servicios proporcionados por el *Web Services*.

2. Confiabilidad

El sistema deberá de manejarse a través de la Internet de manera integrada.

3. De rendimiento

El tiempo de consulta de información de los servicios del *Web Services* para las gestiones administrativas, no deberá exceder los 5 minutos.

4. Licencia e instalación

Esto se realizara de acuerdo a las políticas de la institución.

2. Especificación de Requerimientos de Software

1. Requerimientos Funcionales

- Sistemas Operativos: Windows XP.
- Base de datos en SQL Server 2000.
- La aplicación de Sistema Heredado será en Visual Basic.NET
- La aplicación de *Web Services* será en Visual Studio.NET

2. Requerimientos No Funcionales

- **Funcionalidad**

  El sistema permitirá reutilizar las funcionalidades existentes del Sistema Heredado (alumnos, docentes, cursos, carreras, matricula alumnos), para ser invocadas utilizando los servicios del *Web Services*, y estos poder ser expuestos a nuevos sistemas.

- **Usabilidad**

  El sistema tendrá la capacidad de ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando sea utilizado en condiciones específicas.

- **Fiabilidad**

  El sistema tendrá la capacidad de poder mantener un nivel especificado de desempeño cuando sea requerido.

- **Performance**

  El sistema deberá suministrar un desempeño apropiado en relación a los recursos utilizados, con un tiempo de respuesta no mayor a 5 minutos y una base de datos que soporte un máximo de 30GB, con los módulos suficientes requeridos para las PCs de tipo clientes y 2 servidores como mínimo.

- **Soportabilidad**

  El sistema tendrá la capacidad de poder ser transferido de un ambiente (organizacional, hardware, software) a otro, y poder tener las siguientes características:

  - **Facilidad para adaptarlo:** El sistema trabajara bajo el sistema operativo Windows XP.

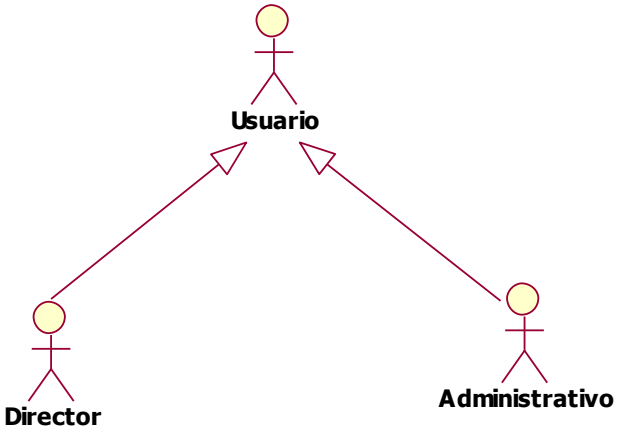
  - **Facilidad para instalarlo:** En el módulo del cliente se instalara el *.NET Framework*, para la aplicación, en el servidor de aplicaciones *Web*, se tendrá que instalar el *Visual Studio.NET*, y para el servidor de base de datos, se tendrá instalado el SQL Server 2000. Las computadoras necesarias para este sistema tendrán las siguientes características:

    - Pentium IV, 1700 MHz, 512 MB RAM, Disco duro 40.

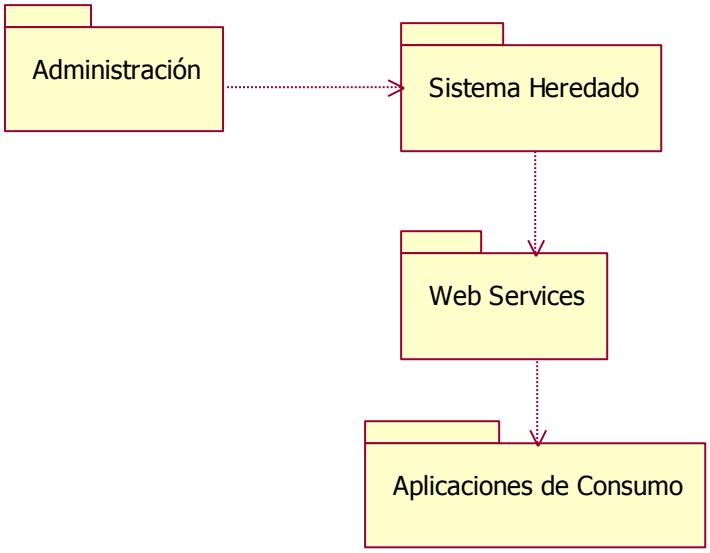
  - **Capacidad de reemplazar:** Podrá funcionar bajo versiones mas actualizadas del software requerido.

3. Vista de Casos de uso del Sistema

a. Relaciones entre actores

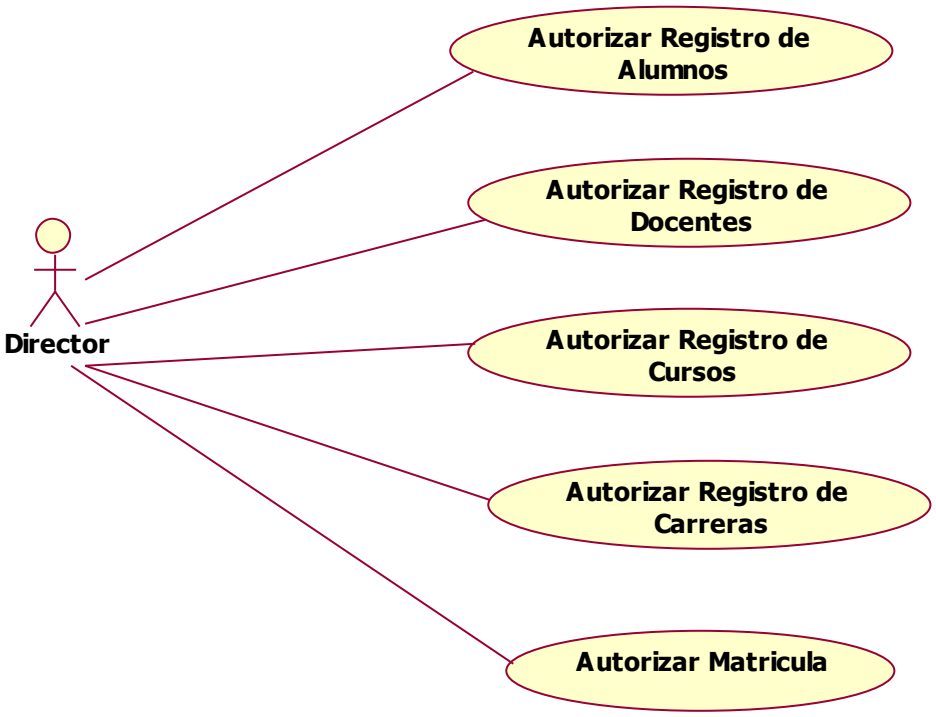


b. Diagrama de paquetes

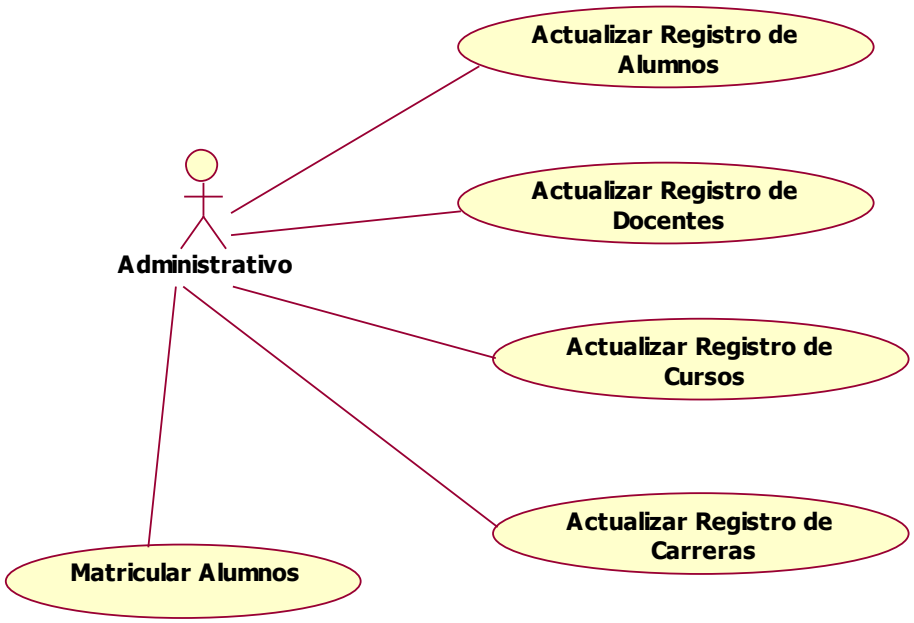


c. Vista de Casos de Uso por Paquete

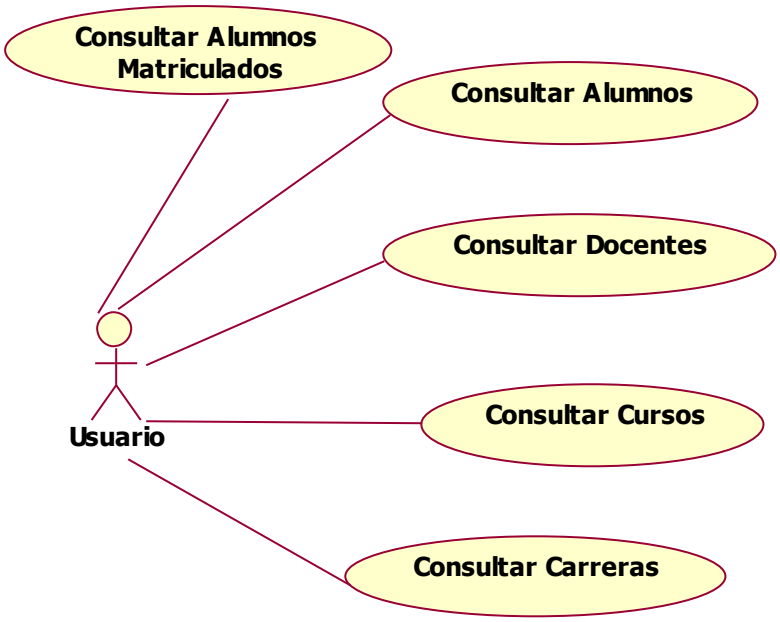
1. Módulo de Administración



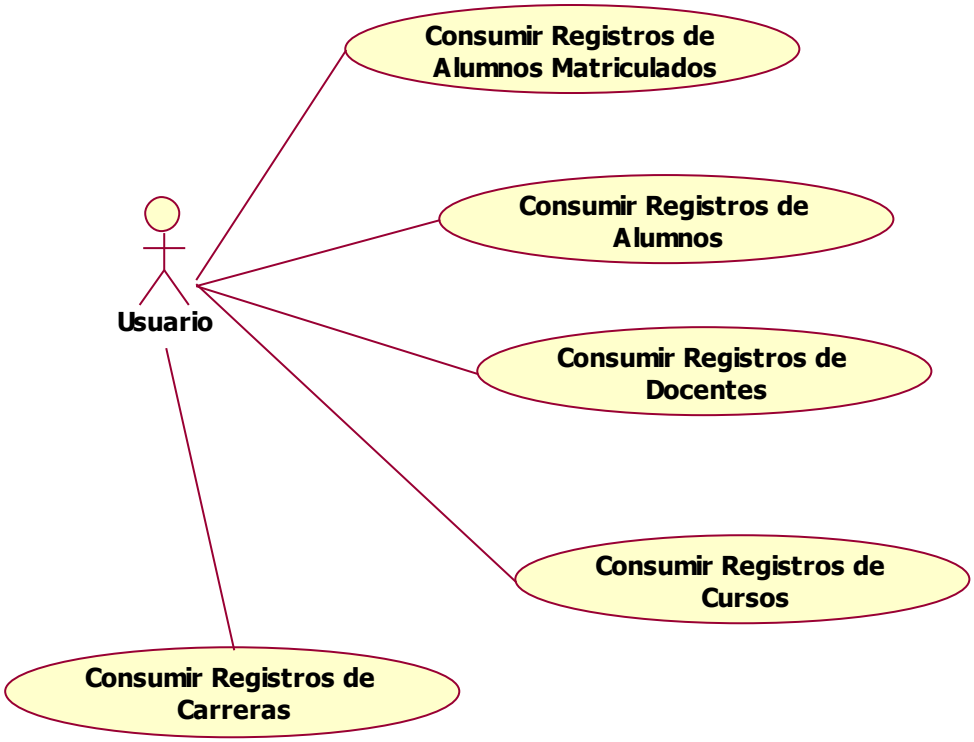
2. Modulo del Sistema Heredado



3. Modulo del Web Services



4. Módulo de Aplicaciones de Consumo



4. Especificaciones de Requerimientos de de Casos de Uso del Sistema

1. Módulo de Administración

a. Autorizar Matricula

El caso de uso es iniciado por el Director, con el objetivo de activar en el sistema el proceso de apertura de matrícula de alumnos.

Escenarios

a. Abrir matricula

b. Cerrar matricula

b. Autorizar Registro de Alumnos

El caso de uso es iniciado por el Director, con el objetivo de activar en el sistema el proceso de inscripción de un alumno.

Escenarios

a. Abrir registro

b. Cerrar registro

c. Autorizar Registro de Docentes

El caso de uso es iniciado por el Director, con el objetivo de activar en el sistema el proceso de inscripción de un docente.

Escenarios

a. Abrir registro

b. Cerrar registro

d. Autorizar Registro de Cursos

El caso de uso es iniciado por el Director, con el objetivo de activar en el sistema el proceso de inscripción de un curso.

Escenarios

a. Abrir registro

b. Cerrar registro

e. Autorizar Registro de Carreras

El caso de uso es iniciado por el Director, con el objetivo de activar en el sistema el proceso de inscripción de una carrera.

Escenarios

a. Abrir registro

b. Cerrar registro

2. Modulo del Sistema Heredado

a. Matricular Alumnos

El caso de uso es iniciado por un docente Administrativo, con el objetivo de proceder a los registro de matrícula de alumnos en el sistema.

Escenarios

a. Realizar procesos de matrícula de alumnos

b. Cerrar proceso de matrícula de alumnos

b. Actualizar Registro de Alumnos

El caso de uso es iniciado por un docente Administrativo, con el objetivo de mantener actualizado el registro de alumnos en el sistema.

Escenarios

a. Adicionar un alumno

b. Modificar datos de un alumno

c. Dar de baja a un alumno

c. Actualizar Registro de Docentes

El caso de uso es iniciado por un docente Administrativo, con el objetivo de mantener actualizado el registro de docentes en el sistema.

Escenarios

a. Adicionar un alumno

b. Modificar datos de un alumno

c. Dar de baja a un alumno

d. Actualizar Registro de Cursos

El caso de uso es iniciado por un docente Administrativo, con el objetivo de mantener actualizado el registro de cursos en el sistema.

Escenarios

a. Adicionar un curso

b. Modificar datos de un curso

c. Dar de baja a un curso

e. Actualizar Registro de Carreras

El caso de uso es iniciado por un docente Administrativo, con el objetivo de mantener actualizado el registro de carreras en el sistema.

Escenarios

a. Adicionar una carrera

b. Modificar datos de una carrera

c. Dar de baja a una carrera

3. Módulo del Web Services

a. Consultar Alumnos Matriculados

El caso de uso es iniciado por el Usuario, con el objetivo de invocar los servicios del *Web Services*, para la consulta de alumnos matriculados y ser mostrados en un estándar XML.

Escenarios

a. Consultar Invocar Alumnos Matriculados

b. Consultar Alumnos

El caso de uso es iniciado por el Usuario, con el objetivo de invocar los servicios del *Web Services*, para la consulta de alumnos y ser mostrados en un estándar XML.

Escenarios

a. Consultar Alumnos

c. Consultar Docentes

El caso de uso es iniciado por el Usuario, con el objetivo de invocar los servicios del *Web Services*, para la consulta de docentes y ser mostrados en un estándar XML.

Escenarios

a. Consultar Docentes

d. Consultar Cursos

El caso de uso es iniciado por el Usuario, con el objetivo de invocar los servicios del *Web Services*, para la consulta de cursos y ser mostrados en un estándar XML.

Escenarios

a. Consultar Cursos

e. Consultar Carreras

El caso de uso es iniciado por el Usuario, con el objetivo de invocar los servicios del *Web Services*, para la consulta de Carreras y ser mostrados en un estándar XML.

Escenarios

a. Consultar Carreras

4. Módulo de Aplicaciones de Consumo

a. Consumir Registros de Alumnos Matriculados

El caso de uso es iniciado por el Usuario, con el objetivo de consultar los servicios del *Web Services*, para obtener la información de los alumnos matriculados.

Escenarios

a. Consumir Registros de Alumnos Matriculados

b. Consumir Registros de Alumnos

El caso de uso es iniciado por el Usuario, con el objetivo de consultar los servicios del *Web Services*, para obtener la información de los alumnos.

Escenarios

a. Consumir Registros de Alumnos

c. Consumir Registros de Docentes

El caso de uso es iniciado por el Usuario, con el objetivo de consultar los servicios del *Web Services*, para obtener la información de los docentes.

Escenarios

a. Consumir Registros de Docentes

d. Consumir Registros de Cursos

El caso de uso es iniciado por el Usuario, con el objetivo de consultar los servicios del *Web Services*, para obtener la información de los cursos.

Escenarios

a. Consumir Registros de Cursos

e. Consumir Registros de Carreras

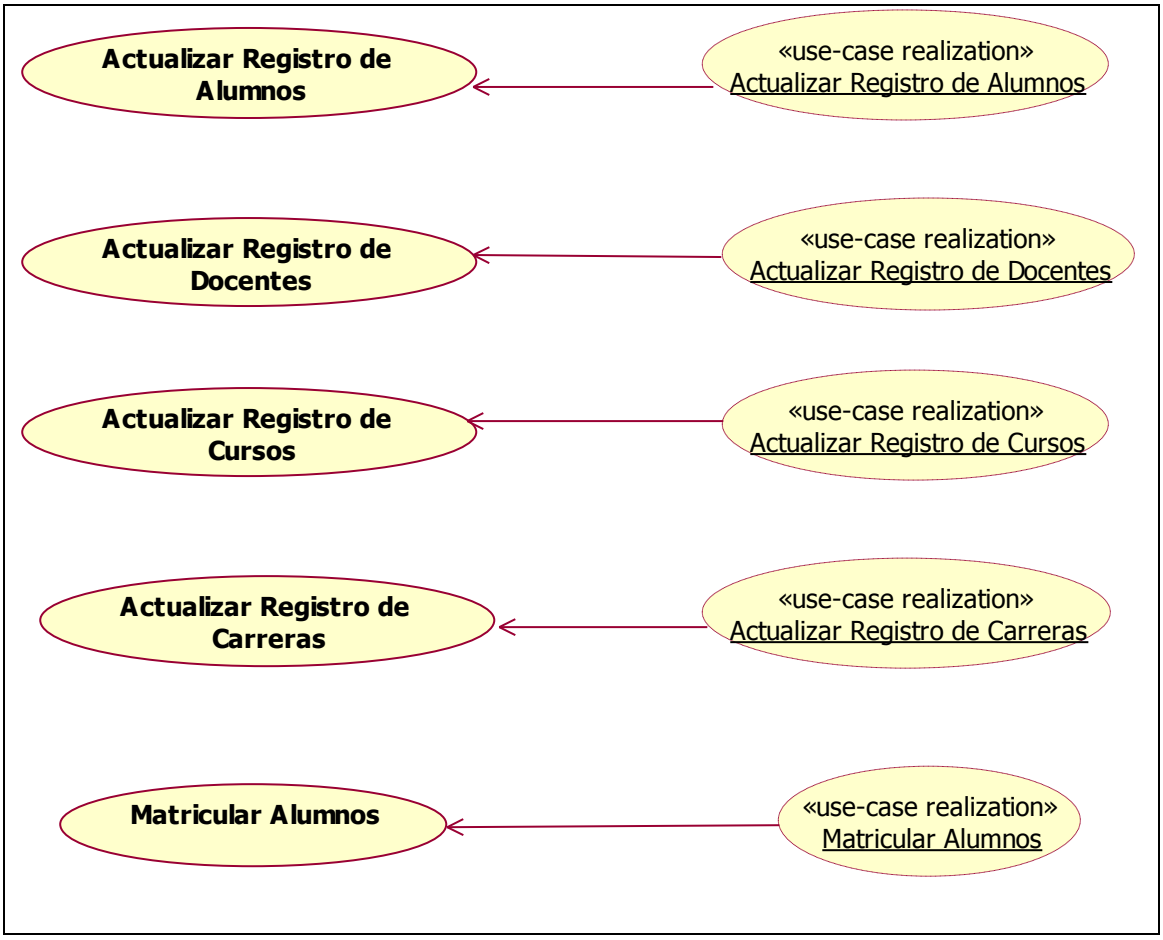
El caso de uso es iniciado por el Usuario, con el objetivo de consultar los servicios del *Web Services*, para obtener la información de las carreras.

