Así como los troncos y cuerdas forman una barca, la cooperación entre componentes posibilita **conseguir un servicio mayor**.

Contenidos

Principales Tablas y Figuras

Tablas

Figuras

1. Resumen descriptivo del proyecto

La razón de este primer punto es dar una visión global y sencilla sobre el proyecto que se expone en este documento, a fin de facilitar su comprensión. La mayor parte del texto aquí citado se ha seleccionado de entre las propias páginas de esta documentación.

Motivación

Este es el momento del despertar de las aplicaciones en la nube. Las aplicaciones que solemos usar están migrando para darnos servicio desde un servidor remoto. Sólo basta un simple navegador para usarlo, pudiendo acceder desde cualquier sitio y sin inconvenientes de mantenimiento por parte del cliente. Para ello es necesario que el servicio respete los estándares para que no haya problemas en la comunicación. XML-Schema del W3C, el estándar para integrar y compartir información en internet. Se hará uso exclusivo en este proyecto de él.

La implantación mundial de la Televisión Digital, además de proporcionar una mayor calidad de emisión y número de canales, ha favorecido la aparición de una serie de tecnologías, que propician la definición de nuevos modelos de negocio. Actualmente el mercado de las aplicaciones software está consolidado y en alza. Algunas compañías como Boxee box, Apple TV, Roku y ahora googleTV se abren camino en el mercado que la TD abre. Aunque no lo explotan aun totalmente, ya que sus soluciones se basan en conectar el televisor a internet mediante un SO propietario. Sin embargo, esta solución parece ser la más viable.

Las aplicaciones de GoogleTV no son ejecutables en cualquier set top box estándar, sino que son específicas de la API que Google facilita para desarrolladores. Esto hace que puedan aprovechar las anteriores aplicaciones de su SO Android, pero traban el que su mercado de TD llegue a mucha más gente. En este paper se describe una implementación open source basada en los estándares sobre la Televisión Digital Interactiva (proyecto DVB de la ETSI) de forma que cualquier decodificador de TD que soporte el estándar, funcionará con él.

Tecnología subyacente y trabajos relacionados

La ingeniería del software basada en componentes y en servicios resuelve en parte la interoperabilidad mediante el uso de mecanismos de mediación. Un Trader se puede definir como un mecanismo de mediación entre componentes software. Habilitando el que distintos componentes cooperen procesando secuencialmente unos datos iniciales para obtener una información o servicio final de una complejidad mayor a la que tiene uno solo de ellos.
Dichos componentes se encuentran en un repositorio. Es un sitio donde se almacena y mantiene información.

Descripción del proyecto

A partir de ahora al hablar sobre trader, el texto se refiere a nuestra interpretación de la especificación OMG Trader. Para comprender cómo funciona hay que saber qué es un tipo de servicio (TS). Un TS es una entidad, con propiedades, que representa a un conjunto de funcionalidades muy similares. Un TS puede heredar de uno o varios TS.

Aquí radica una parte de la potencia del trader: creación de tipos de TS a partir de otros, o lo que es lo mismo, cooperación de componentes, es decir, la mediación entre componentes para conseguir un servicio mayor por colaboración. Finalmente un componente en el trader es una "oferta de servicio". Toda oferta hereda de un TS, su representación de funcionalidad y sus propiedades. Por ejemplo, un componente-A que suma números reales, podría heredar de un TS llamado "sumador de reales". Otro B que los resta, heredaría de "restador de reales". Ambos tipos de servicio podrían unirse en otro TS llamado "calculadora simple de reales" heredando de los TS anteriores. El código en este TS estaría formado por ambos componentes dichos.
La granularidad de los TS se va definiendo conforme se crean por parte de los usuarios.

La otra parte de la potencia del trader consiste en la jerarquía entre TS. Esto es, al registrar un TS se puede especificar otro que semánticamente hablando sea de funcionalidad similar y más amplia. Por ejemplo, un desarrollador crea un componente con la funcionalidad (especializada por cualquier motivo) de sumar y restar números enteros. Lo registra en el trader bajo el TS de "calculadora simple de enteros", sería conveniente que indicara al TS "calculadora simple de reales" como superior en la jerarquía. De este modo, cuando un cliente pregunte al trader si tiene algún servicio de calculadora de enteros, si la oferta de componentes del TS "…enteros" no se puede usar porque ya no esté disponible o sea muy cara, el sistema puede ofrecer la alternativa de las ofertas disponibles del TS "…reales" con un porcentaje de certeza elevado, de que este otro tipo de servicio le puede ser útil al cliente.

Como resumen, la potencia del trader estriba en la organización de los componentes software entre los que media, a través de relacionar componentes con TS y organizar estos últimos, estableciéndose relaciones entre ellos que permiten al trader manejarlos. Por último se comprueba la interoperabilidad entre dos componentes para que sus códigos puedan ejecutarse usando la información resultante del anterior obteniendo progresivamente un resultado de mayor riqueza. La información de interoperabilidad está asociada a cada oferta y se encuentra en el repositorio IR.