Escenarios

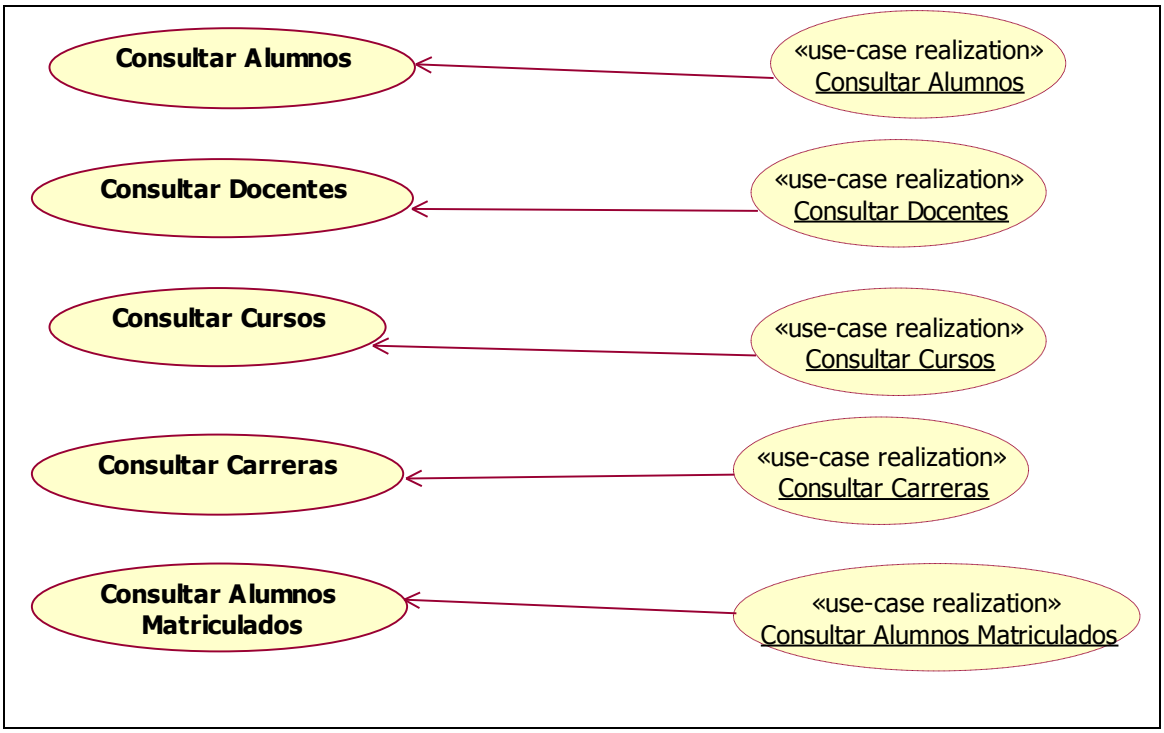
a. Consumir Registros de Carreras

5. Diagramas de Realización de Casos de Uso

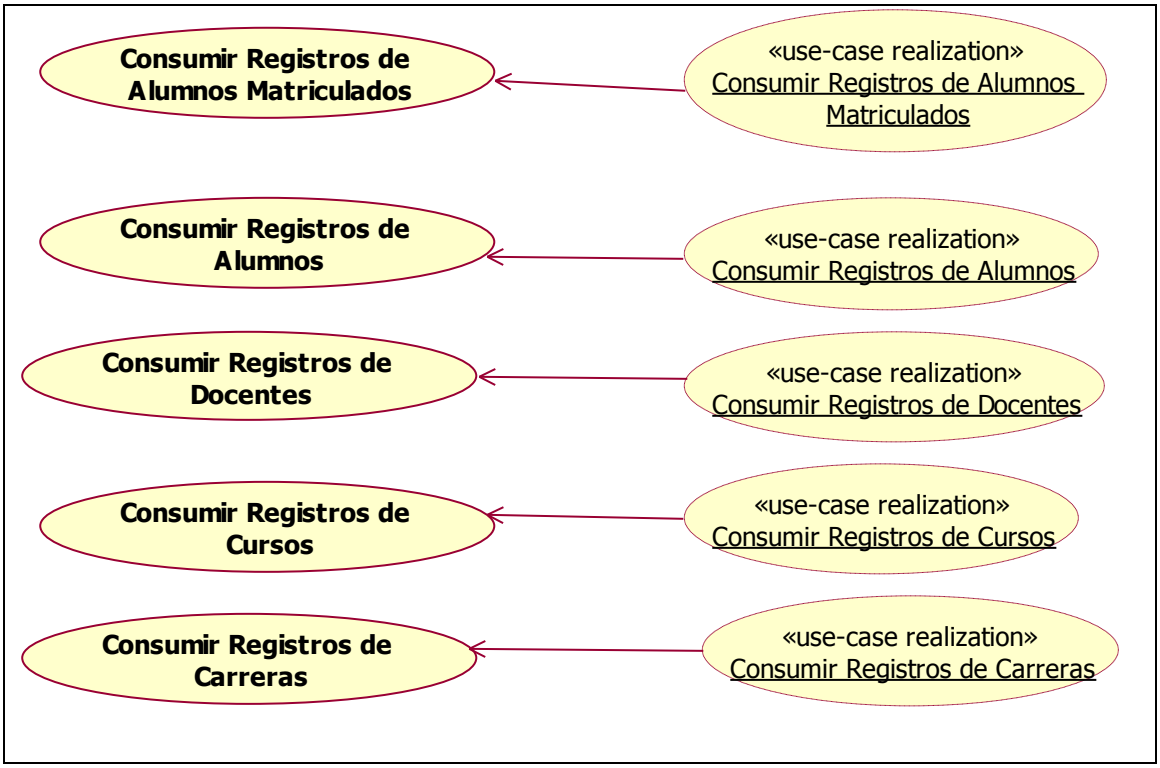
1. Módulo de Sistema Heredado



2. Modulo del Web Services

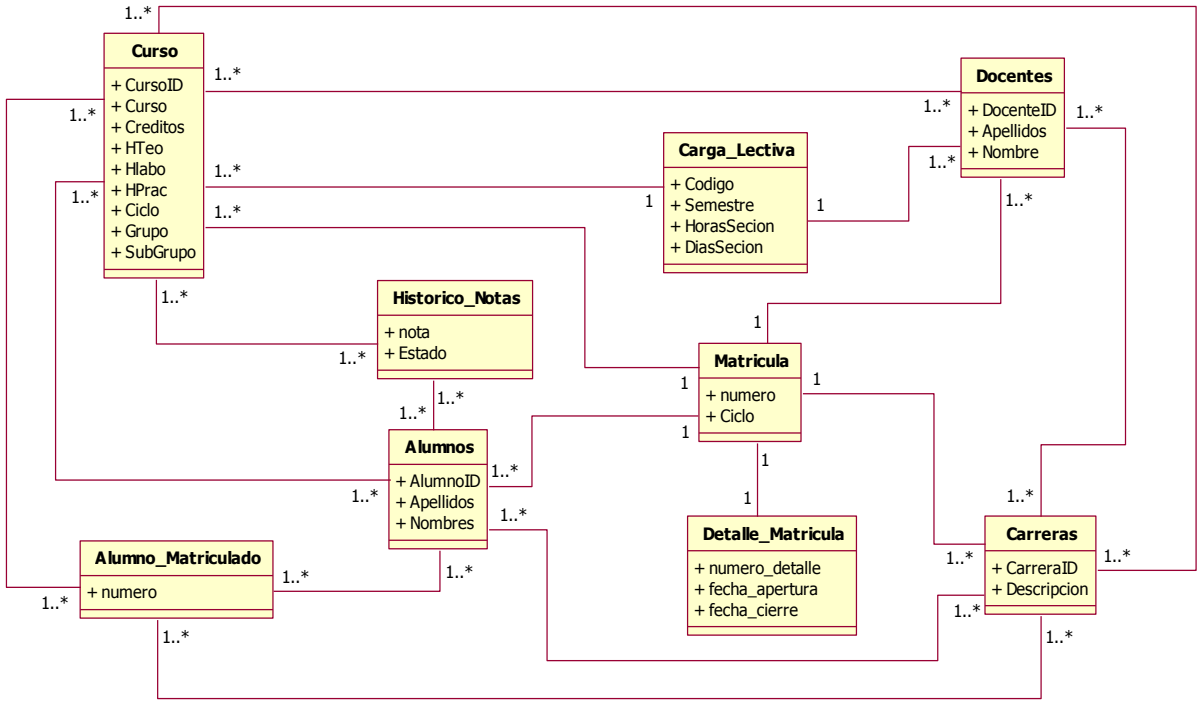


3. Módulo de Aplicaciones de Consumo

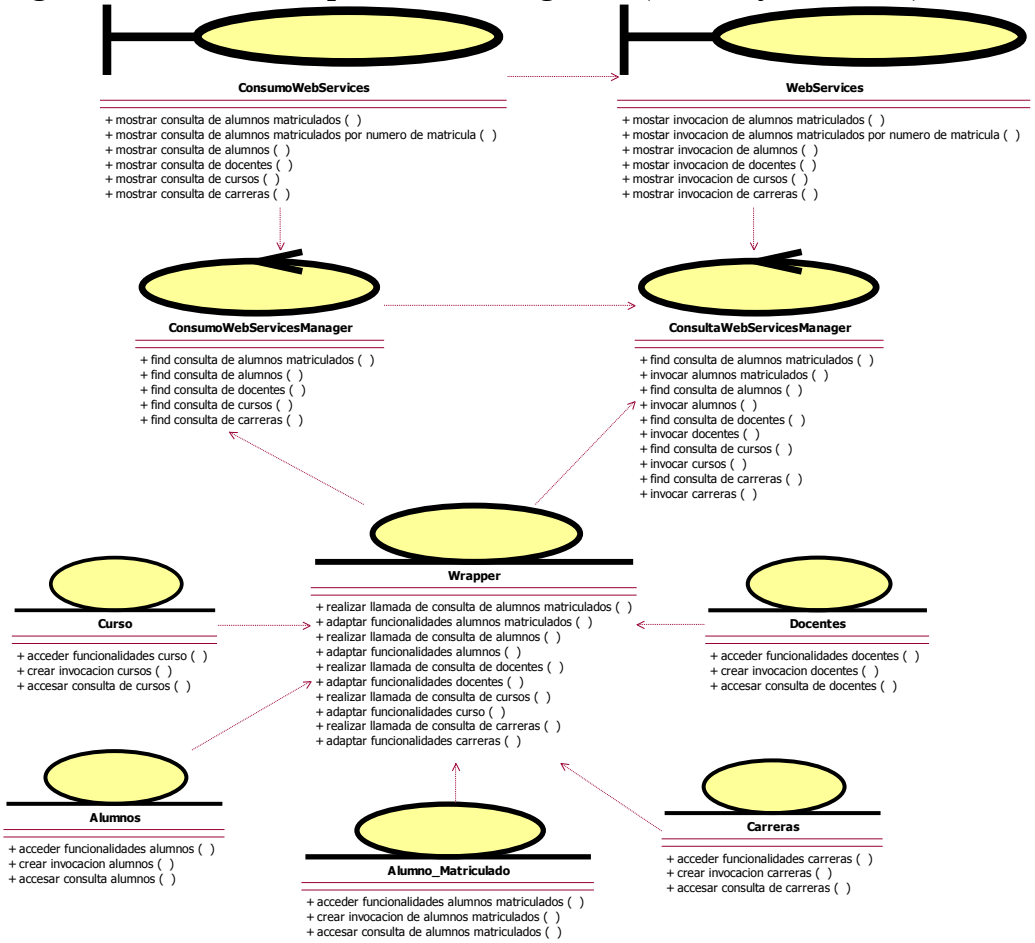


6. Diagramas de Clases

a. Diagrama de Clases (Modelo Lógico)



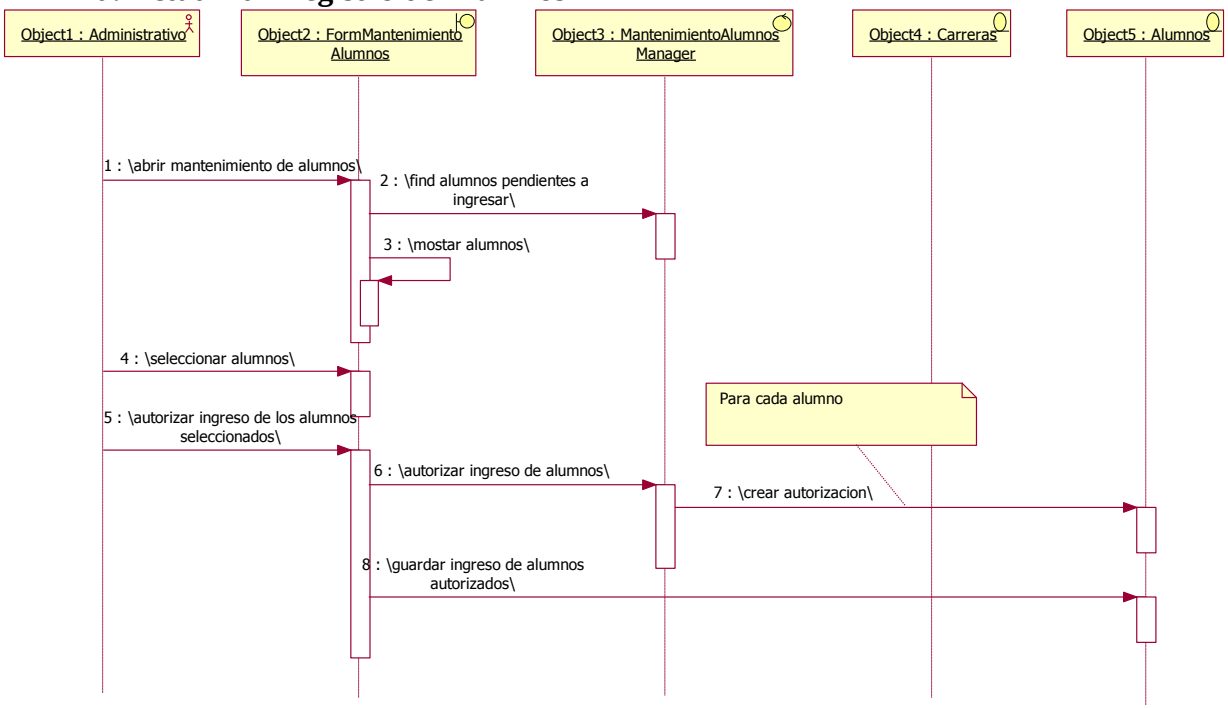
b. Diagrama de Clases de Aplicaciones Integradas (Prueba y Consumo).



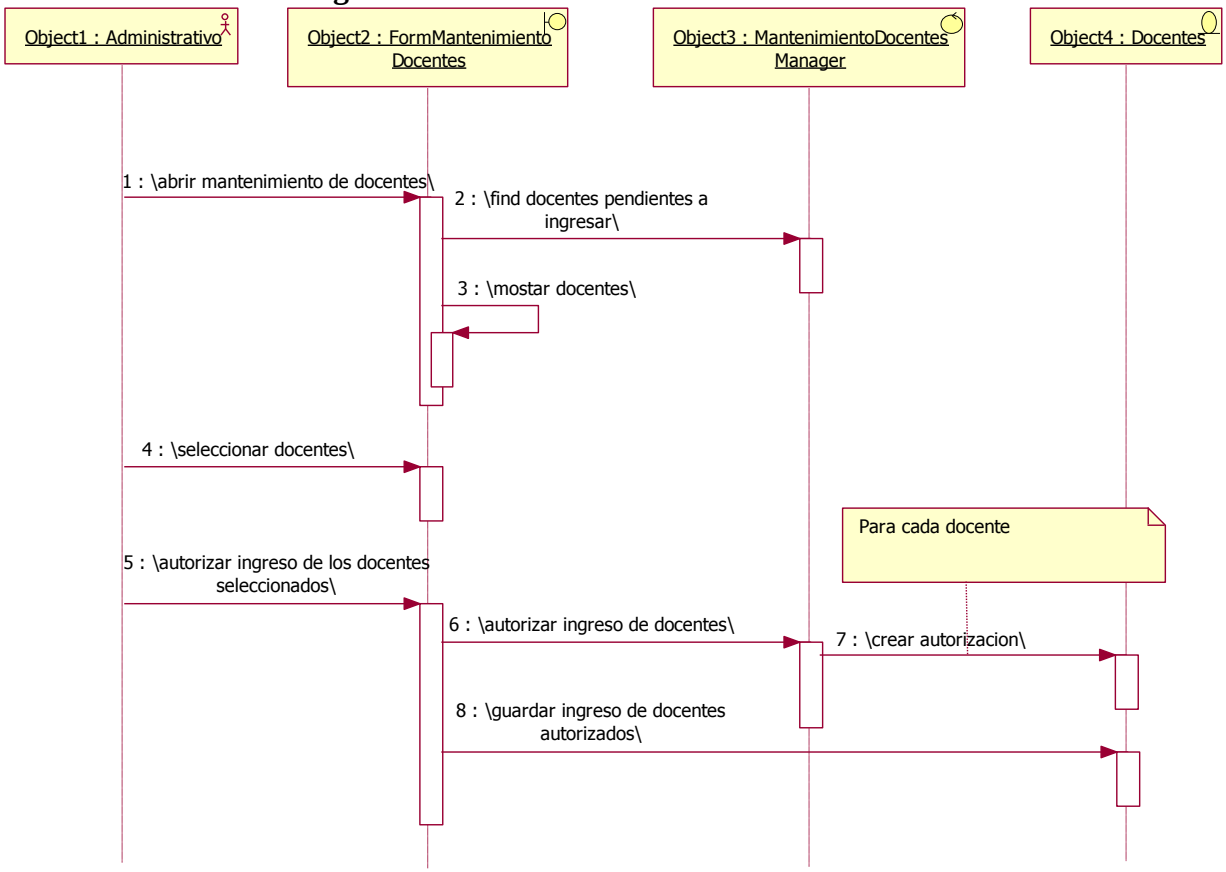
7. Diagramas de Secuencia

1. Módulo de Sistema Heredado

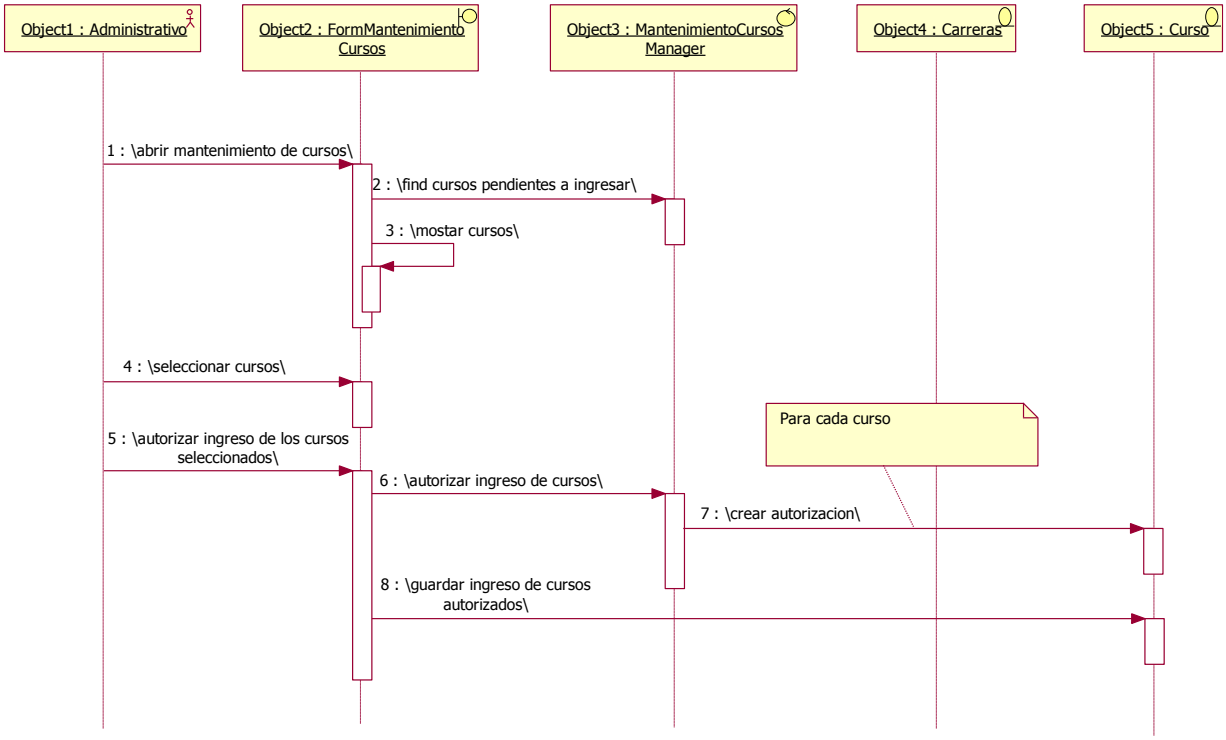
a. Actualizar Registro de Alumnos



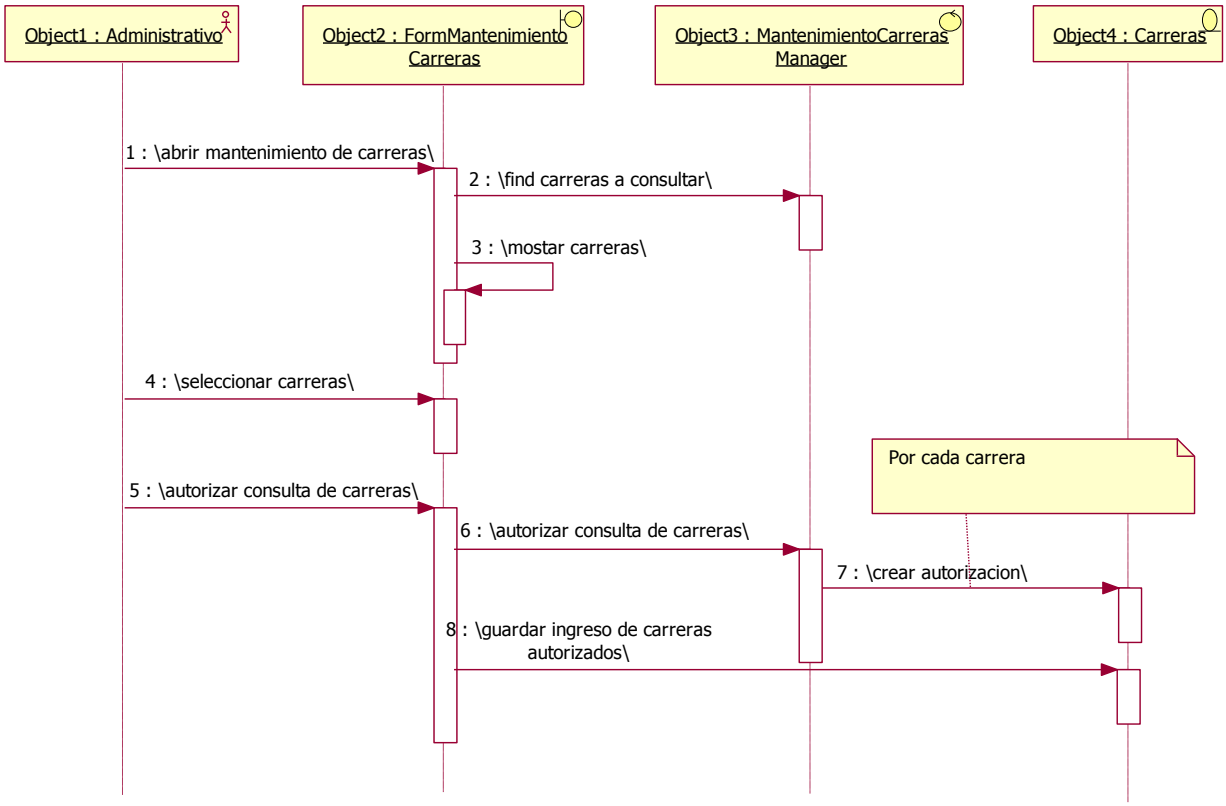
b. Actualizar Registro de Docentes



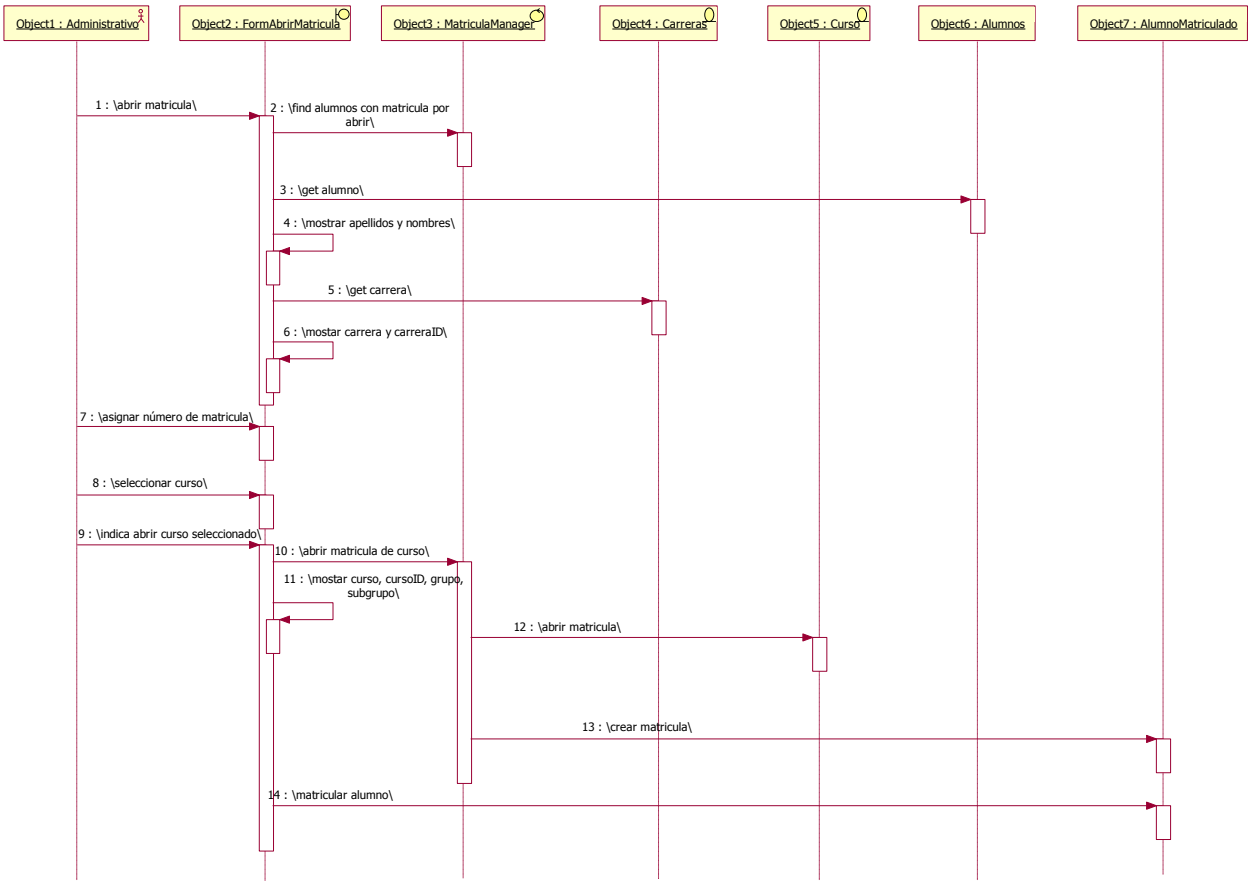
c. Actualizar Registro de Cursos



d. Actualizar Registro de Carreras

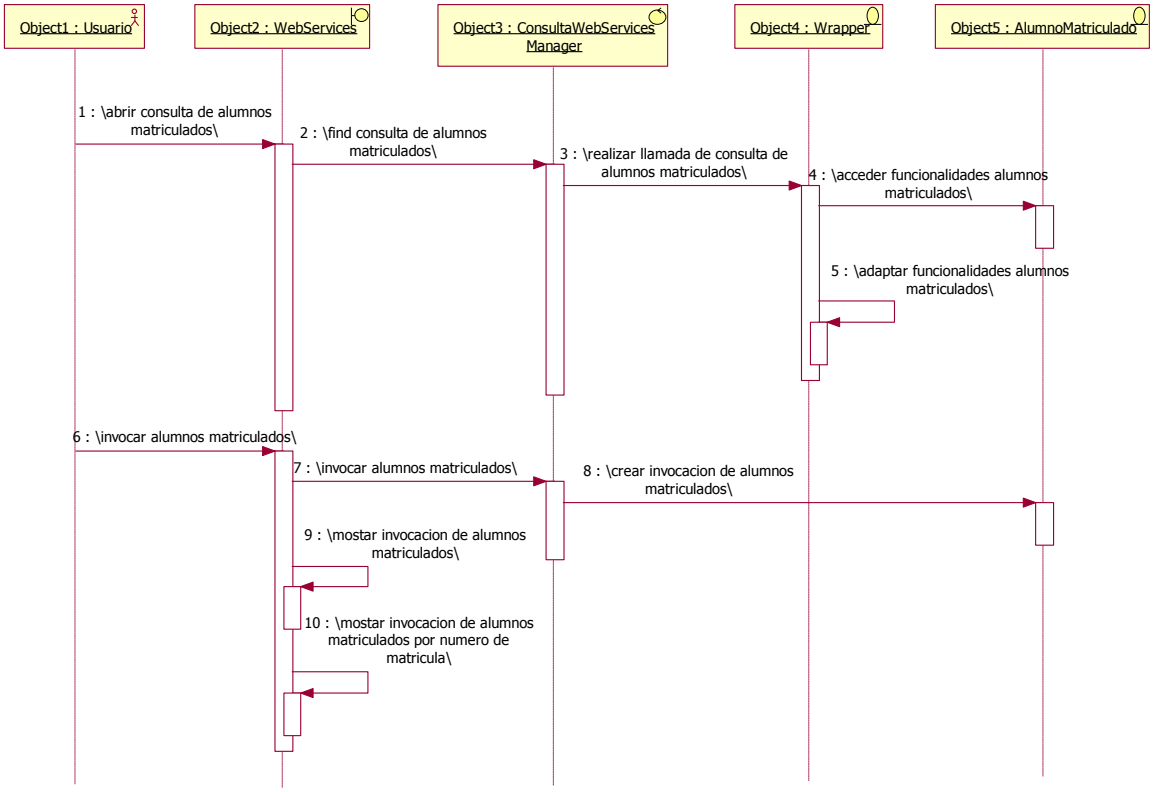


e. Matricular Alumnos

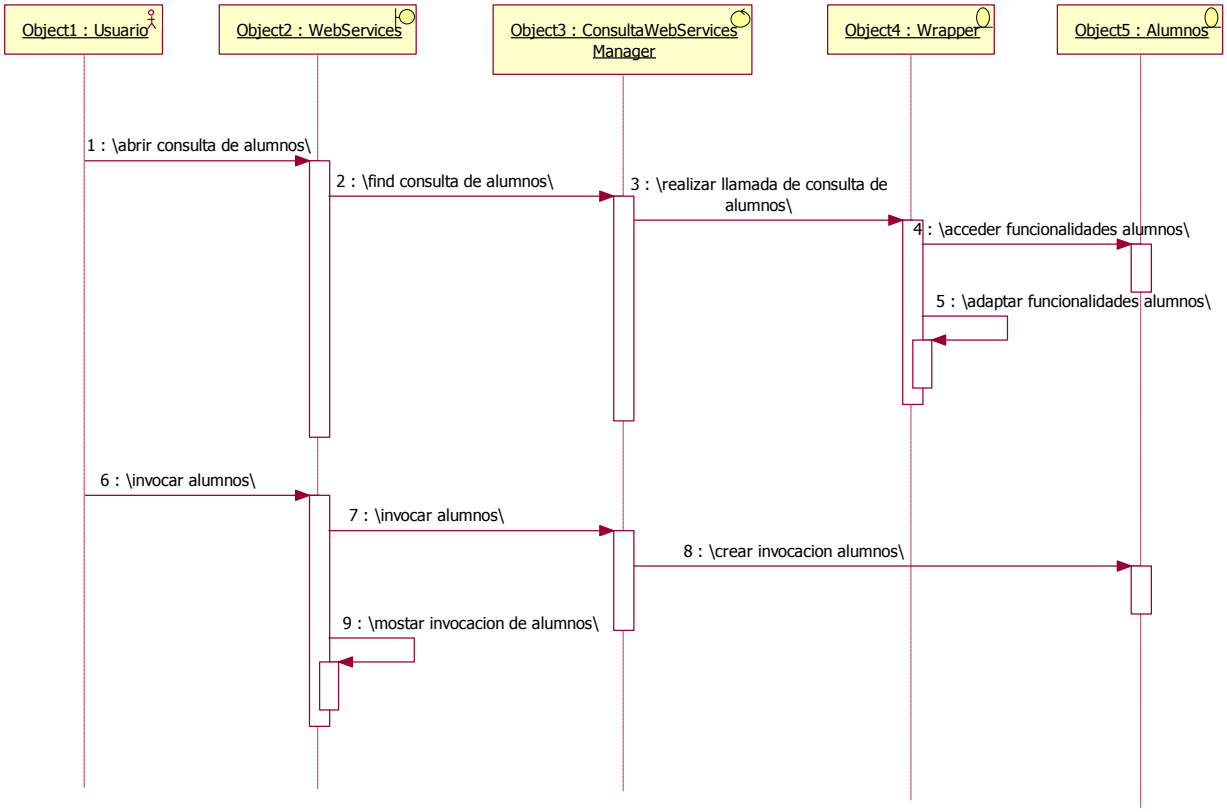


2. Modulo del Web Services

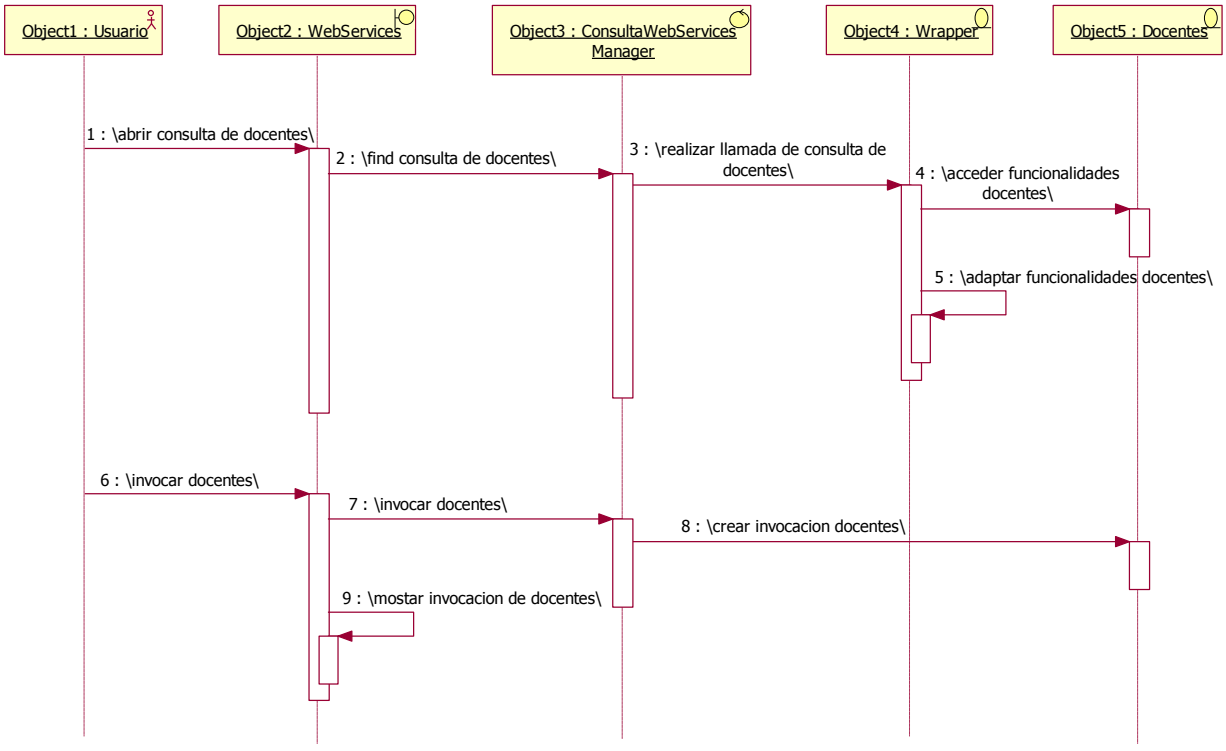
a. Consultar Alumnos Matriculados



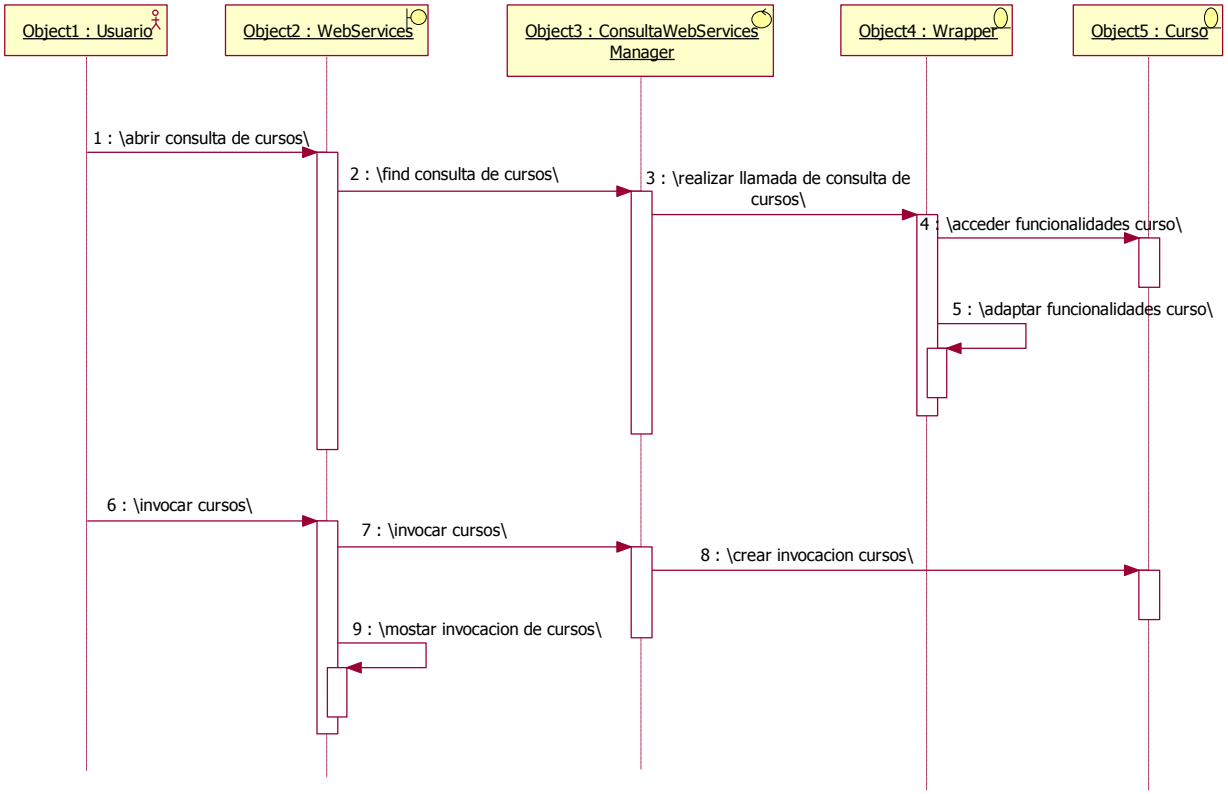
b. Consultar Alumnos



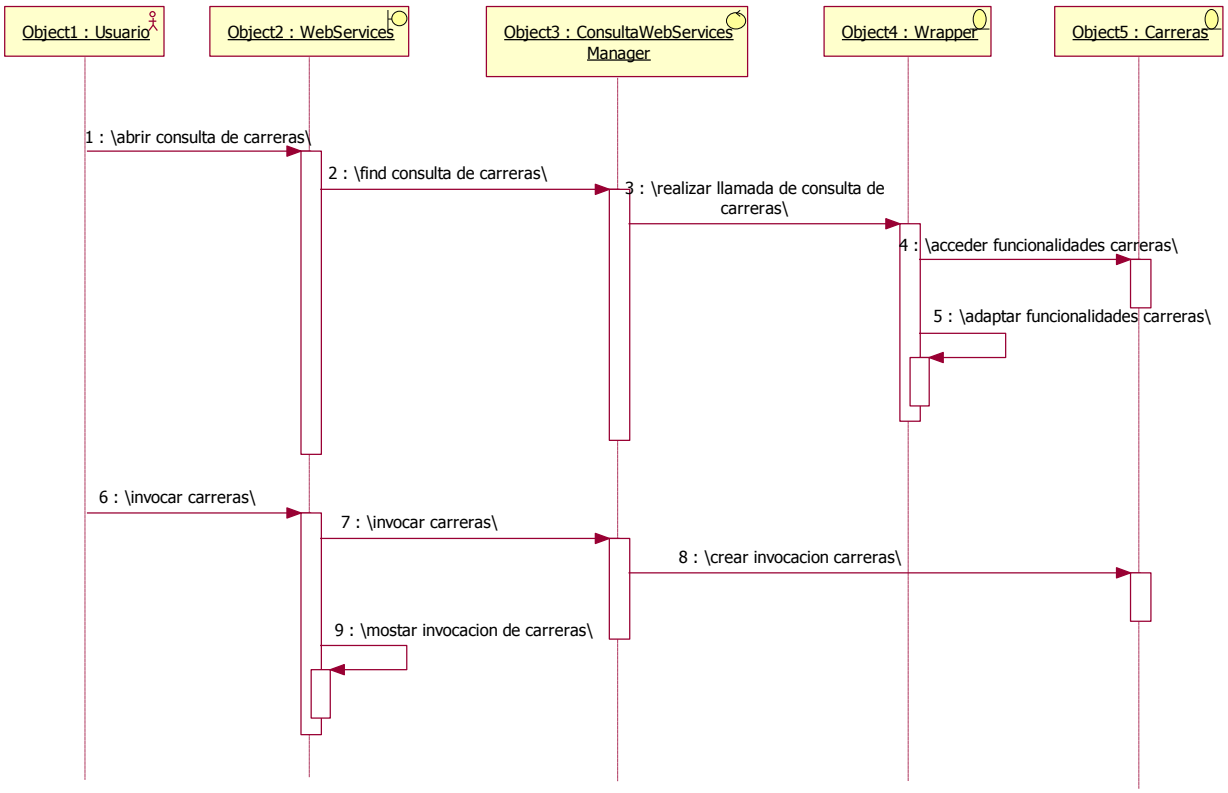
c. Consultar Docentes



d. Consultar Cursos

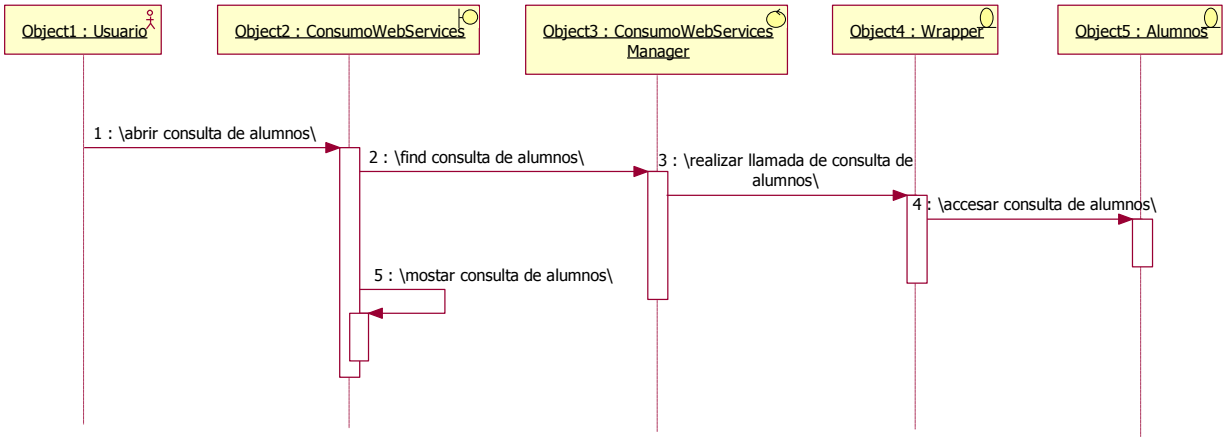


e. Consultar Carreras

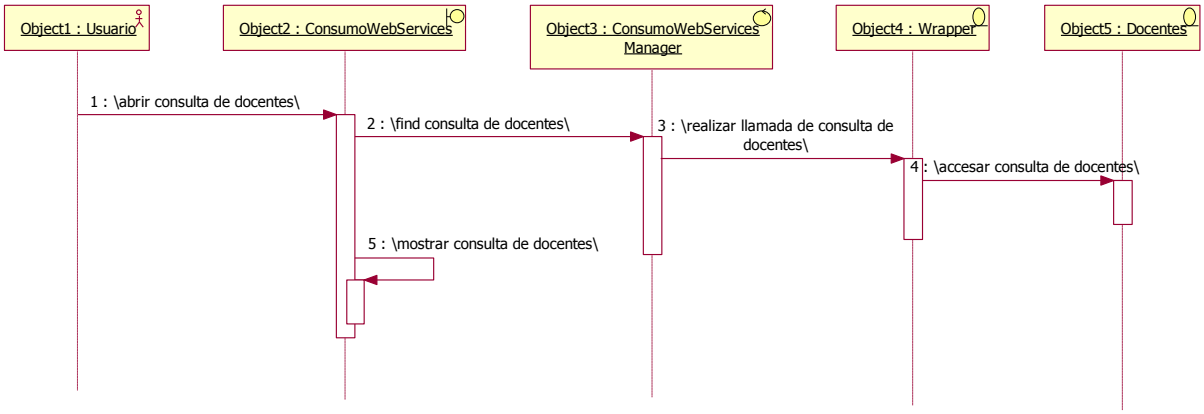


3. Módulo de Aplicaciones de Consumo

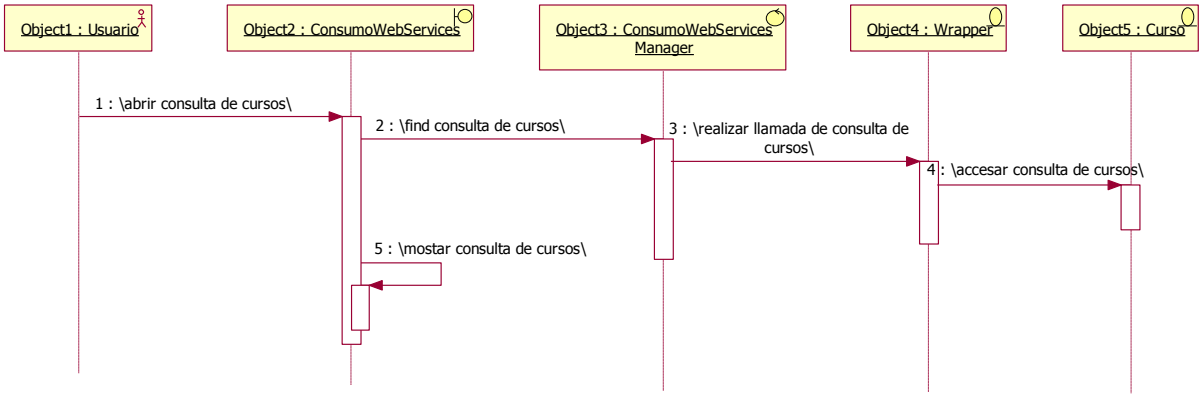
a. Consumir Registros de Alumnos



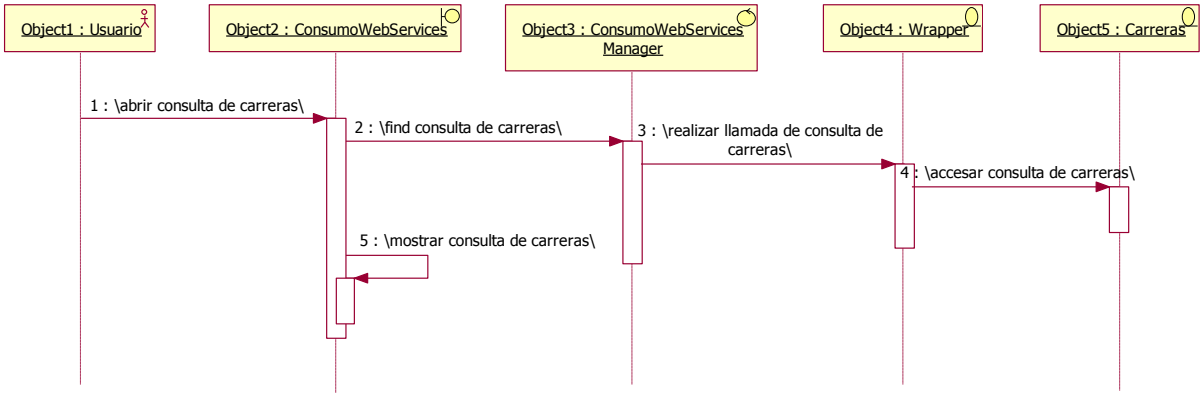
b. Consumir Registros de Docentes



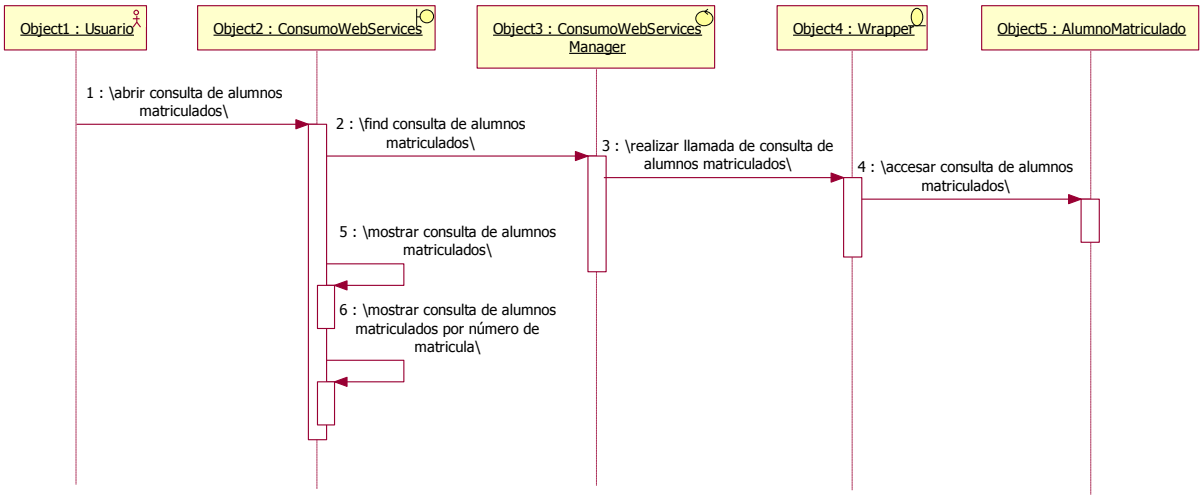
c. Consumir Registros de Cursos



d. Consumir Registros de Carreras



e. Consumir Registros de Alumnos Matriculados



Modelo de Diseño

1. Arquitectura de Software

1.1 Vista Lógica

1.1.1. Vista General

a) Diagrama de Paquetes entre Capas

Capa de Presentación

En esta capa se ubican todos los componentes de la interfaz de usuario: form.

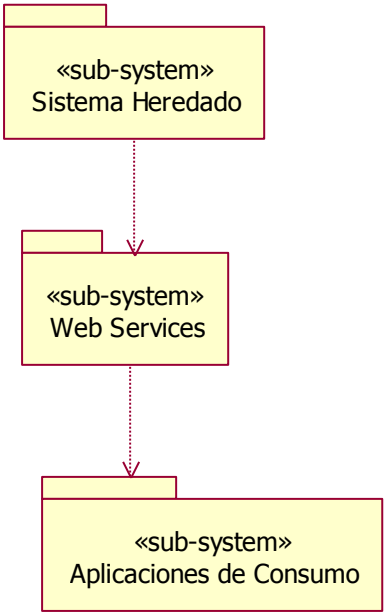
Capa de Negocio

Toda la lógica de negocio se ubica en esta capa.

Capa de Persistencia

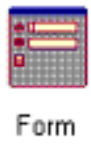
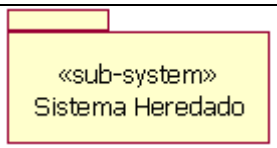
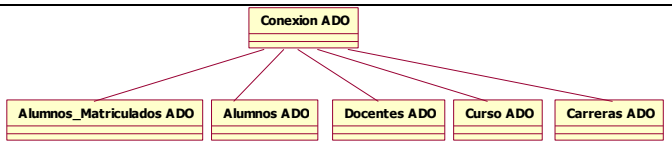
Esta capa contiene todas las clases encargadas de las operaciones con la base de datos.

b) Diagrama de Paquetes entre Subsistemas



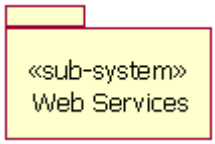
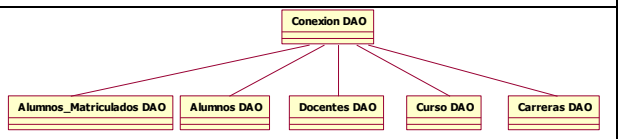
c) Mecanismos de Diseño de Tres Capas (Capa de Persistencia)

1. Sistema Heredado

Presentación	**Lógica del Negocio**	**Persistencia**
Interfaces de Usuario del Sistema Heredado	Lógica del Negocio del Sistema Heredado	Conexión de Datos del Sistema Heredado
		

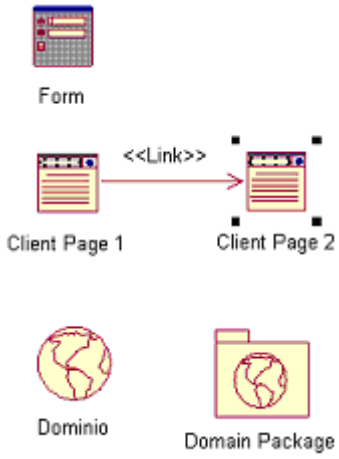
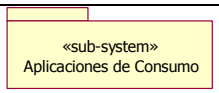
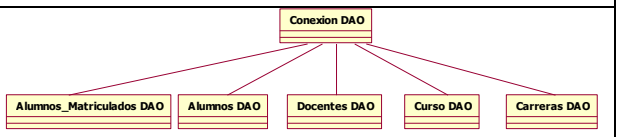
Persistencia: En Sistema Heredado, se va a manejar la persistencia utilizando los patrones **ADO (*Objetos de Datos ActiveX*),** para cada una de las entidades del negocio identificadas, que se comunicaran a un ADO general, el cual se encarga de hacer la conexión directa a la base de datos.

2. Web Services

Presentación	**Lógica del Negocio**	**Persistencia**
Interfaces de Usuario del *Web Services*	Lógica del Negocio del *Web Services*	Conexión de Datos del Web Services
		

Persistencia: En el servicio de *Web Services*, se va a manejar la persistencia utilizando los patrones **DAO (*Data Access Object*),** para cada una de las entidades del negocio identificadas, que se comunicaran a un DAO general, el cual se encarga de hacer la conexión directa a la base de datos.