Un objeto trader trabaja con tres tipos de repositorios independientes:

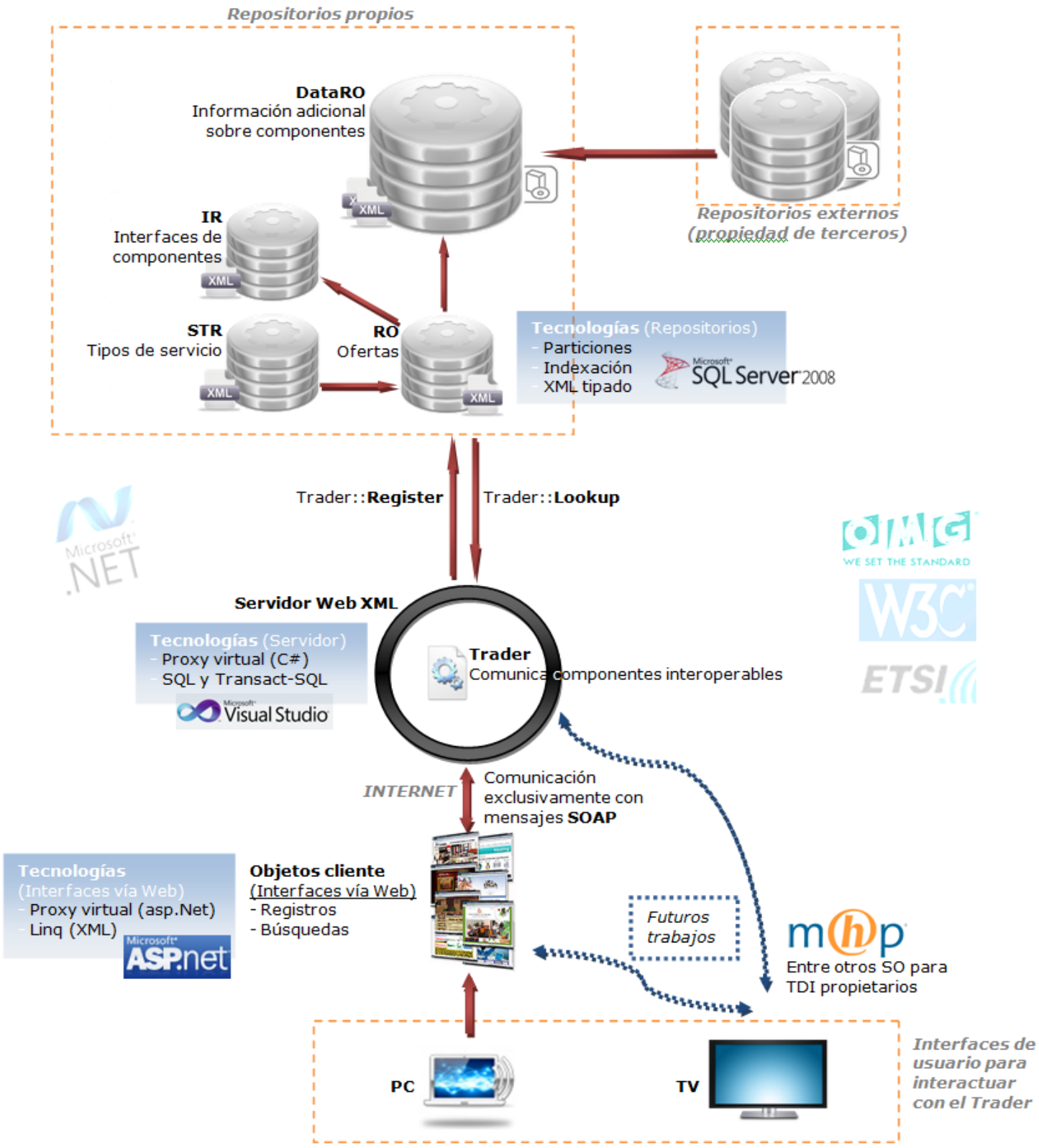
- **RO** (Offers of Repository). Depósito de las ofertas donde los desarrolladores registran sus componentes como instancias de servicios de los TS.

- **IR** (Interfaz Repository). Repositorio que almacena las interfaces de los componentes ofertados. Una capacidad o servicio exportado por un objeto cliente debe tener asociado una interfaz computacional que responde a la funcionalidad

del servicio anunciado. La especificación trader acepta la definición de interfaces en OMG IDL [1].

- **STR** (Service Type Repository). Depósito para los tipos de servicio.

Se ha añadido DataRO con la información adicional de las ofertas, que no es necesaria para el funcionamiento del trader pero si para el mercado de negocio. Esta clase de información se nombra en la especificación general sobre trader, pero no se específica donde guardarse. Por ejemplo, el propio código del componente (que puede estar en el repositorio propio o en alguno externo, ajeno a nuestro sistema), licencias de uso, etc.
(Véase el diagrama del sistema completo en la siguiente figura)



La figura 1 muestra la estructura de todo el sistema junto con las herramientas y estándares usados. Las flechas indican el flujo habitual de información.