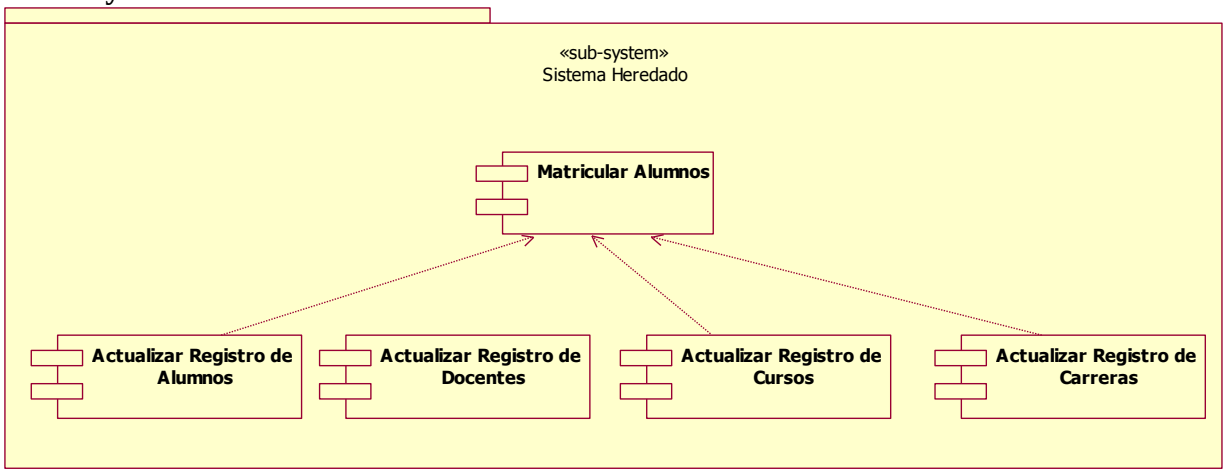
3. Aplicaciones de Consumo

Presentación	**Lógica del Negocio**	**Persistencia**
Interfaces de Usuario de las Aplicaciones de Consumo	Lógica del Negocio de Aplicaciones de Consumo	Conexión de Datos de las Aplicaciones de Consumo
		

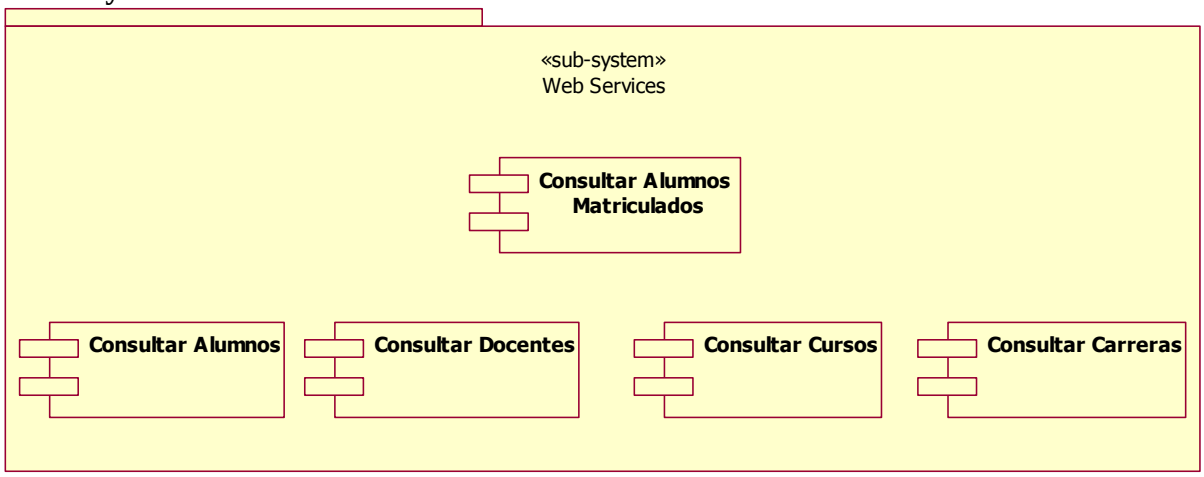
Persistencia: En el servicio de *Web Services*, se va a manejar la persistencia utilizando los patrones **DAO (*Data Access Object*),** para cada una de las entidades del negocio identificadas, que se comunicaran a un DAO general, el cual se encarga de hacer la conexión directa a la base de datos.

d) Diagrama de Paquetes por cada Subsistema (Capa de Lógica del Negocio)

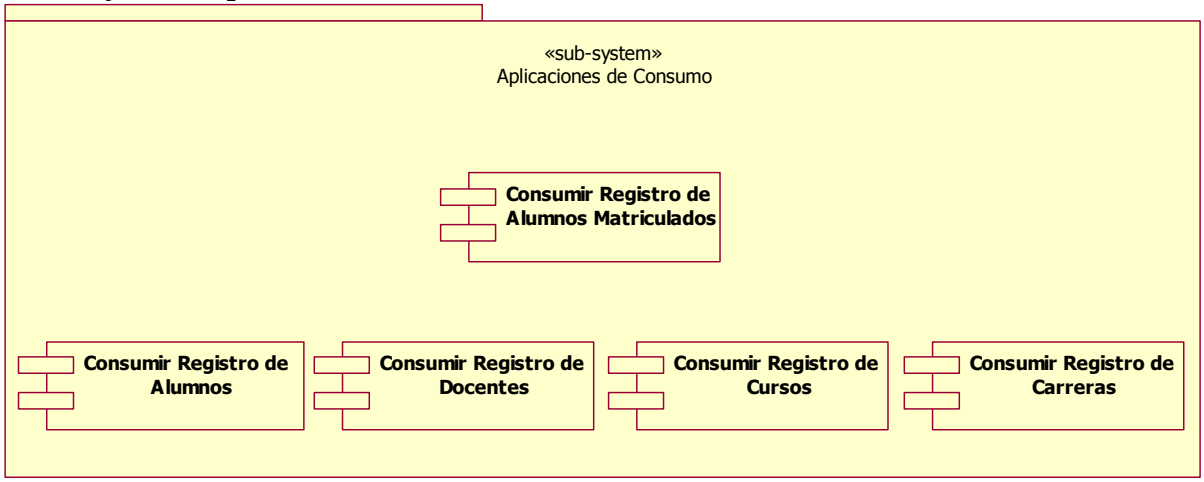
1. sub-system: Sistema Heredado



2. sub-system: Web Services



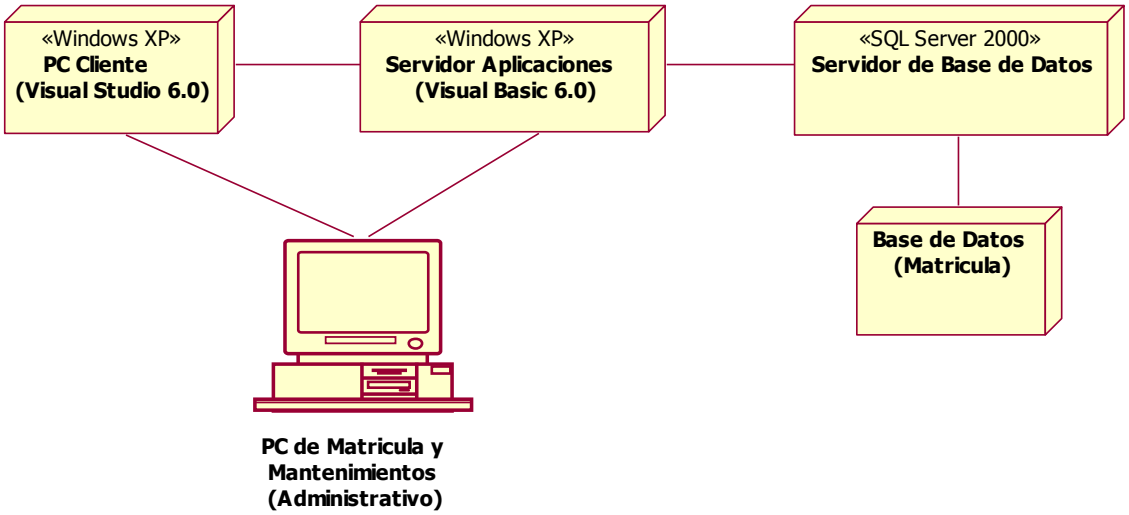
3. sub-system: Aplicaciones de Consumo



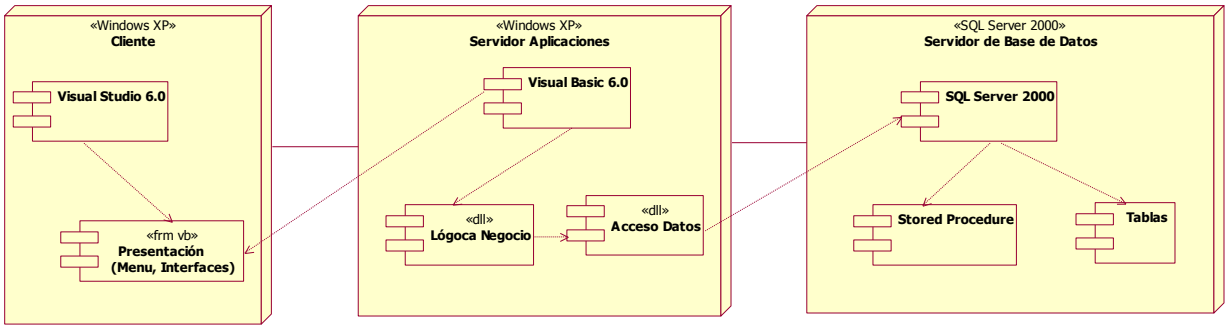
1.2. Vista de Despliegue

1. Sistema Heredado

a. Diagrama de Despliegue mostrando la Arquitectura de Hardware/Software del Sistema Heredado

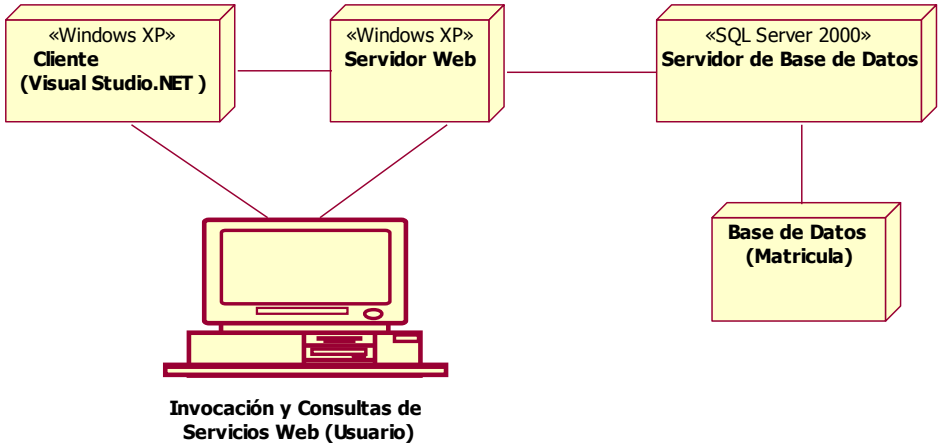


b. Diagrama de Despliegue del sistema, indicando la ubicación de los componentes de distribución del Sistema Heredado

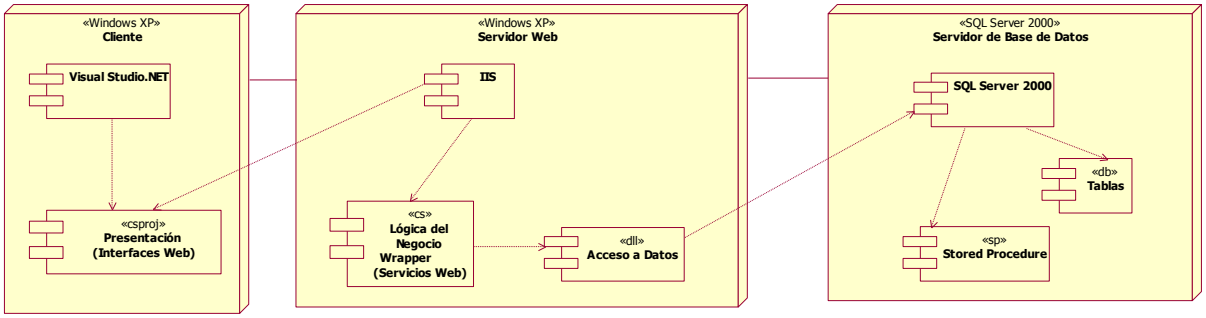


2. Web Services

a. Diagrama de Despliegue mostrando la Arquitectura de Hardware/Software del Web Services

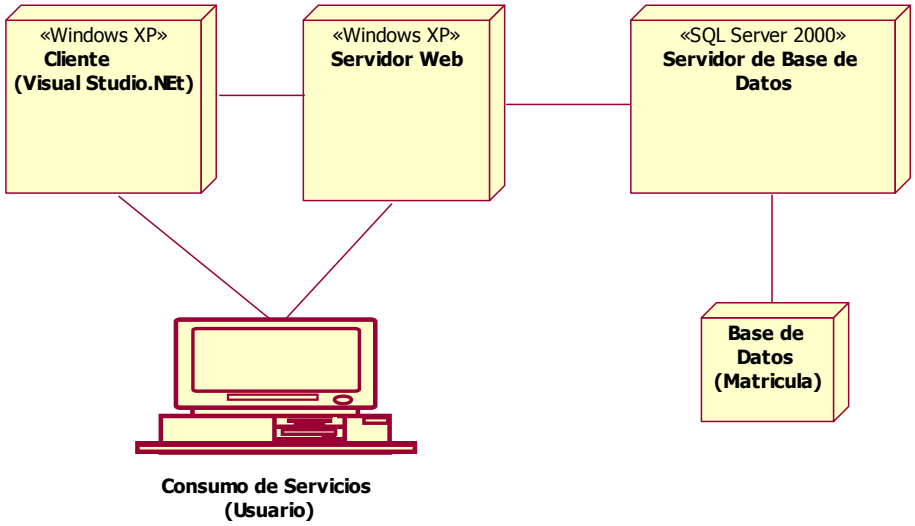


b. Diagrama de Despliegue del sistema, indicando la ubicación de los componentes de distribución del Web Services

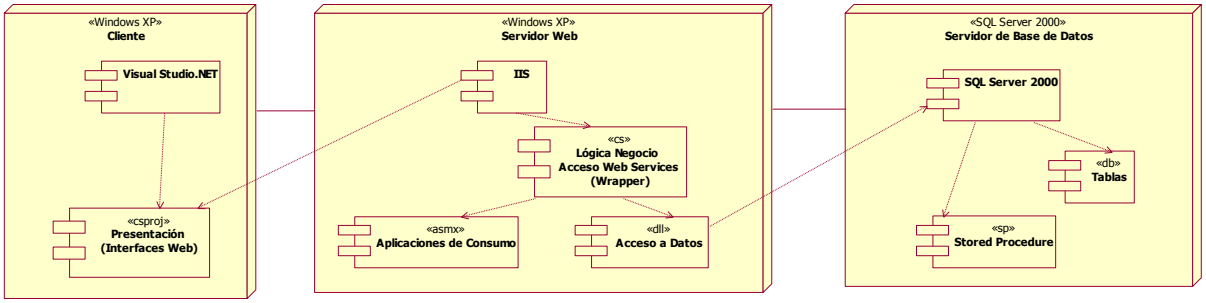


3. Aplicaciones de Consumo

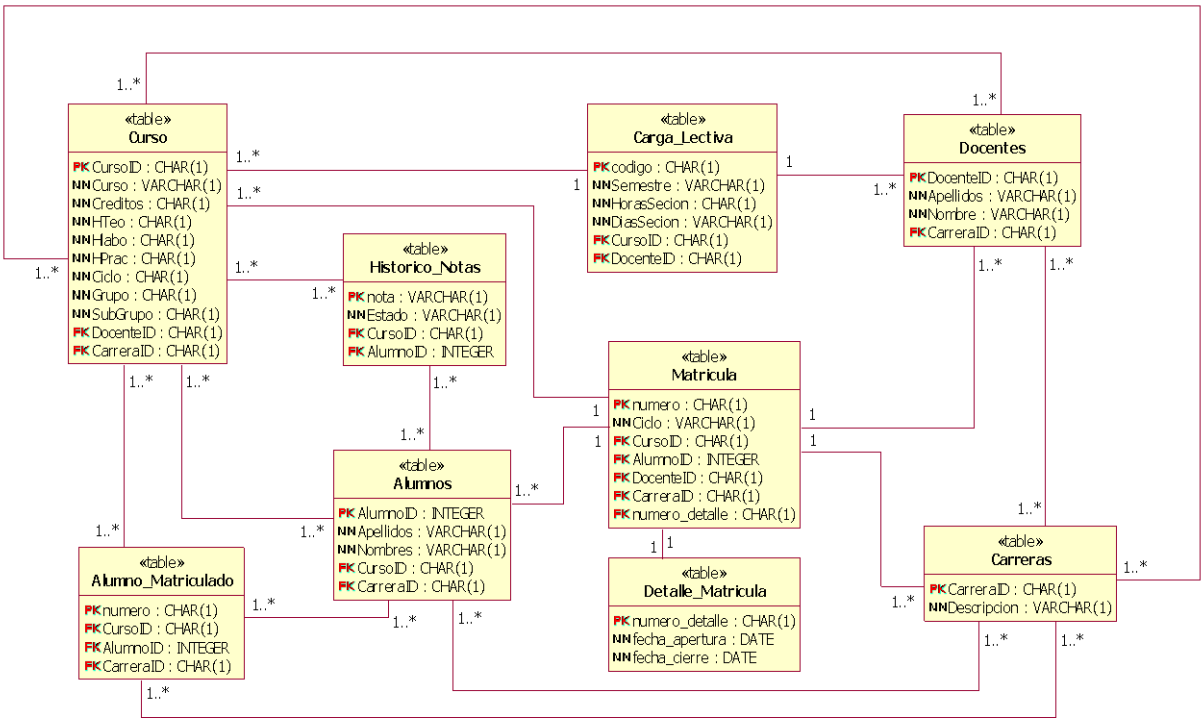
a) Diagrama de Despliegue mostrando la Arquitectura de Hardware/Software de la nueva Aplicación de Consumo



b) Diagrama de Despliegue del sistema, indicando la ubicación de los componentes de distribución de Aplicaciones de Consumo



1.3 Vista de Datos (Modelo Físico)



ANEXO 2

INTERFACES DEL SISTEMA HEREDADO

Interfaces del Sistema Heredado

A continuación, se muestran las interfaces que representan al Sistema Heredado, con sus respectivas funcionalidades (Alumnos, Docentes, Cursos, y Carreras, Matricula de Alumnos). Desarrollado en Visual Basic 6.0.

- **Menú de Acceso a las funcionalidades del Sistema Heredado**

En esta interfaz, se acceden a las funcionalidades del Sistema Heredado. (Ver figura 7.1).

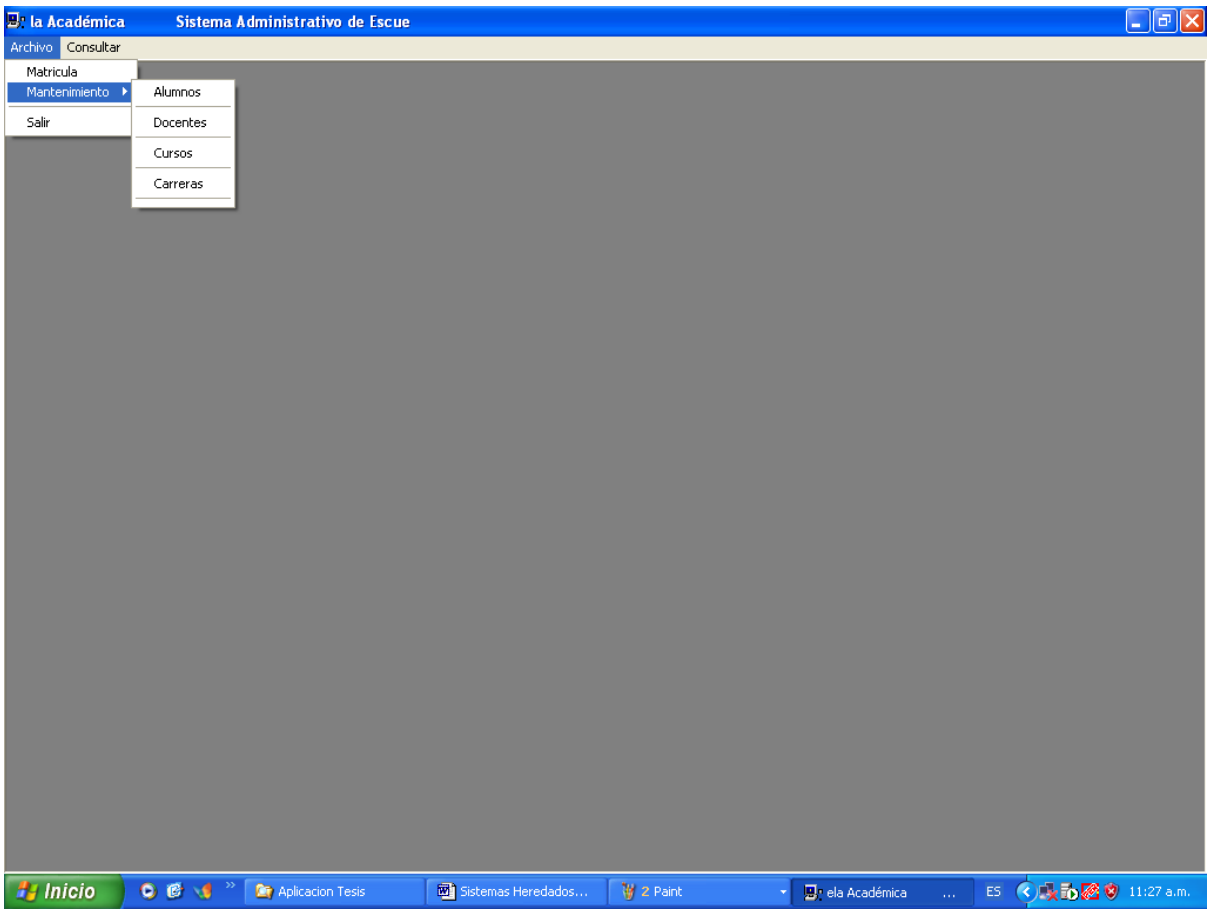


Figura 7.1. Sistema Heredado: Interfaz de menú de la aplicación

- **Interfaz de Matricula de Alumnos**

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto al registro de matrícula de alumnos. (Ver figura 7.2).

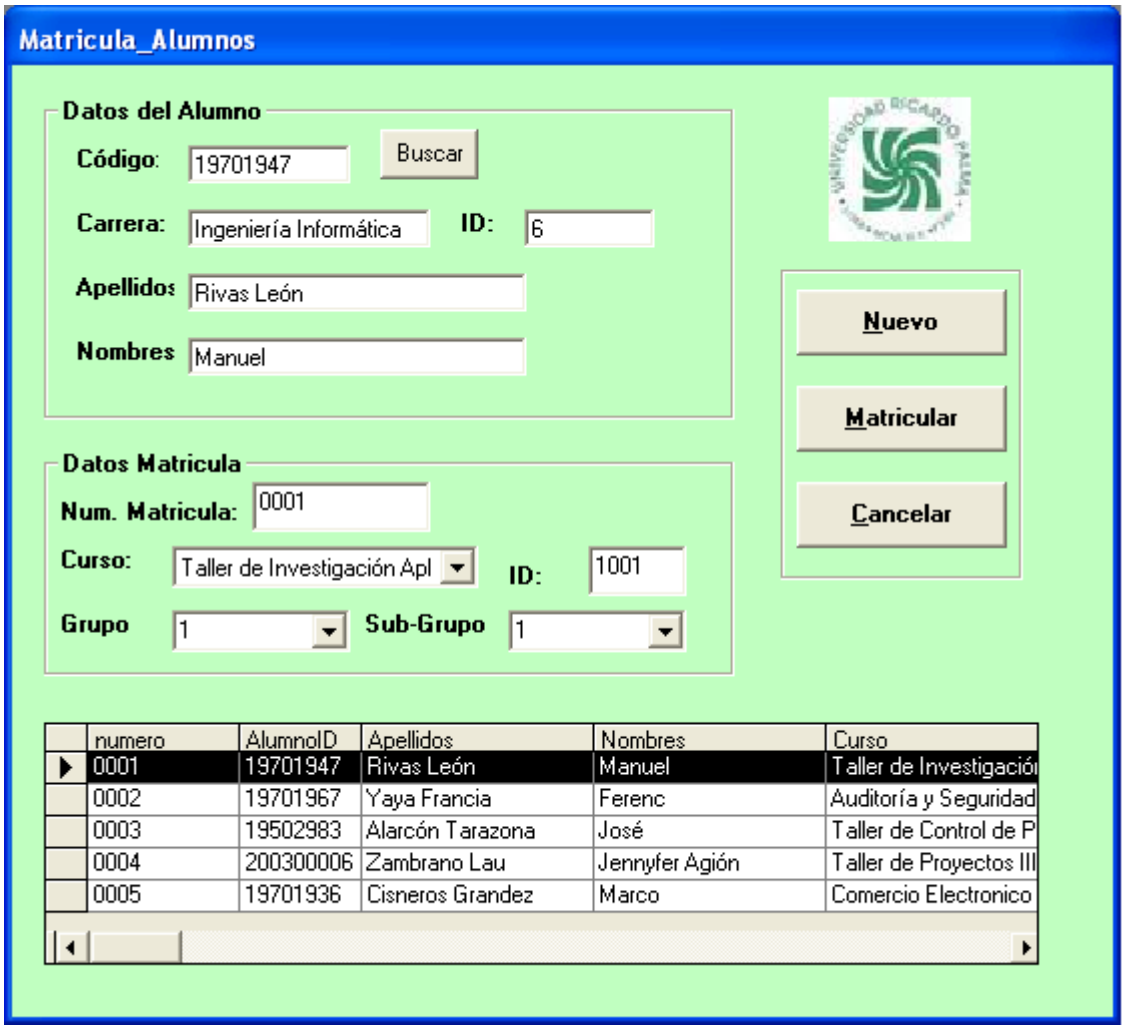


Figura 7.2. Sistema Heredado: Interfaz de Matricula de Alumnos

- **Interfaz de Mantenimiento de Alumnos**

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto al mantenimiento de los alumnos. (Ver figuras 7.3).

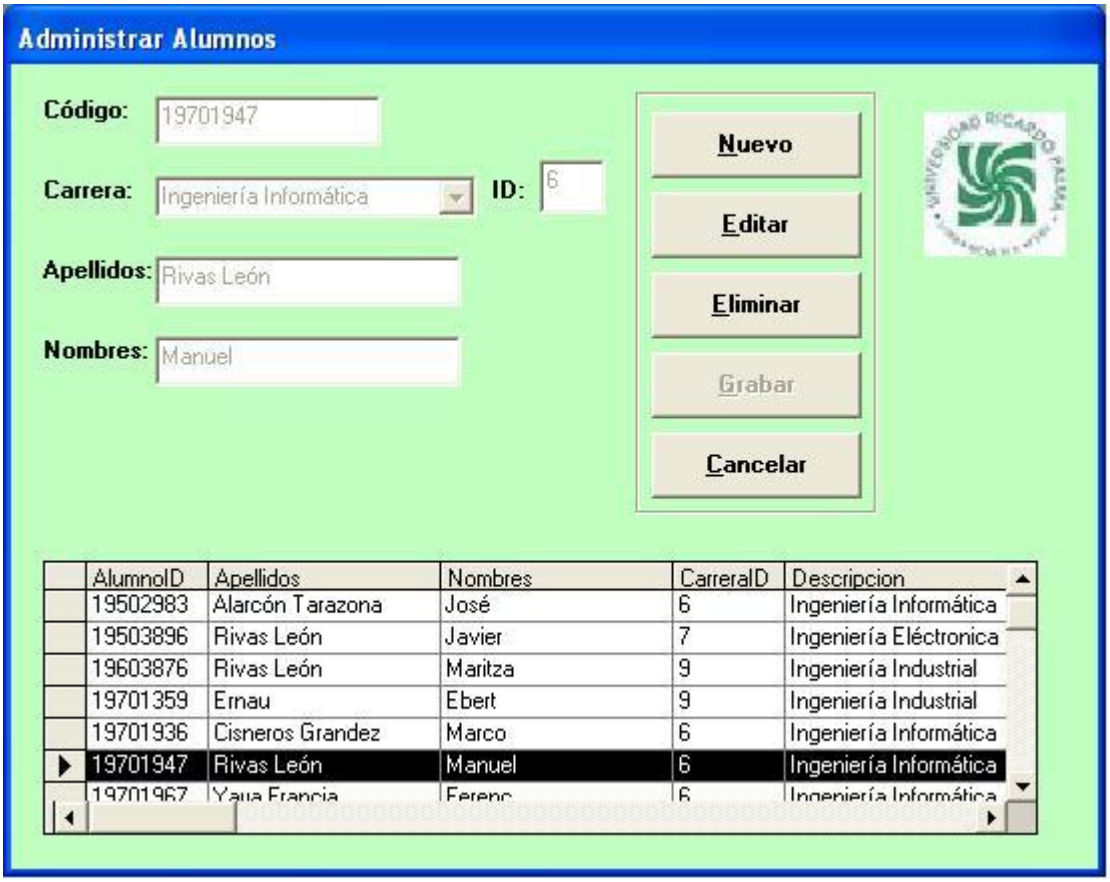


Figura 7.3. Sistema Heredado: Interfaz de Mantenimiento de Alumnos

- **Interfaz de Mantenimiento de Docentes**

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto al mantenimiento de los docentes. (Ver figura 7.4).

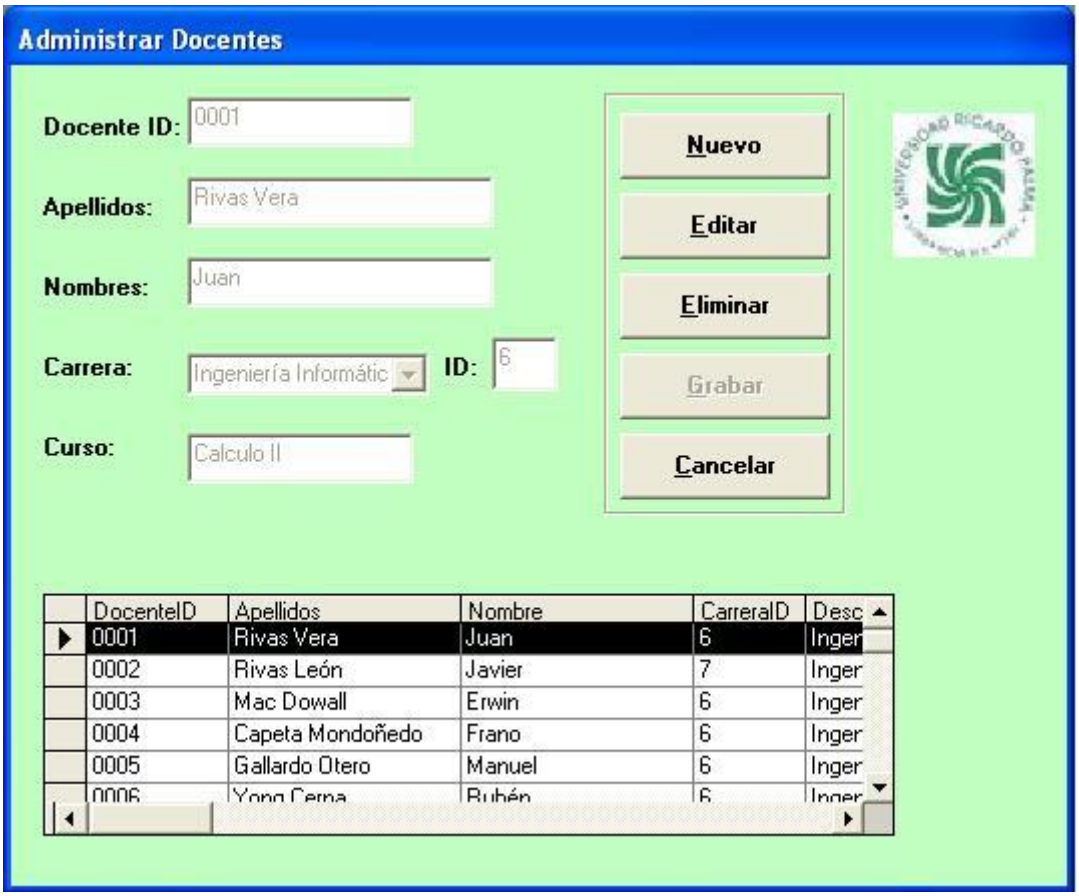


Figura 7.4. Sistema Heredado: Interfaz de Mantenimiento de Docentes

- **Interfaz de Mantenimiento de Cursos**

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto al mantenimiento de los cursos. (Ver figura 7.5).

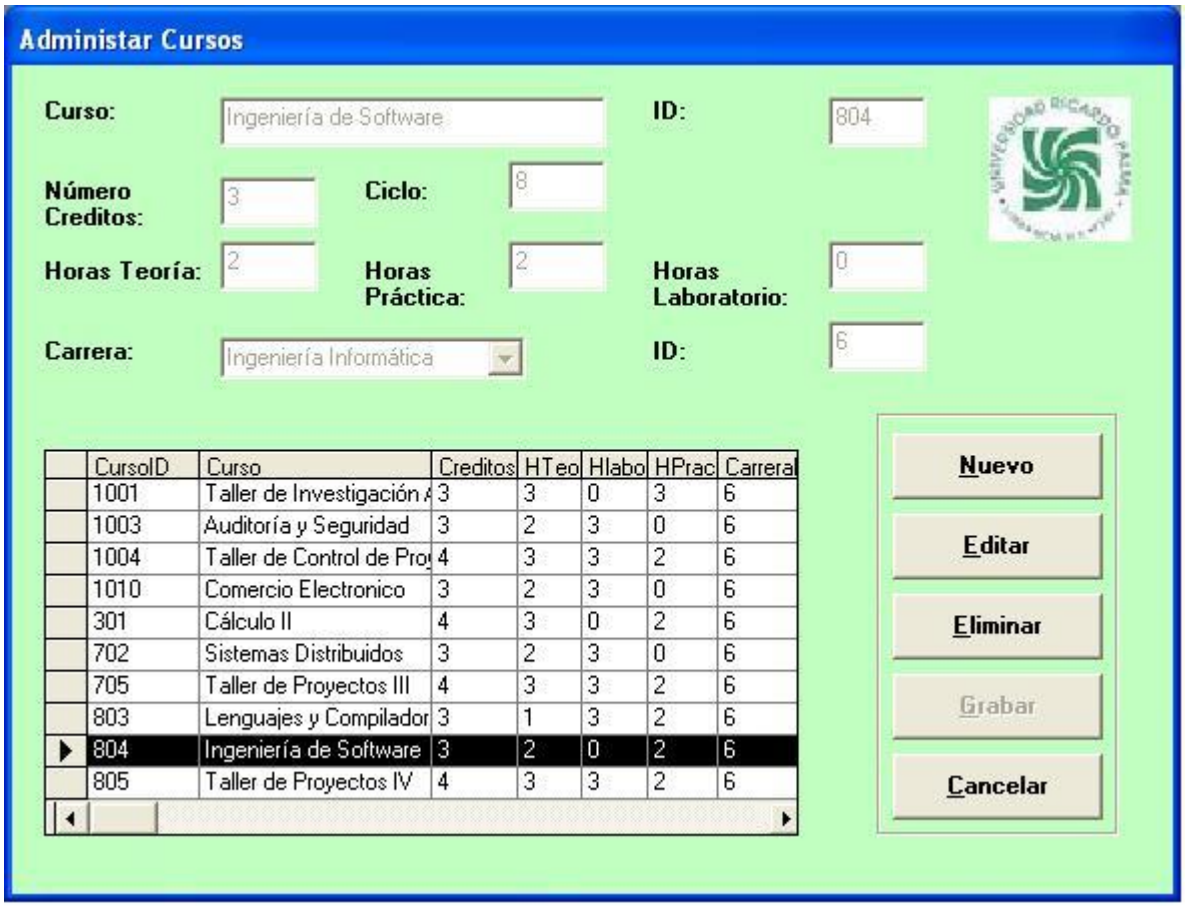


Figura 7.5. Sistema Heredado: Interfaz de Mantenimiento de Cursos

- **Mantenimiento de Carreras**

En esta interfaz, se aprecia la funcionalidad del Sistema Heredado con respecto al mantenimiento de carreras registradas. (Ver figura 7.6).

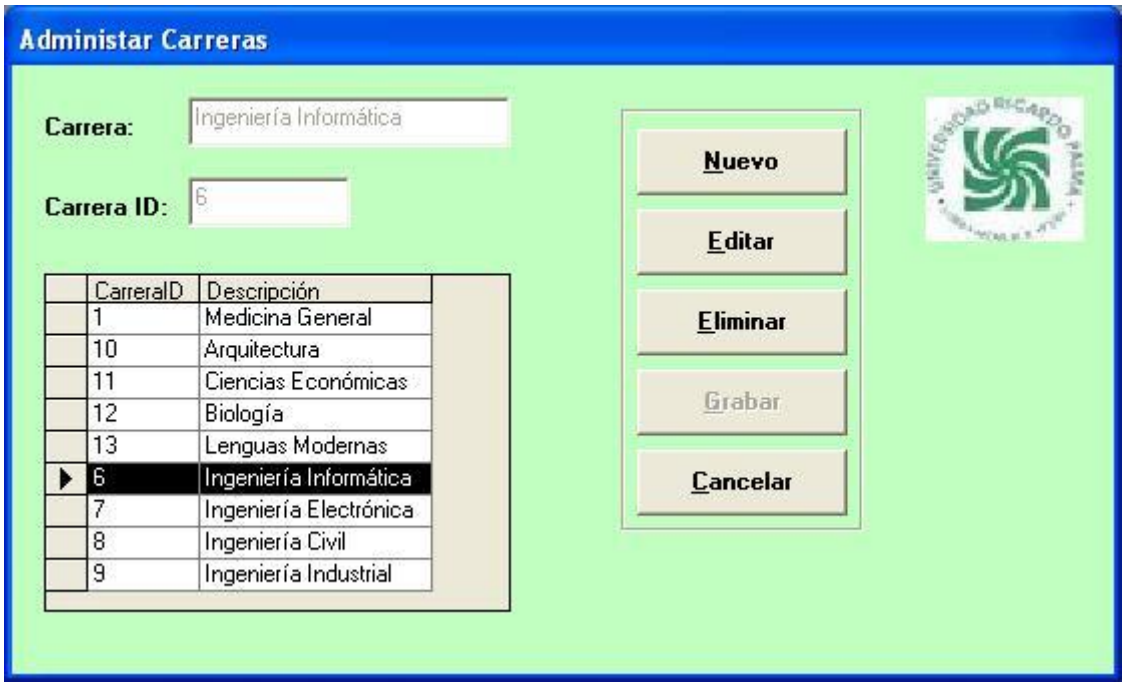


Figura 7.6. Sistema Heredado: Interfaz de Mantenimiento de Carreras

ANEXO 3

CODIGO Y INTERFACES DEL WEB SERVICES

Código del Servicio Wrapper

A continuación, se muestra el código del servicio *Wrapper*. (Ver código 7.1)

```
using System;
using System.Data;
using System.Data.SqlClient;
namespace SistemaLegacy
{
     /// <summary>
     /// Descripción breve de Wrapper.
     /// </summary>


     /// <summary>
     /// SE DEFINE EL ADAPTADOR DE SERVICIOS
     /// </summary>
     public class Wrapper
     {
          private SqlDataAdapter da;
          private DataSet ds = new DataSet();
          private SqlConnection con;

          public Wrapper()
          {
          //cadena de conexión a la base de datos
          //la base de datos se establecerá o indicará en
Web.config
          string[] cadena ;
          cadena                                          =
System.Configuration.ConfigurationSettings.AppSettings.GetV
alues("conexion");
          con = new SqlConnection(cadena[0]) ;
          }

//RETORNA MEDIANTE EL DATASET TODOS LOS CURSOS REGISTRADOS
EN LA BASE DE DATOS (STORED PROCEDURES)
          public DataSet VerCursos ()
          {
               da=new SqlDataAdapter("spCursos",con);

     da.SelectCommand.CommandType=CommandType.StoredProcedu
re;
               da.Fill(ds,"cursos");
               return ds;
```

```
          }
//RETORNA MEDIANTE EL DATASET TODOS LOS ALUMNOS REGISTRADOS
EN LA BASE DE DATOS (STORED PROCEDURES)
          public DataSet ConsultaAlumnos ()
          {
               da = new SqlDataAdapter("spConsultaAlumno",
con);

     da.SelectCommand.CommandType=CommandType.StoredProcedu
re;
               da.Fill (ds, "Alumnos");
               return ds;
          }
//RETORNA   MEDIANTE   EL   DATASET   TODOS   LAS   CARRERAS
REGISTRADOS EN LA BASE DE DATOS (STORED PROCEDURES)
          public DataSet ConsultaCarreras ()
          {
               da = new SqlDataAdapter("spCarrera", con);

     da.SelectCommand.CommandType=CommandType.StoredProcedu
re;
               da.Fill (ds, "Carreras");
               return ds;
          }
//RETORNA   MEDIANTE   EL   DATASET   TODOS   LOS   DOCENTES
REGISTRADOS EN LA BASE DE DATOS (STORED PROCEDURES)
          public DataSet ConsultaDocentes()
          {
               da                    =                    new
SqlDataAdapter("spConsultaDocentes", con);

     da.SelectCommand.CommandType=CommandType.StoredProcedu
re;
               da.Fill (ds, "Docentes");
               return ds;
          }

          public DataSet ConsultaAlumnoMatriculado (string
numero)//Busca alumnos matriculados por numero de matricula
          {
               da                    =                    new
SqlDataAdapter("spAlumnoMatriculado", con);

     da.SelectCommand.CommandType=CommandType.StoredProcedu
re;
               da.SelectCommand.Parameters.Add   ("@numero",
numero);
               da.Fill (ds, "matriculado");
               return ds;
          }
```

```
          public  DataSet  ConsultaMatriculados  ()//MUESTRA
TODOS LOS ALUMNOS EXISTENTES
          {
               da                    =                    new
SqlDataAdapter("spConsultarMatriculados", con);

     da.SelectCommand.CommandType=CommandType.StoredProcedu
re;
               da.Fill (ds, "Todas");
               return ds;
          }
     }
}
```

Código 7.1: Código del *Wrapper*

Código del Servicio Cliente del *Web Services*

A continuación, se muestra el código de llamada al servicio *Wrapper* por parte del servicio cliente de *Web Services*, para la invocación de los servicios (invocación de funcionalidades adaptadas del Sistema Heredado). (Ver código 7.2)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.Data.SqlClient;

namespace SistemaLegacy
{
     /// <summary>
     /// Descripción breve de Consultas.
     /// </summary>
     public           class           Consultas           :
System.Web.Services.WebService
     {
          private DataSet ds = new DataSet();
          public Consultas()
          {
               InitializeComponent();
          }
          #region Component Designer generated code
          //Required by the Web Services Designer
```

```
        private IContainer components = null;
        /// <summary>
        /// Required method for Designer support - do not
modify
        ///  the  contents  of  this  method  with  the  code
editor.
        /// </summary>
        private void InitializeComponent()
        {
        }
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #endregion

//DEVUELVE LA DATA DE ALUMNOS REGISTRADOS EN LA BD MEDIANTE
LA INVOCACIÓN DE LA CLASE WRAPPER
        [WebMethod]
        public DataSet ConsultaAlumnos ()
        {
            Wrapper  w=new  Wrapper();  //Creación  del
objeto para invocar la clase Wrapper (w)
            return   w.ConsultaAlumnos();   //   invoca
mediante el obj creado, los alumnos registrados
        }
//DEVUELVE  LA  DATA  DE  CARRERAS  REGISTRADOS  EN  LA  BD
MEDIANTE LA INVOCACIÓN DE LA CLASE WRAPPER
        [WebMethod]
        public DataSet ConsultaCarreras ()
        {
            Wrapper  w=new  Wrapper();  //Creación  del
objeto para invocar la clase Wrapper (w)
            return  w.ConsultaCarreras();    //  invoca
mediante el obj creado, las carreras registradas
        }
//DEVUELVE LA DATA DE CURSOS REGISTRADOS EN LA BD MEDIANTE
LA INVOCACIÓN DE LA CLASE WRAPPER
        [WebMethod]
        public DataSet VerCursos()
        {
            Wrapper  w=new  Wrapper();   //Creación  del
objeto para invocar la clase Wrapper (w)
```

```
                return w.VerCursos();     // invoca mediante
el obj creado, los cursos registrados
            }
//DEVUELVE LA DATA DE DOCENTES REGISTRADOS EN LA BD
MEDIANTE LA INVOCACIÓN DE LA CLASE WRAPPER
            [WebMethod]
            public DataSet ConsultarDocentes()
            {
                Wrapper w=new Wrapper();    //Creación del
objeto para invocar la clase Wrapper (w)
                return w.ConsultaDocentes();     //  invoca
mediante el obj creado, los docentes registrados
            }
//DEVUELVE LA DATA DE ALUMNOS MATRICULADOS PARA LA
INVOCACIÓN DE LA CLASE WRAPPER
            [WebMethod]
            public DataSet Matriculados()
            {
                Wrapper w=new Wrapper();    //Creación del
objeto para invocar la clase Wrapper (w)
                return w.ConsultaMatriculados(); //  invoca
mediante el obj creado, los alumnos matriculados
            }
//DEVUELVE LA DATA DE UN ALUMNO MATRICULADO PARA LA
INVOCACIÓN DE LA CLASE WRAPPER
            [WebMethod]
            public DataSet Matriculado(string numero)
            {
                Wrapper w=new Wrapper();     //Creación del
objeto para invocar la clase Wrapper (w)
                return w.ConsultaAlumnoMatriculado(numero);
// invoca mediante el obj creado, un alumno matriculado
            }
        }
}
```

Código 7.2: Código de llamada al servicio *Wrapper* por parte del *Web Services*

Interfaces del *Web Services*

A continuación, se muestra la aplicabilidad de la nueva estrategia de *Web Services* y sus estándares. Esta estrategia permite reutilizar y exponer las funcionalidades existentes del Sistema Heredado.

- **Interfaces de Prueba del *Web Services***

A continuación, se muestran las interfaces de invocación de servicios del Web Services (reutilizando las funcionalidades del Sistema Heredado).

En esta interfaz, se muestran los servicios de invocación de consulta del *Web Services*, las cuales se encuentran ubicadas dentro de un repositorio de servicios (UDDI) y establecidos dentro de un WSDL. (Ver figura: 7.7).

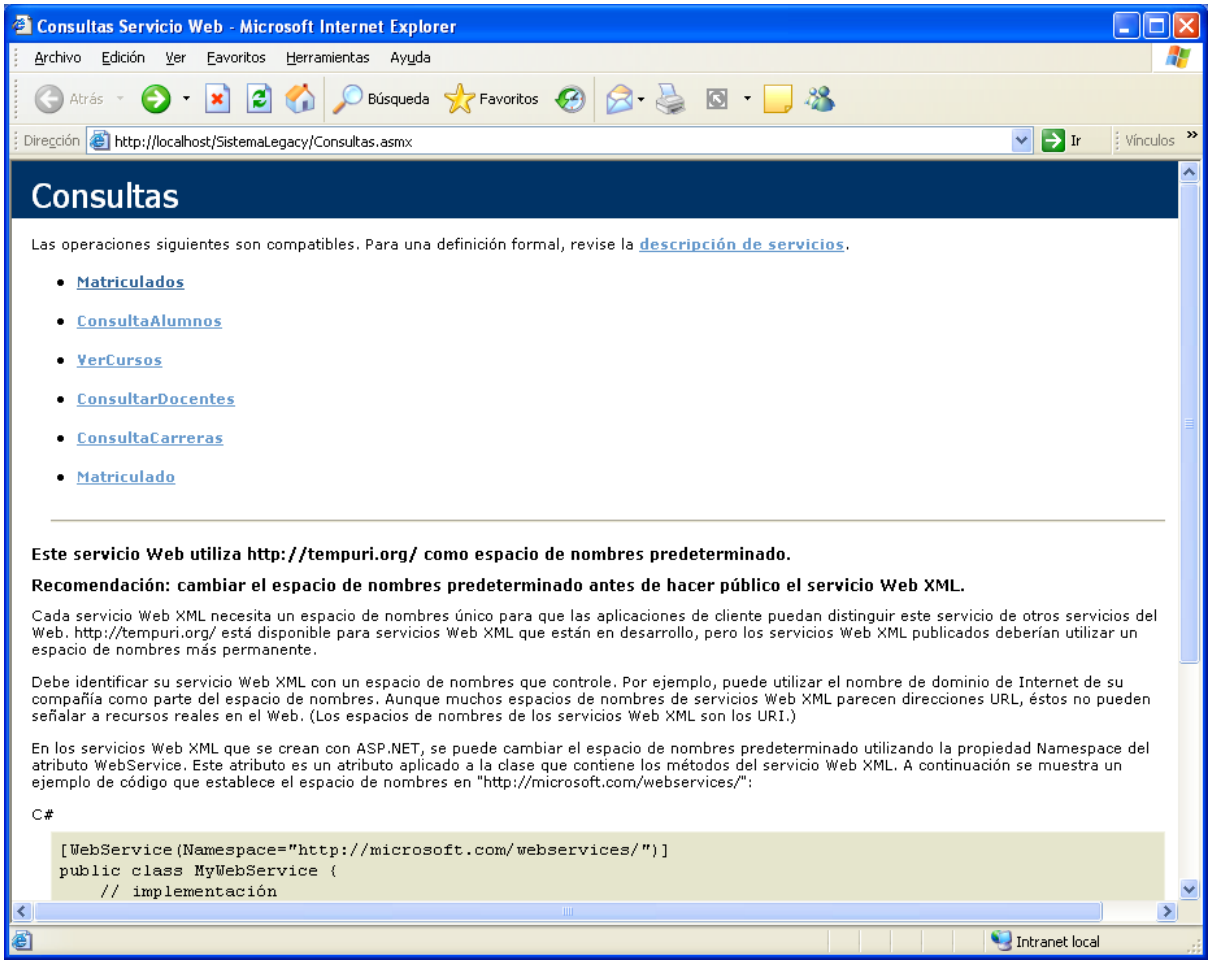


Figura 7.7. Prueba del Web Services: Interfaz invocación de servicios

- **Invocar Alumnos Matriculados**

En estas interfaces, se permite demostrar la invocación de la funcionalidad del Sistema Heredado con respecto a los alumnos matriculados utilizando *Web Services*, el cual

brinda como respuesta de servicio estándar en XML. También, se puede apreciar el procesamiento de mensajes por medio de SOAP. (Ver figuras: 7.8, 7.9, 7.10, 7.11).

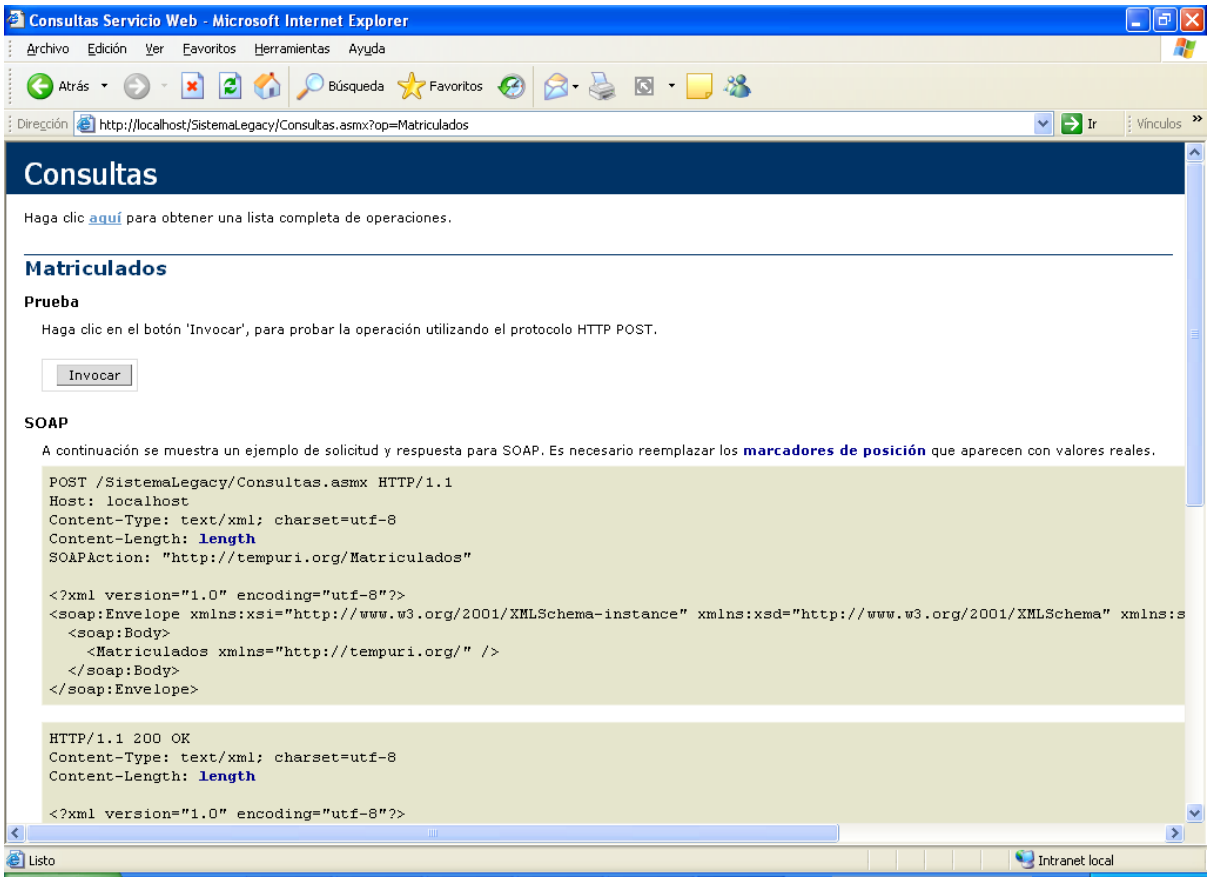


Figura 7.8. Prueba del Web Services: Interfaz invocación de servicios de consulta de Alumnos Matriculados

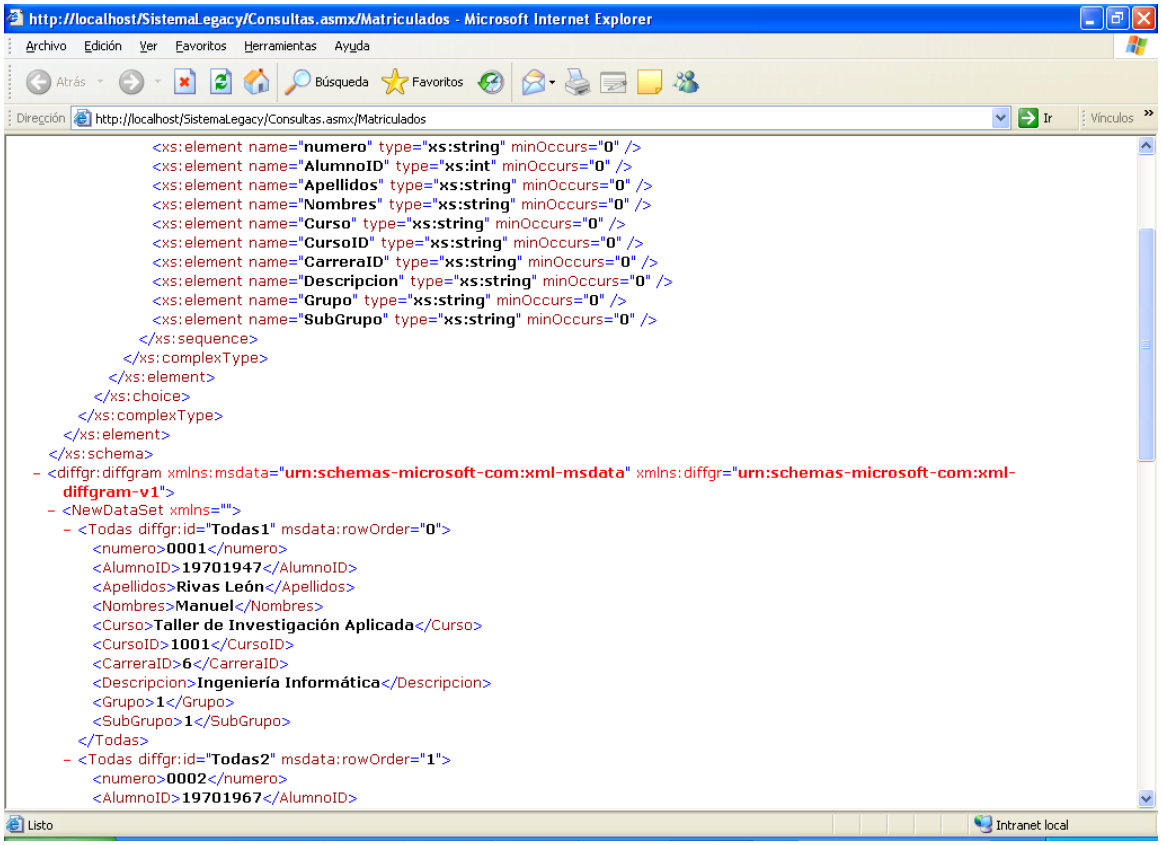


Figura 7.9. Interfaz *Web* de Respuesta al servicio de invocación de Alumnos Matriculados en XML

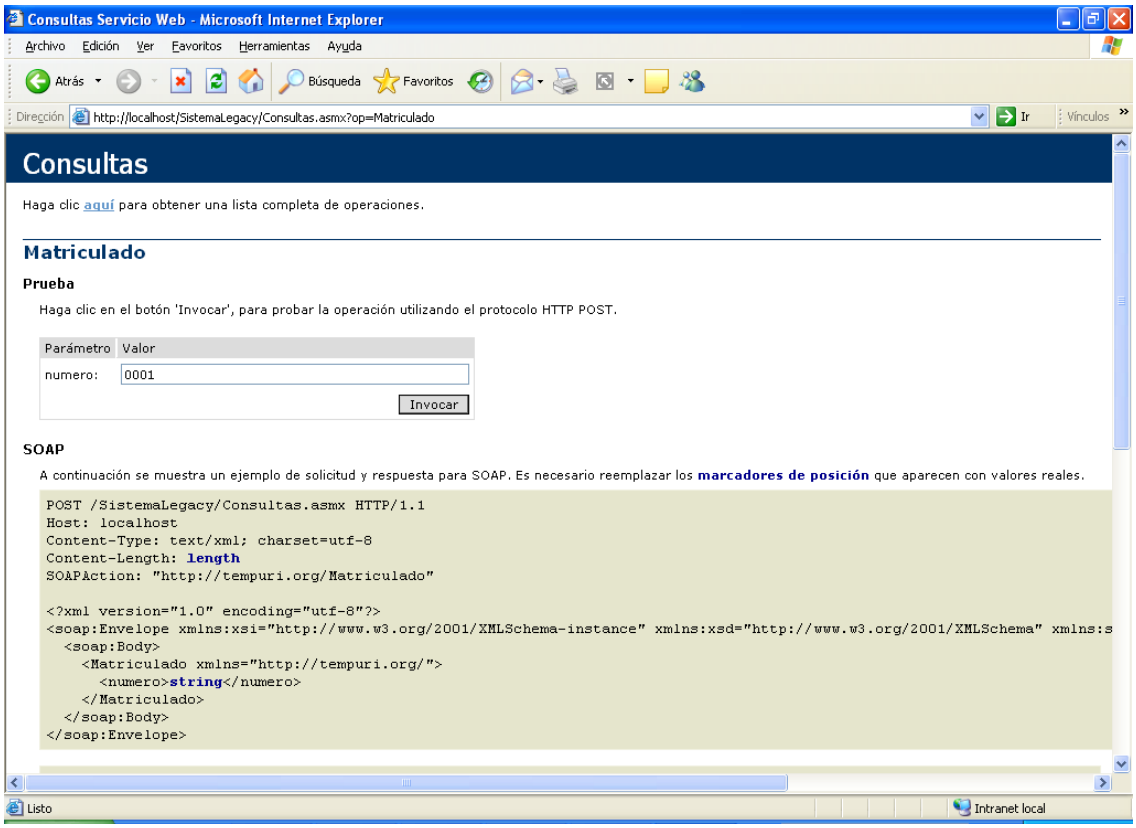


Figura 7.10. Prueba del Web Services: Interfaz invocación de servicios de consulta de Alumnos Matriculados por orden de matricula

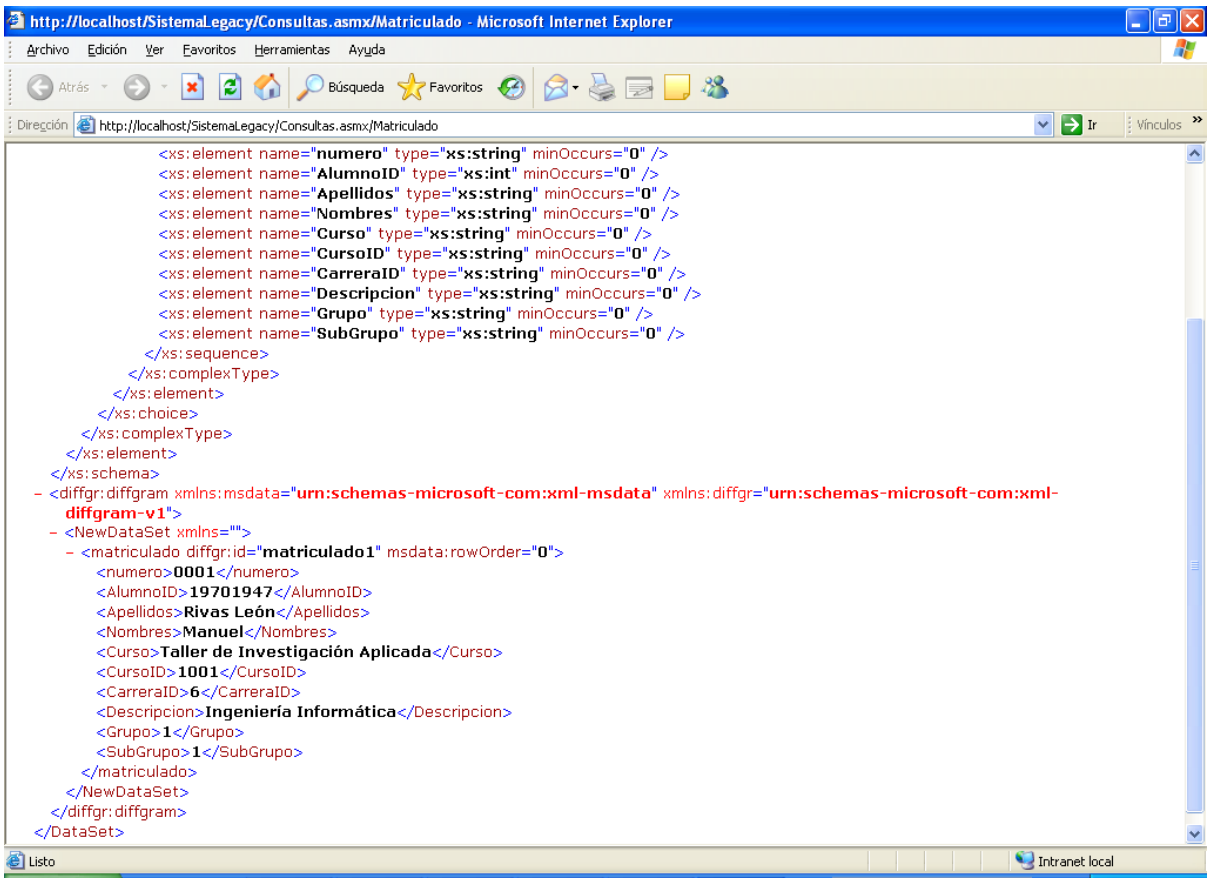


Figura 7.11. Interfaz *Web* de Respuesta al servicio de invocación de Alumnos Matriculados por orden de matrícula en XML

- **Invocar Alumnos**

En estas interfaces, se permite demostrar la invocación de la funcionalidad del Sistema Heredado con respecto a los alumnos utilizando *Web Services*, el cual brinda como respuesta de servicio estándar en XML. También, se puede apreciar el procesamiento de mensajes por medio de SOAP. (Ver figuras: 7.12, 7.13).

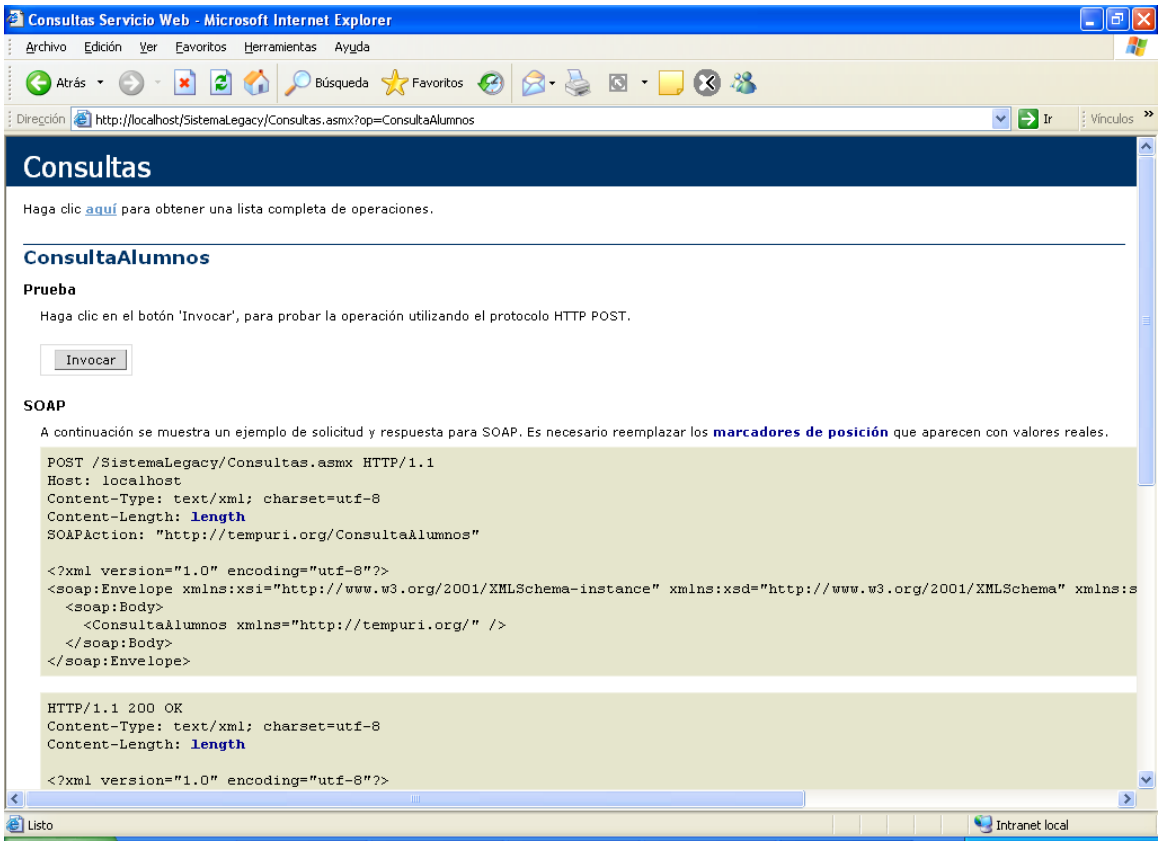


Figura 7.12. Prueba del Web Services: Interfaz invocación de servicios de consulta de Alumnos

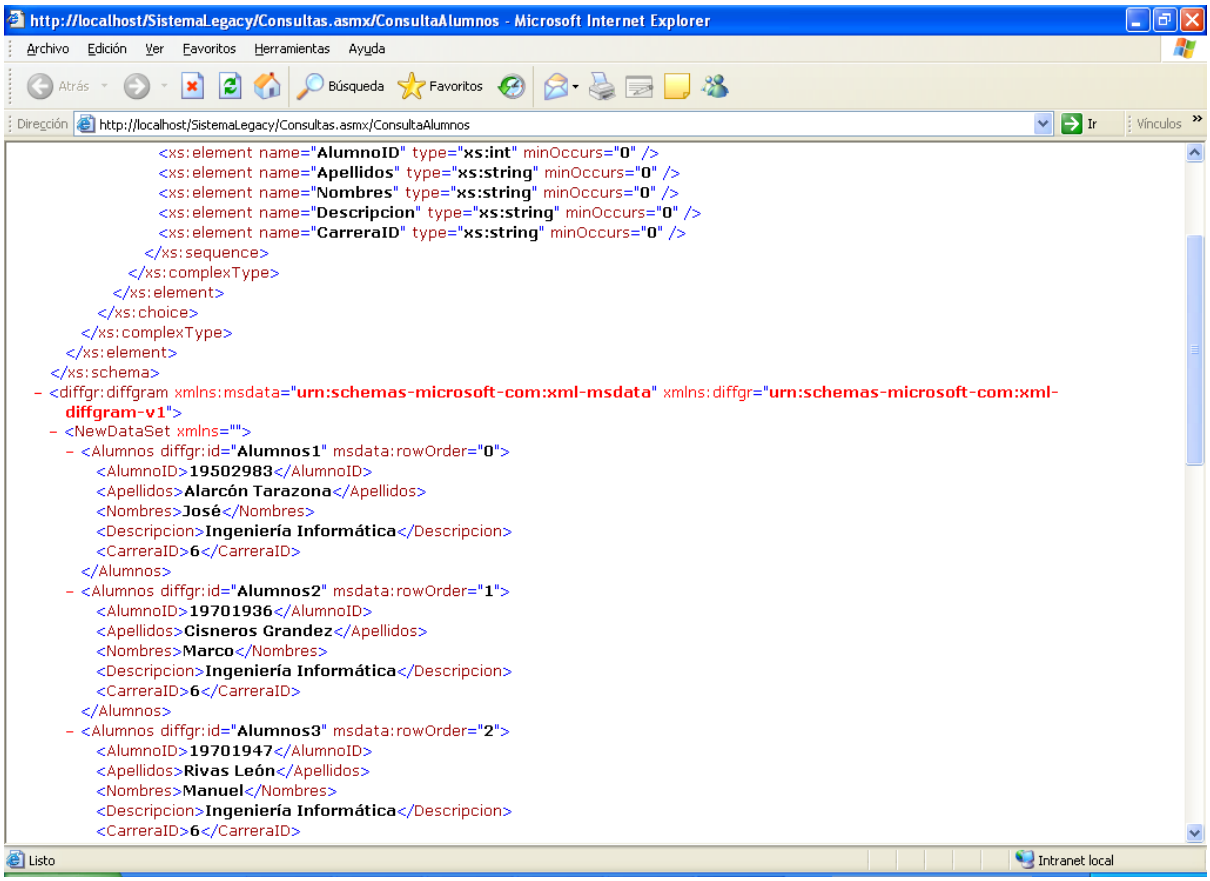


Figura 7.13. Interfaz *Web* de Respuesta al servicio de invocación de Alumnos en XML

- **Invocar Docentes**

En estas interfaces, se permite demostrar la invocación de la funcionalidad del Sistema Heredado con respecto a los docentes utilizando *Web Services*, el cual brinda como respuesta de servicio estándar en XML. También, se puede apreciar el procesamiento de mensajes por medio de SOAP. (Ver figuras: 7.14, 7.15).

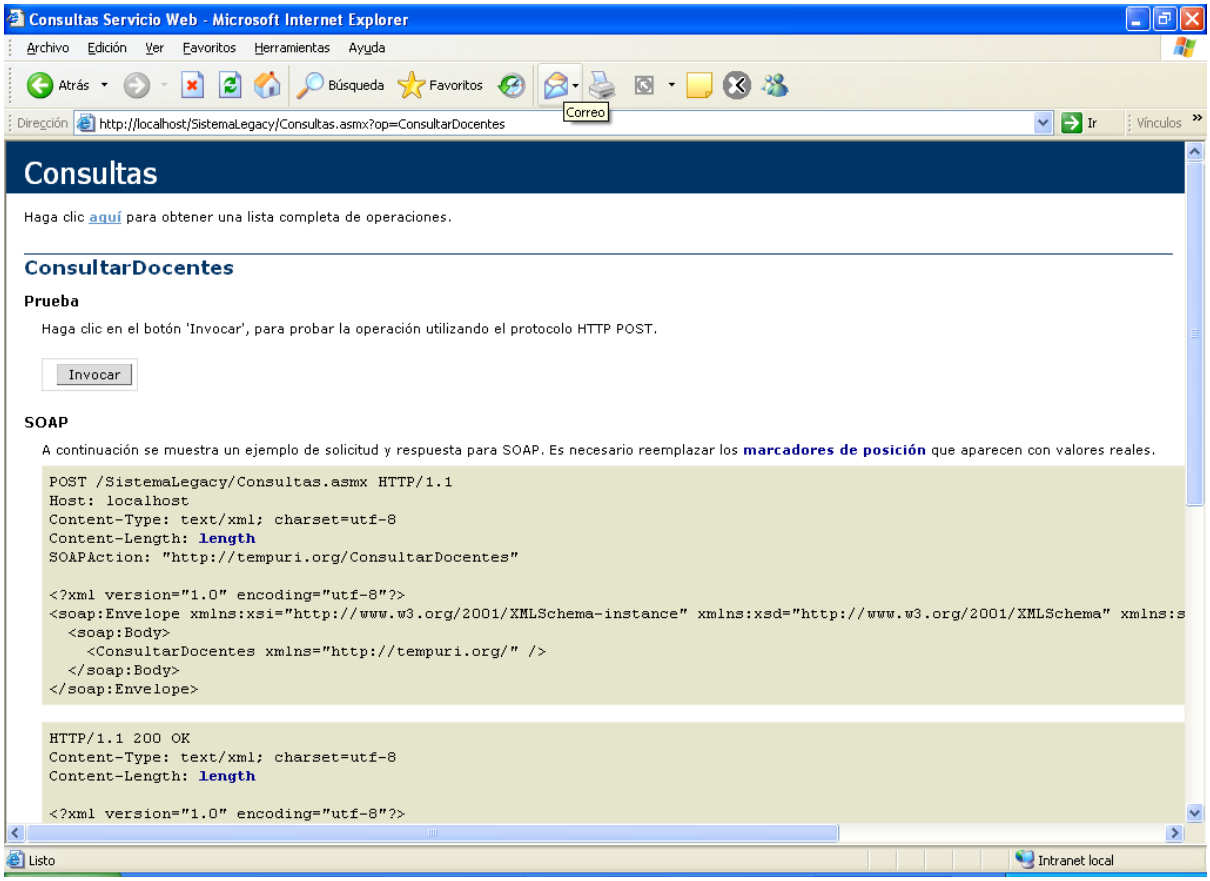


Figura 7.14. Prueba del Web Services: Interfaz invocación de servicios de consulta de Docentes

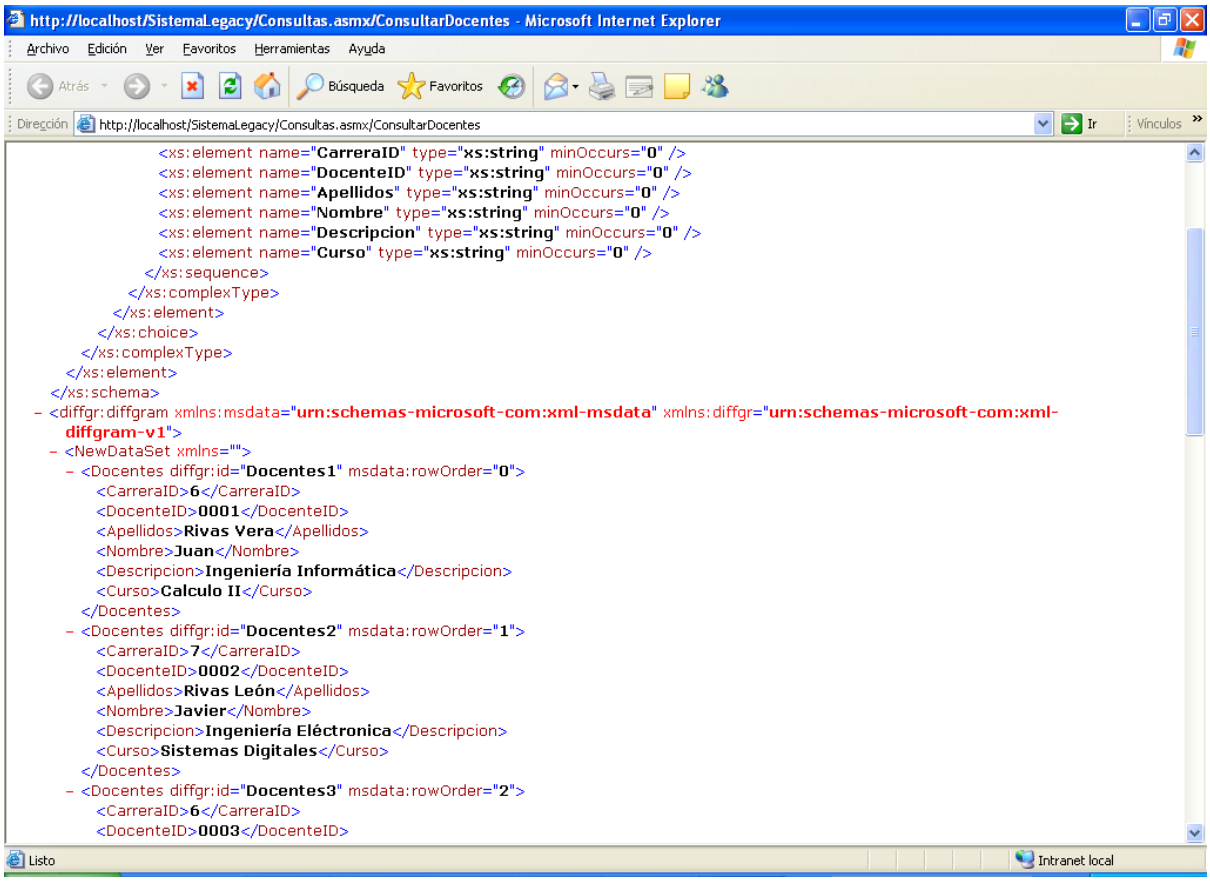


Figura 7.15. Interfaz *Web* de Respuesta al servicio de invocación de Docentes en XML

- **Invocar Cursos**

En estas interfaces, se permite demostrar la invocación de la funcionalidad del Sistema Heredado con respecto a los cursos utilizando *Web Services*, el cual brinda como respuesta de servicio estándar en XML. También, se puede apreciar el procesamiento de mensajes por medio de SOAP. (Ver figuras: 7.16, 7.17).

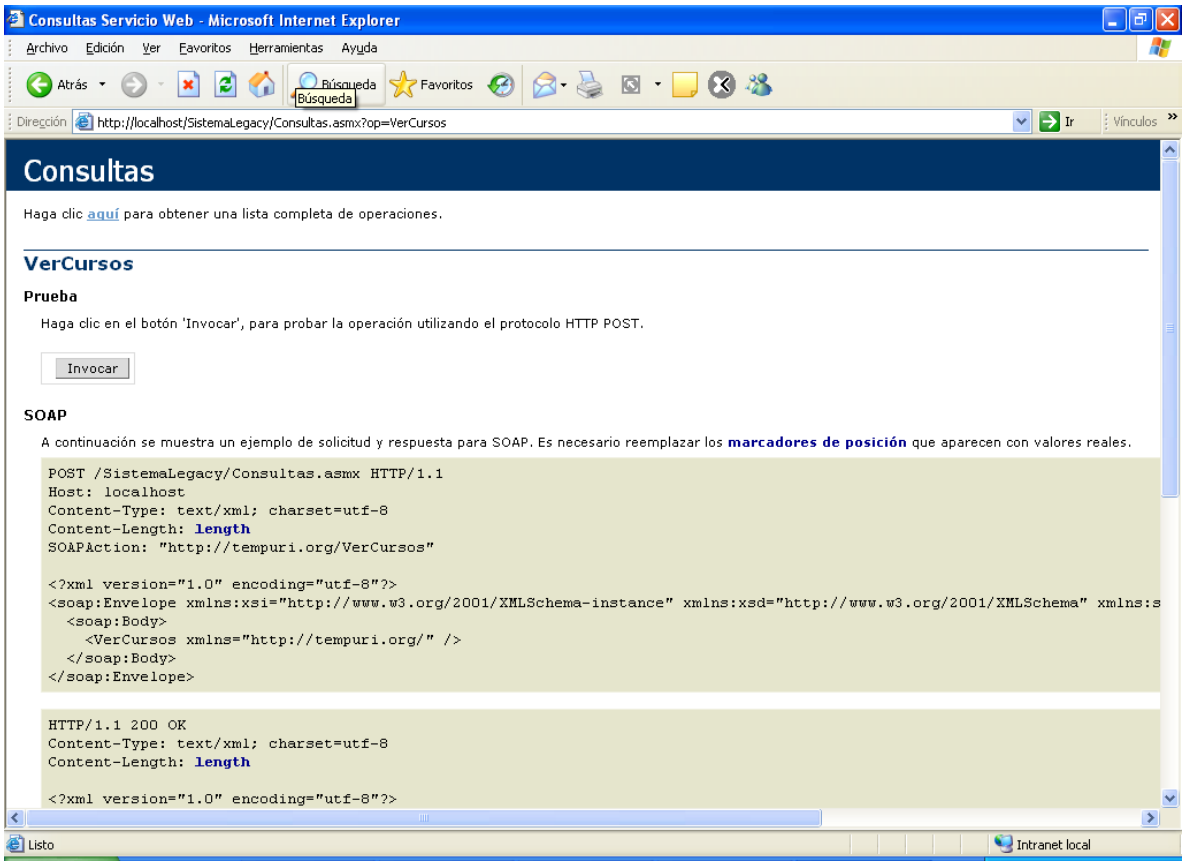


Figura 7.16. Prueba del Web Services: Interfaz invocación de servicios de consulta de Cursos

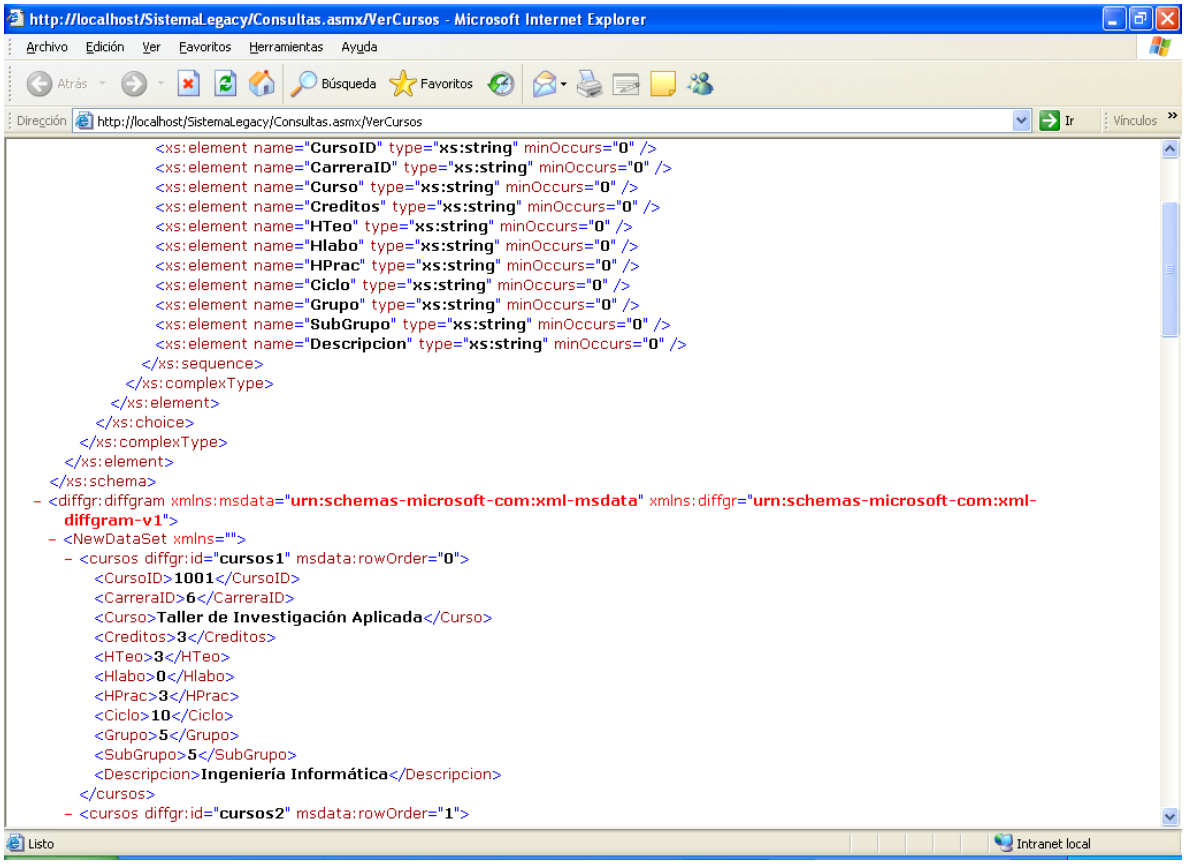


Figura 7.17. Interfaz *Web* de Respuesta al servicio de invocación de Cursos en XML

- **Invocar Carreras**

En estas interfaces, se permite demostrar la invocación de la funcionalidad del Sistema Heredado con respecto a las carreras utilizando *Web Services*, el cual brinda como respuesta de servicio estándar en XML. También, se puede apreciar el procesamiento de mensajes por medio de SOAP. (Ver figuras: 7.18, 7.19).

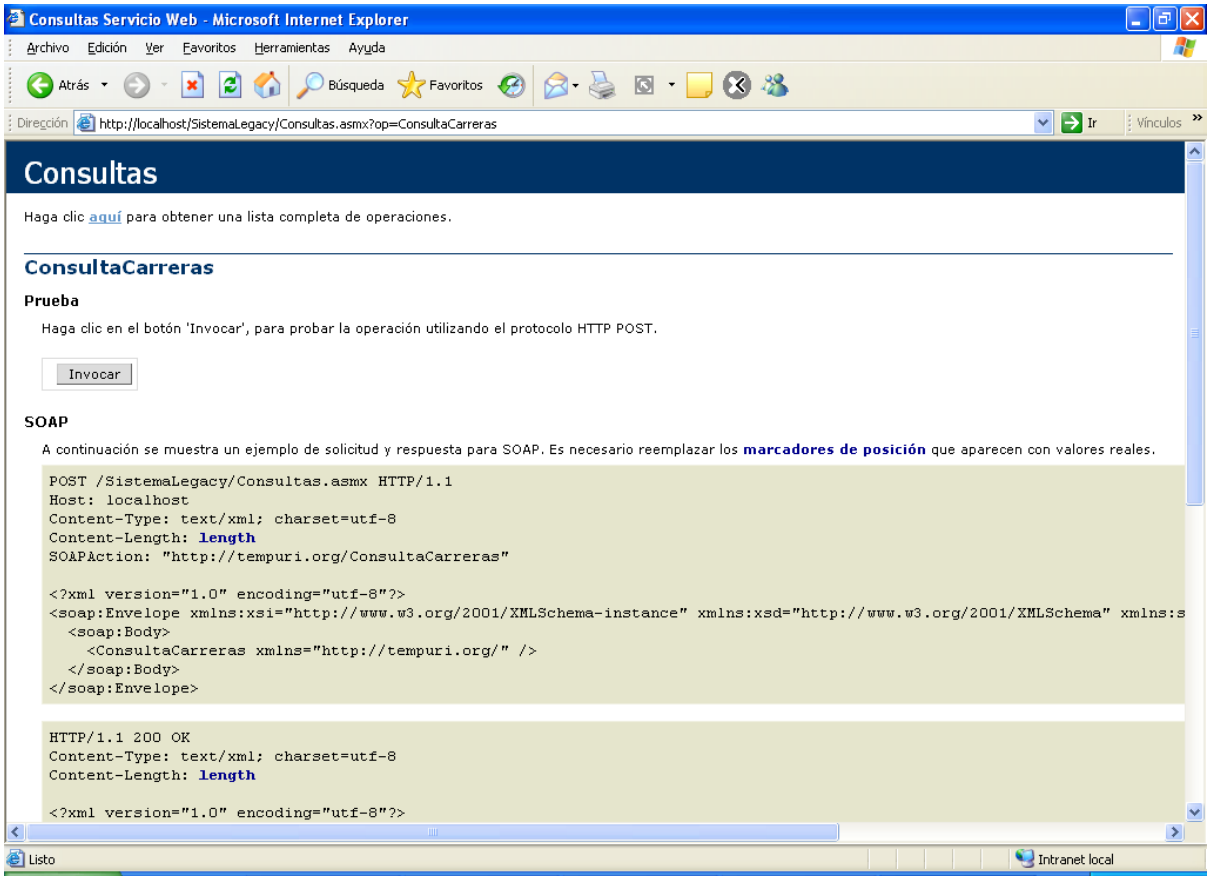


Figura 7.18. Prueba del Web Services: Interfaz invocación de servicios de consulta de Carreras

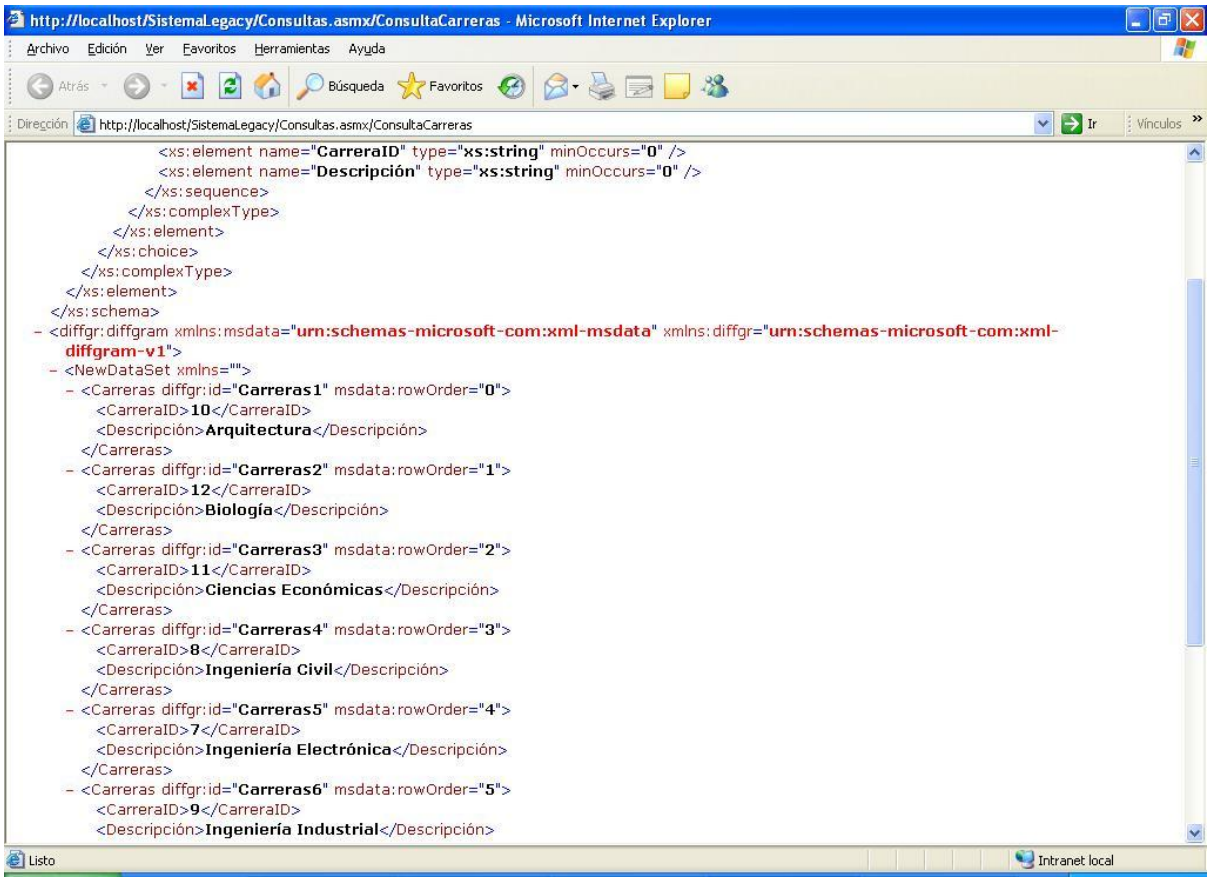


Figura 7.19. Interfaz *Web* de Respuesta al servicio de invocación de Carreras en XML

ANEXO 4

INTERFACES DE LA APLICACIÓN DE CONSUMO

Interfaces de Consumo del Web Services

A continuación, se muestran las interfaces de consumo del *Web Services*, como servicios de consulta de usuario en nuevos tipos de aplicaciones integrados. (Ver figuras 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26).

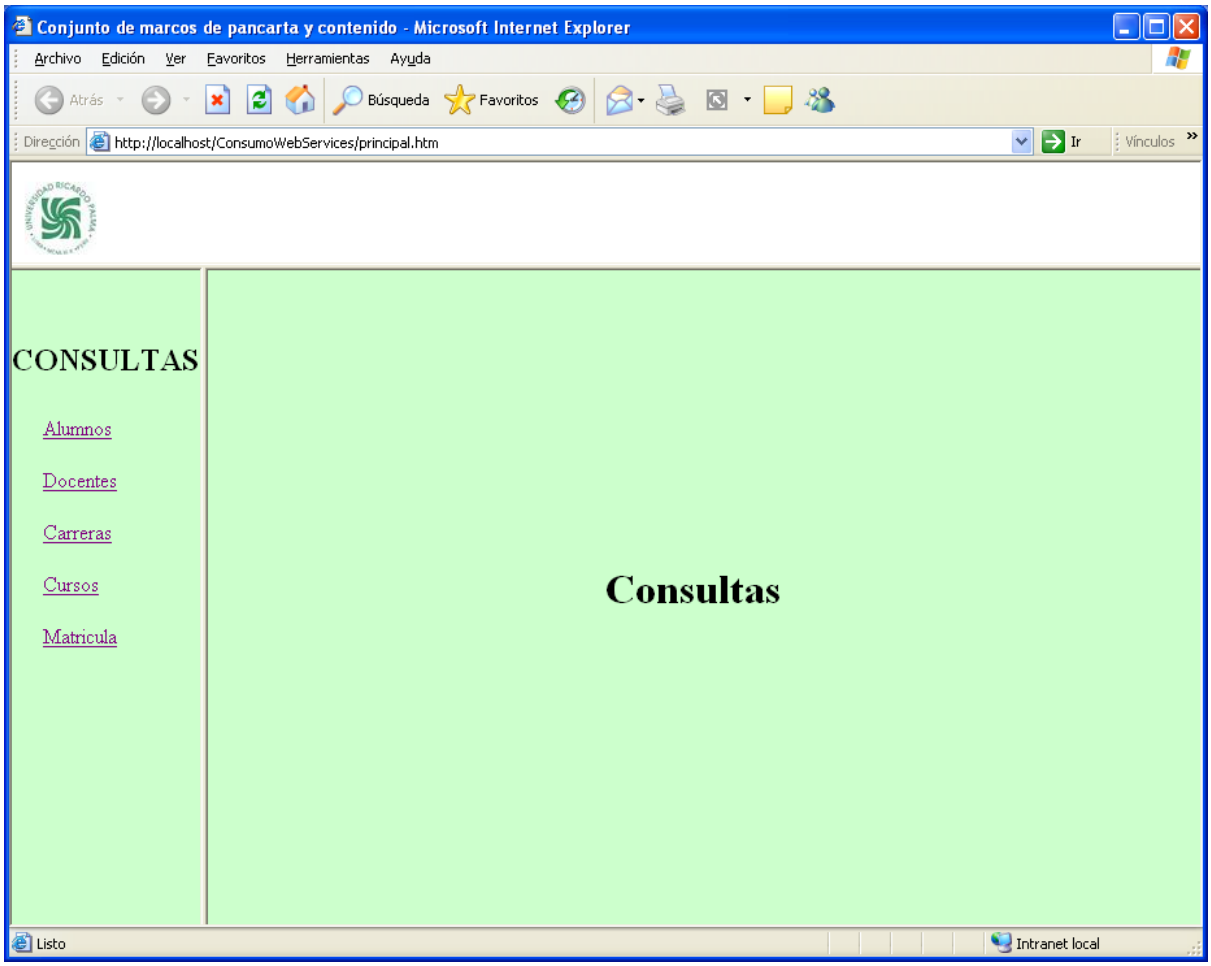


Figura 7.20. Consumo *Web Services*: Consultas

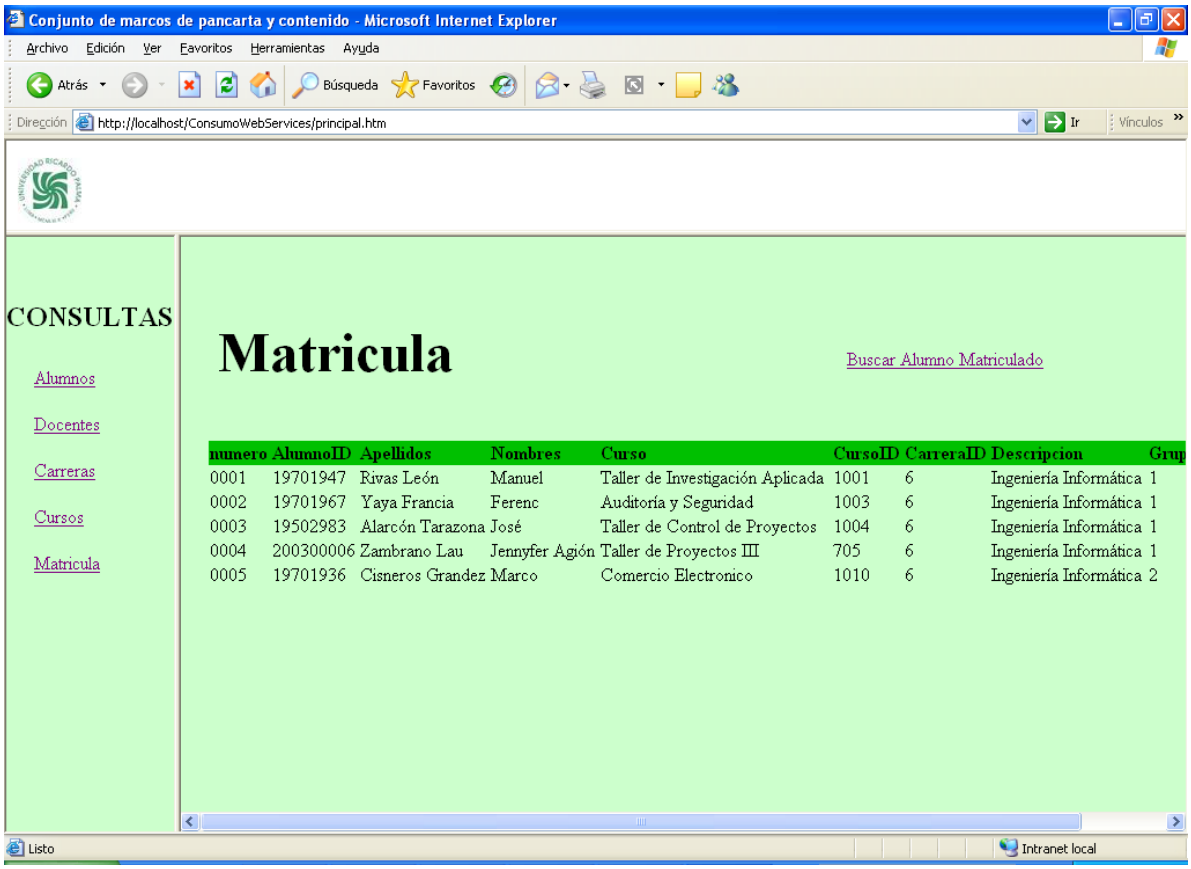


Figura 7.21. Consumo *Web Services* Alumnos Matriculados

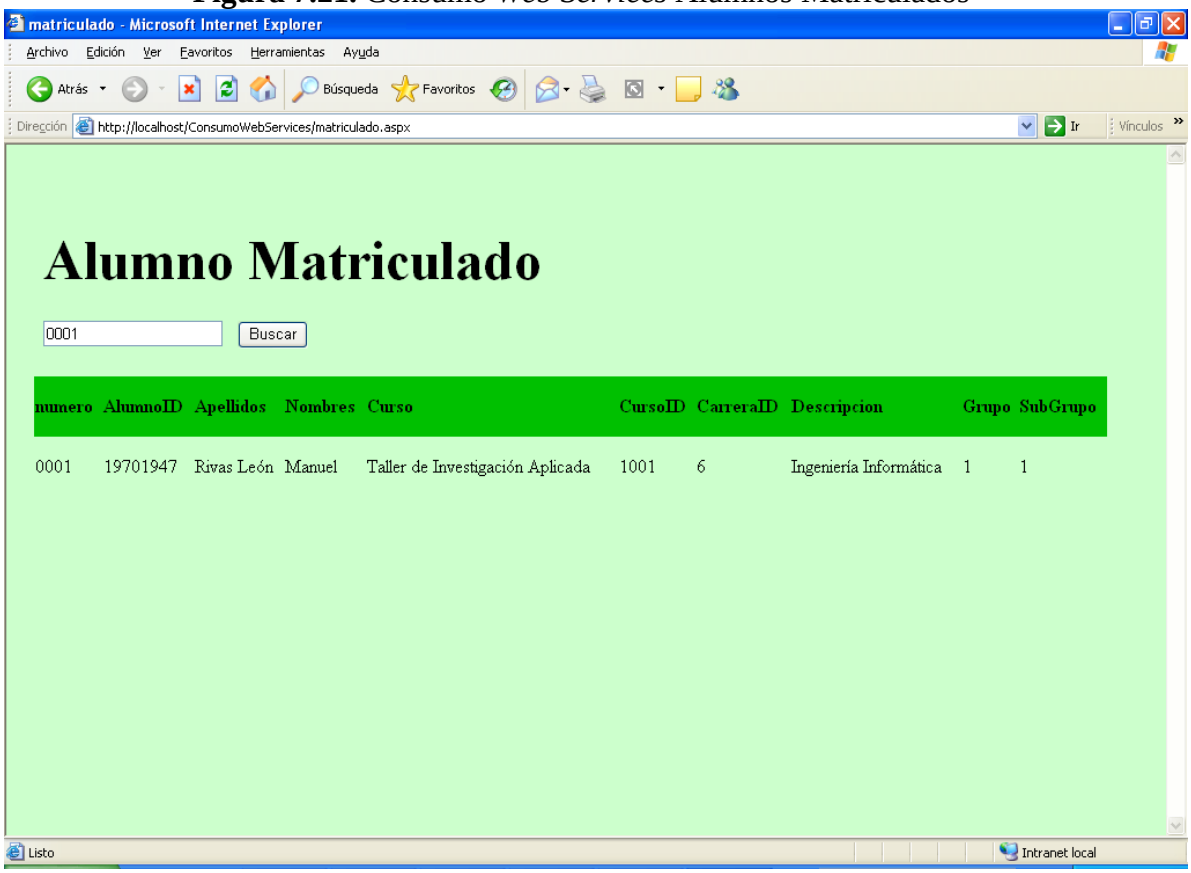


Figura 7.22. Consumo *Web Services* Alumnos Matriculados por búsqueda de número matricula

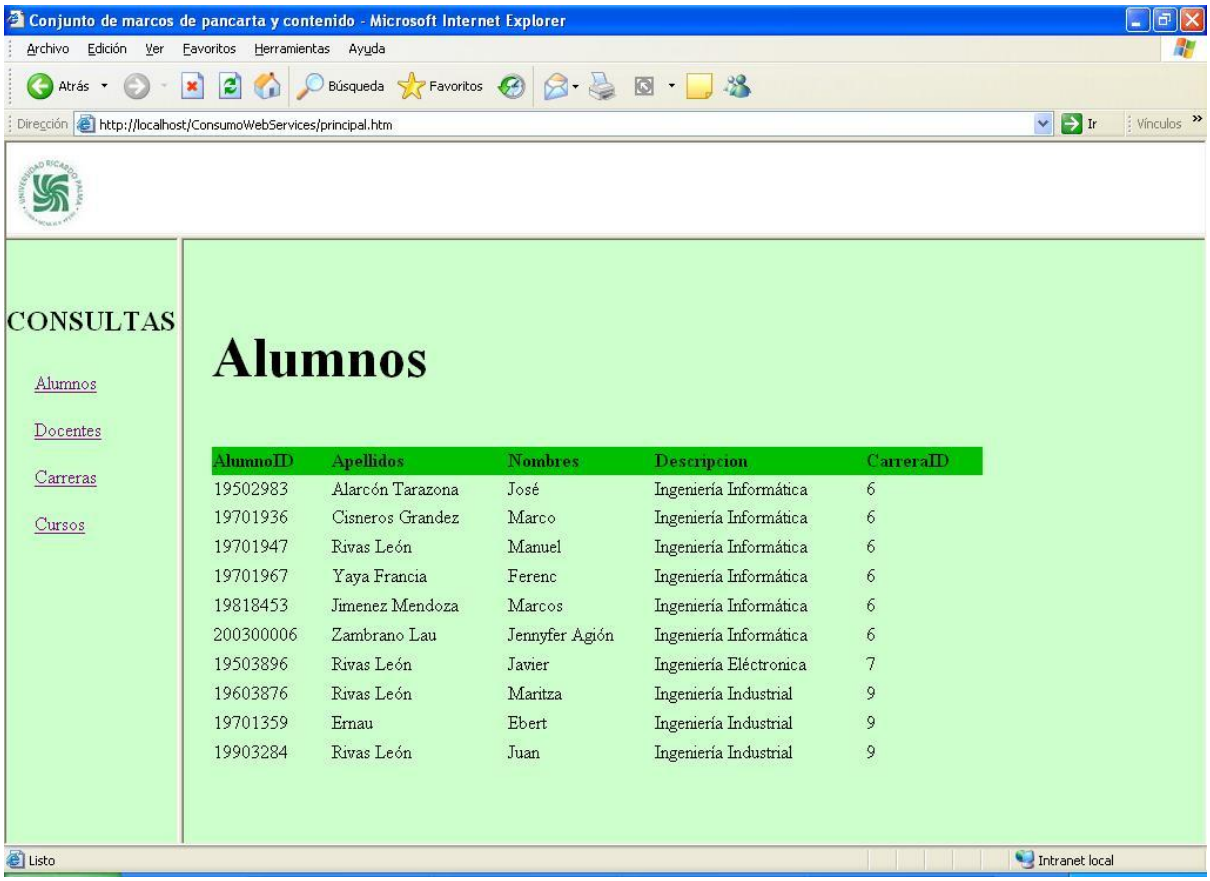


Figura 7.23. Consumo *Web Services* Alumnos

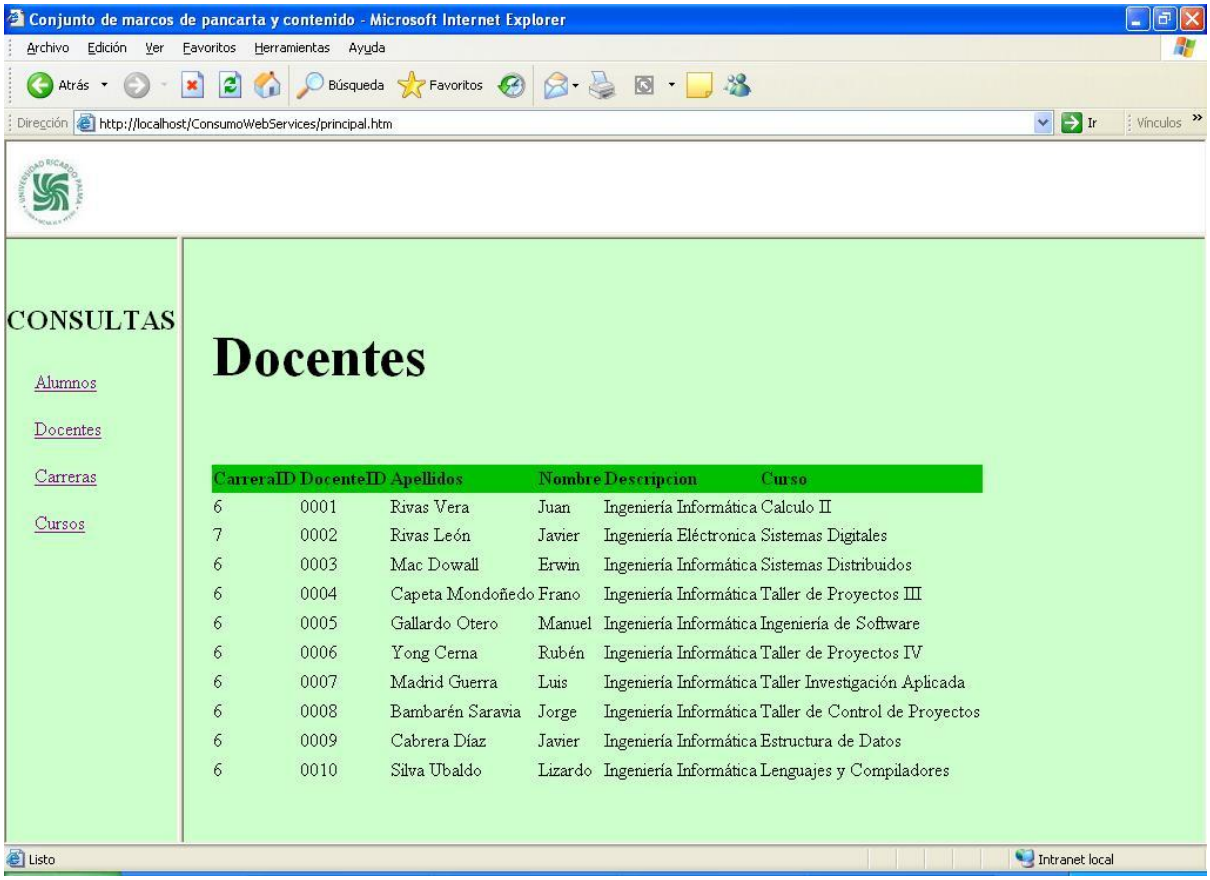


Figura 7.24. Consumo *Web Services* Docentes

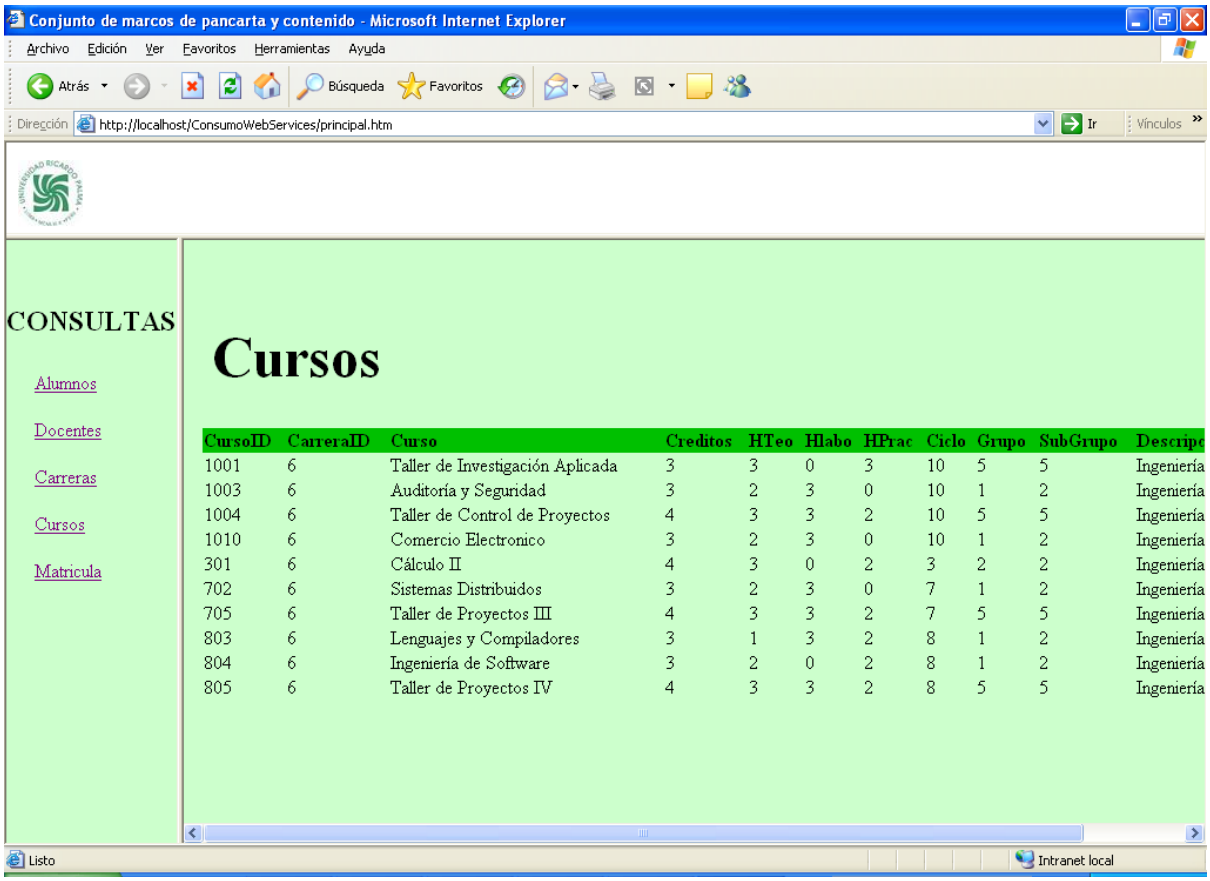


Figura 7.25. Consumo *Web Services* Cursos

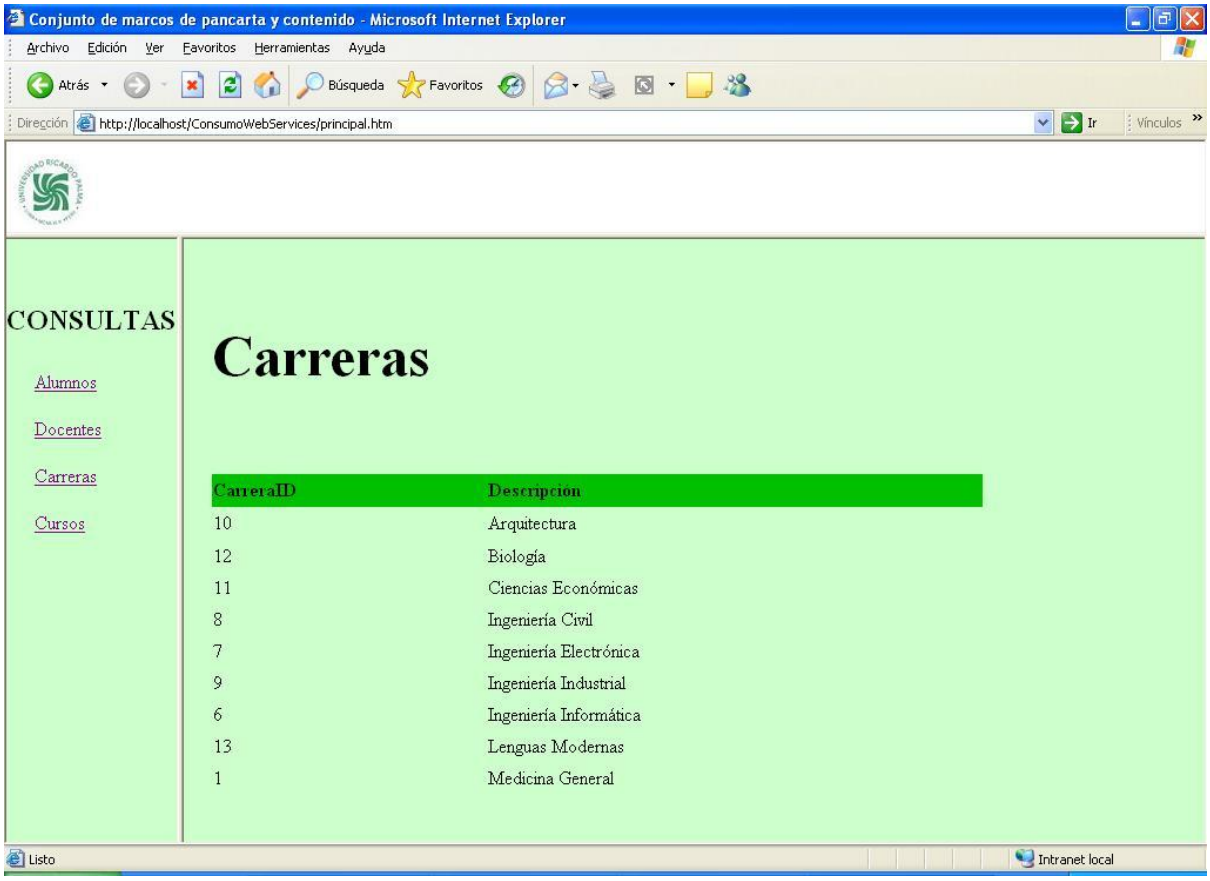


Figura 7.26. Consumo *Web Services* Carreras

Bibliografía

Artículos Científicos

Jaime Sarabia Alvarez-UDE, Carmen López Rincón, "XML, datos Heredados y bibliotecas digitales" Universidad Complutense de Madrid (Enero, 2004). Pag 1-14. pm02.pdf

Ignacio García, Macario Polo, Francisco Ruiz, Mario Piattini, "Servicios Web" Universidad de Castilla-La Mancha, España. (Enero, 2005). Pag 1-200. serviciosweb.pdf

Pablo Andrés Bianco, "Desarrollo de Aplicaciones Basadas en XML, Web Services para Dispositivos Móviles con Microsoft.NET Compact Framework" (Abril, 2005). Pag 1-81. 155_banco.pdf

M.Katrib, J.L.Pastrana, E.Pimentel "Composición de Coordinación de Componentes mediante Conectores y WebServices". (Enero, 2004). Pag 1-9. 138.pdf

Raul Ruggia, Jorge Besil, Carla Pais, Dario Sande. "Interoperabilidad entre Servidores de Aplicaciones Heterogeneos" (Julio, 2003) Pag 1-12.CITA03_InteropServAplic(Final).pdf

Vicente Pelechano, Marta Ruíz, Joan J. Fons, Pedro Valderas. "Desarrollo de Aplicaciones Web basadas en Servicios WEB XML. Un Caso Práctico". Pag 1-19. eecw02.pdf

Leonardo rodríguez, Andrés Vignaga, Felipe Zipitría, "Estudio de Interoperabilidad .NET/J2EE", (Noviembre, 2004), Pag 1-15. tr0208.pdf

Maria da Graca Campos Pimentel, Cesar augusto Camillo Teixeira, André Santanche,"XML: Explorando sus Aplicaciones en la Web" (Noviembre, 2000), Pag 2-42, xml1.pdf

Fernando Rodríguez, “Integración y Sistemas Legados” (Octubre, 2004), Pag 1-17, integración.pdf

Tesis Doctorado

Jorge A. Torres Jiménes Paris, "Lenguaje de Especificación de Calidad de Sericio de Componentes Dsistribuidos". Universidad Politécnica de Madrid.(Agosto, 2003). Pag 1-12. torres2.pdf

José Manuel Moya Fernández, "Cosintesis de Sistemas heterogéneos complejos", Universidad Politécnica de Madrid (Agosto, 2002). Pag 1-197. main.pdf

Paulo de Figueiredo Pires, "Webtransact: a framework for specifying and coordinating reliable web services compositions", Universidad Federal de Río de Janeiro (Abril, 2002), Pag 1-274. pires02webtransact.pdf

Artículos del ACM

Santiago Comella-Dorda, Kurt Wallnau, Robert C. Seacord, John Robert "A Survey of Legacy System Modernization Approaches" (Abril, 2002) Pag 1-30. 00tn003.pdf

Peter o'Kelly "Modelling Your Business in XML" (Septiembre, 2002) Pag 1-9. 44.pdf

M.Mecella, B.Percini. "Designing wrapper components for e-services in integrating heterogeneous systems" (Febrero, 2001). Pag 1-14. 10100002.pdf

Nicholas Routledge, Linda and Andrew Goodchild "UML and XML Schema". (Octubre, 2001). adc2002.pdfgr

Maria Wahid Chowdhury, Muhammad Zafar lqbal, "Integration of Legacy Systems in Software Architecture" (Septienbre, 2004) Pag 1-4. chowhury-Iqbal.pdf

Ali Arsanjani, Brent Hailpern, Joanne Martin, Peri Tarr, "Web Services"

David Skogan, "UML as a Schema for XML based Data Interchange". Pag 1-11. david99uml.pdf

K.Gottschalk, S.Grahan, H.Kreger, J.Shell. "Introduction to Web services architecture" (Julio, 2002). Pag 1-8. gottschallk.pdf

Michael Christoffel, Bethina Scmitt, Jurgen Schneider. "Semi-Automatic wrapper generation and adption, living with heterogeneity in a market environment". (Febrero, 2004). Pag 1-8. iceis.pdf

Harry M.Sneed & Rodolf Majnar, "A case study in software wrapping" (Noviembre, 1998) Pag 1-14. ICSM1998_cASE_sTUDY_EN_11-1998.pdf

Fei Cao, Barrett R.Bryant, Carol C.Burt, Jeffrey G.Gray, Rajeev R.Raje, Andrew M.Olson, Mikhail Auguston (Noviembre 2002). Pag 1-9. idpt03-cao.pdf

Matjaz B. Juric, Iban Rozman, Marjan Hericko, Tomaz Domajnko, "Integrating object architecture". (Octubre, 2002) Pag 1-5. IntegratingLegacy.pdf

Leonid Erlikh, "integrating legacy Systems using web Services",(agosto, 2002) Pag 1-6. IntegratingLegacyErlikh.pdf

Margot Postema, "KDD Techniques for Abstraction of Legacy Software System", (Mayo, 2004). Pag 1-9. kdd-techniques-for-abstraction.pdf

Michel Klein, Dieter Fensel, Frank Van Harmelen, Ian Horrocks, "The relation between ontologies and XML schemas"(Abril, 2004). Pag 1-13. klein00relation.pdf

Ying Zou and Kostas Kontogiannis, "Towards a Web-centric Legacy System Migration Framework", (Marzo, 2004). Pag 1-5. ncc3.pdf

Francisco Curbera, Ignacio Silva-Lepe and Sanjiva Weerawarana, "On the Integration of Heteroneneous Web Service Patners". (Agosto, 2001). Pag 1-5. on-the-integration-of-XML.pdf

Wenfei Fan, Minos Garafalakis, Ming Xiong, Cibei Jia, "Composable XML Integration Grammars", (Noviembre, 2003) Pag 1-10. p2-fan.pdf

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. “Patrones de Diseño”, 1995, pags. 139 –150

Sakid Abdul Mondal, Kingshuk Das Gupta, "Choosing a Middleware for Web-Integration of a Legacy Application", (Junio, 2002). Pag 1-4. p50-abdul.pdf

Ernesto Damiani, Sabrina De Capitani Di Vimercati, Stefano Paraboschi, Pierangela Samarati, "A Fine-Frained Access Control System for XML Docuents". (Octubre, 2003). Pag 1-34. p169-damiani.pdf

Devyani Pujari, Niral Jhaveri, Juan Carlos Guzmán, "XML middleware to services on STB" (Agosto, 2003) . Pag 1-2. p183-pujari.pdf

Xiang Fu, Tevfik Bultan, Jianwen Su, "Model Checking XML Manipulating Software", (Septiembre, 2004) . Pag 1-11. p252.pdf

Sneed Harry M.., *"Encapsulating Legacy Software for Use in Client-Server Systems"*, Working Conference in Reverse Engineering, IEEE Computer Society Press, 1996, pp.104-119.

K.Hogg,P.Chilcott, M.Nolan, B.Srinivasan, "An Evaluation of Web Services in the Design of a B2B Application", (Abril, 2002) . Pag 1-10. p331-hogg.pdf

Shankar R.Ponnekanti and armando Fox, "Interoperability Among Independently Evolving Web Services" (Noviembre, 2004), Pag 1-21. LNCS 3231-Interoperability Among Independently Evolving WebServices.pdf

Paolo Ciancarini, robert Tolksdorf, Franco Zambonelli, "Coordination Middleware for XML-centric Applications" (Diciembre, 2002). Pag 1-8. p336-ciancarini.pdf

Xiang Fu, Tevfik Bultan, Jianwen Su, "Analysis of Interacting BPEL Web Services"(Marzo, 2003) Pag 1-11. promisesf.pdf

Vinayak Borkar, "Liquid Data for WebLogic: Integrating Enterprise Data and Services"(Junio, 2004), Pag 1-2. 140_Liquid.pdf

Ralf-Dieter Schimkat, Stefan Muller, Wolfgang Kuchlin, "A Lightweight, Message-Oriented Aplication Server for the WWW" (Marzo, 2000), Pag 1-8. p934-schimbkat.pdf

Petri Vuorimaa, Teemu Ropponen, Niklas von knorring, and Mikko Honkala, "A Java Based XML Browser for Consumer Devices" (Mayo, 2002), Pag 1-6. p1094-vuorimaa.pdf

Masaichiro Yoshioka, Takuma Sudo, Akihiro Yoshikawa, Keiichi Sakata, "Legacy System Integration Technology for Legacy Application Utilization from Distributed Object Environment" (Noviembre, 2004) pag 1-7. r6_112.pdf

Margot Postema, Heinz W.Schmidt, "Reverse Engineering and Abstraction of Legacy Systems" (Mayo 2004) Pag 1-15. reverse-engineering-and-abstraction.pdf

Tanja Sollazzo, Siegfried Handschuh, Steffen Staab, Martin Frank, "Semantic web Service Architecture-Evolving Web Service standards oward the Semantic Web", (Mayo, 2004) Pag 1-7. sollazzo02semantic.pdf

E. Nesime Tatbul, "the Use of XML in Software Engineering" (Mayo, 2000) Pag 1-8. tatbul00use.pdf

Mariano Rico Almodóvar, "Interacción Persona-Agente en los Servicios Web Semánticos, Una propuesta de Sistema de intermediación", (Mayo, 2004), Pág. 1-122. TII.M.Rico.2004.pdf

Jia-Lang Seng, Wayne Tsai, "A Structured Transformation Approach for Legacy Information Systems-A Cash Receipts/Reimbursements Example" (Enero, 1999), Pág. 1-10. 00017051.pdf

Ian Gorton, Anna Liu "Architectures and Technologies for Entyerprise Application Integration" (Enero, 2004), Pag 1-2. 21630726.pdf

Libro: Michael Morrison "XML al descubierto"

Tesis Maestría

Msc. Erwin Mac Dowall Reynoso "XML Aplicada ao Gerenciamento e Automação de Sistemas de Energia Elétrica" Pós Graduação em Computação. Instituto de Computação. Universidade Federal Fluminense.

Jorge Abin de Maria. "Integración de Aplicaciones Encapsuladas para el Desarrollo de Sistemas de Información Cooperativos". Instituto de Computación. Facultad de Ingeniería. Universidad de la Republica. (Noviembre, 2002). Pág. 1-258. TesisMaestria.pdf

Otras Referencias

McKeen, J.D., Smith, H.A., "***New Developments in Practice II: Enterprise Application Integration***", Communications of the Association for Information Systems (Volume 8, 2002), 451-466.

Johannesson, P., Wangler, B., Jayaweera, P., "***Application and Process Integration – Concepts, Issues, and Research Directions***", Information Systems Engineering Symposium 2000, eds. S. Brinkkemper, E. Lindencrona, and A. Sölvberg, Springer Verlag, 2000.

Wegner, P., "***Interoperability***", ACM Computing Surveys, Vol. 28, No.1, March 1996, p285-287.

Pollock, J.T., "***The Big Issue : Interoperability vs. Integration***", EAI Journal, October 2001.

McKeen, J.D., Smith, H.A., "***New Developments in Practice II: Enterprise Application Integration***", Communications of the Association for Information Systems (Volume 8, 2002), 451-466.

Mc Naughton, G.A, Gordon, M.E., "***Common Interfaces for Enterprise Integration - Experience With NRECA's MultiSpeak Specification***", 2001 RURAL ELECTRIC POWER CONFERENCE - 45th Annual Conference, Little Rock, Arkansas April 29 – May 1, 2001.

Becker, D., Falk, H., Gillerman, J., Mauser, S., Podmore, R., Schneberger, L., "***Standards-Base Approach Integrate Utility Applications***", IEEE Computer Applications in Power, Volume 13, Number 4, October 2000.

Schreiber, R., "***Middleware Demystified***", Datamation 41, 6 (April 1, 1995) : pp 41-45.

Mylopoulus, J., Gal, A., Kontogiannis, K., Stanley, M., "***A Generic Integration Architecture for Cooperative Information Systems***", First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), June 19 - 21, 1996, Brussels, Belgium.

Bennet, K.H., "***Legacy Systems: Coping With Success***", IEEE Software, January 1995, Vol 12, No. 1, pp 19-23.

Brodie, M., Stonebraker, M., "***Migrating Legacy Systems : Gateways, Interfaces and the Incremental Approach***", Morgan Kaufmamm, San Francisco, 1995.

Bisbal, J., Lawless D., Wu, B., Grimsom, J., "***Legacy Information Systems : Issues and Directions***", IEEE Software, September-October, 1999.

Sneed H.M., Majnar, R., "***A Case Study in Software Wrapping***", in Proc. Of ICSM'98, IEEE Computer Society, 1998, pp. 86-93.

Sneed H.M., "***Encapsulating Legacy Software for Use in Clien/Server Systems***", IEEE Proceeding of WCRE'96, 1996.

Mowbray, T. Zahari, R.: *"**The Essential CORBA**"*, John Wiley & Sons, New York, 1994, p. 231-265.

Michael Christoffel, Bethina Schmitt, Jürgen Schneider: "***Semi-Automatic Wrapper Generation And Adaption***. ***Living with heterogeneity in a market environment".***