El trader está alojado en un servidor Web XML. Usando un proxy virtual en el lado cliente y otro en el lado servidor está automatizada la encapsulación / des-encapsulación de los mensajes según la estructura de datos y metadatos, propia de un mensaje SOAP, conformando un documento XML. De forma que por internet viajan exclusivamente mensajes SOAP [2]. La comunicación entre cliente y trader se hace paso a paso por este esquema:

- **Objeto cliente**. Interfaces de usuario para usar las interfaces del trader "register" y "lookup".

- **Proxy del lado cliente** (asp.Net)
  Encapsula en mensajes SOAP el registro de ofertas y peticiones de búsqueda. Des-encapsula el conjunto ofertas resultantes de una consulta y el listado de TS registrados.

- ***Internet*** (mensajes SOAP)

- **Proxy del lado servidor** (C#) Lleva a cabo las encapsulaciones y des-encapsulaciones inversas al proxy del lado cliente.

- **Servidor Web XML**. En su interior se aloja el trader en forma de dos métodos Web principales correspondientes a sus interfaces implementadas.

- **Repositorio**. Consta de los 4 repositorios descritos más adelante. La información se almacena según el estándar XML-Schema. Pueden estar físicamente en el mismo servidor o en cualquier otro.

Gramáticas

Toda la información residirá en documentos XML dado el carácter distribuido (cliente-servidor, además con acceso a repositorios independientes) y multiplataforma (múltiples SO en la TDI) del proyecto, por lo que es necesario definir sus gramáticas que formarán la definición de nuestro modelo COTS [3] (componentes comerciales) adaptado al nuevo entorno de la TDI.

Cada repositorio está destinado a contener la información que satisfaga las necesidades del sistema: a) datos necesarios al trader, b) datos que tiene todo COTS y c) relativos a la plataforma de explotación (TDI). Se escribirán como una instancia de su gramática correspondiente. Esto es, cada componente tiene asociada unas plantillas con información donde principalmente se especifica su funcionalidad, características y calidad (RO), además de interoperabilidad para con otros componentes (IR). Este esquema se obtendrá de una investigación en trabajos previos con trader [4].

El carácter comercial de los componentes COTS ha incurrido en un estudio de los mercados más similares que se dan en la actualidad, anteriores a la aparición de la TDI, analizando 13 grandes comercios (Android Market, AppStore, Market Eclipse, Microsoft Store, etc.) bajo 20 criterios, como si se permite el acceso por distintas plataformas, si hay una optimización para cada una de ellas, si existe un comercio paralelo al de aplicaciones, si está disponible desde la TD, peculiaridades sobre el desarrollo de esas aplicaciones, etc. Puede ver la tabla comparativa en el apartado: "Tabla comparativa: Análisis de Web comerciales"

También se escribe una gramática para los TS (STR). Y otras para poder servir las oportunidades de negocio e información adicional (DataRO) como icono, imagen, código, opinión del usuario sobre las ofertas, etc.
Una vez está la estructura básica de almacenamiento, lo siguiente es conocer qué datos característicos tiene todo componente software. Para ello se investigó como se almacenan los plug-ing de Eclipse y Mozilla Firefox [5]. Extrayendo elementos como el tamaño del componente, propietario, licencia de uso, etc. Estos datos se encuentran repartidos entre las gramáticas para los repositorios RO y DataRO según el criterio de si son útiles para la búsqueda de ofertas o no, respectivamente.

Además de estudiar la nueva plataforma y sus estándares establecidos como MHP [6]. Es necesario conocer la naturaleza de los componentes entre los que el trader mediará. En este caso están destinados a ofrecer servicios en la TDI. Se ha investigado el estándar desarrollado por la ETSI, MHP, examinando principalmente su sistema de información [7], con objeto de encontrar datos útiles para la búsqueda de componentes. Entre los encontrados destacan:

- BAT permite identificar los componentes/aplicaciones destinados a un determinado emisor, canal o programa.

- Profile y versión.
  Tipo y versión del estándar MHP respectivamente, necesario para determinar la interoperabilidad entre componentes, además de conocer si un Xlet (aplicaciones MHP) está firmado [8] o no, ya que el estándar sólo permite comunicaciones entre firmados o no firmados.

Sin olvidar que el medio donde darán servicio es la televisión y ésta está sujeta a leyes de emisión. La gramática admite indicar una referencia de control parental [9]. Nuestro sistema está preparado para funcionar con el estándar MHP o cualquier otro tipo de componentes. En MHP es público el Sistema de Información, pero en sistemas privados no, la gramática se ha escrito siendo conscientes de ello. Dependiendo del SO al que pertenezca el componente se pueden almacenar unos u otros datos relativos a componentes de tal SO.

La implantación mundial de la **Televisión Digital**, además de proporcionar una mayor calidad de emisión y número de canales, ha favorecido la aparición de una serie de tecnologías, que propician la definición de nuevos modelos de negocio.

Ávidos de abordar este nuevo sector, se establece en este escrito las bases para poder llevar a cabo una incursión comercial, teniendo por objetivo primero dar soporte a componentes y aplicaciones.
En primer lugar se define un modelo de componente COTS adaptado al nuevo entorno que ha dado la TDI.

Esta especificación atiende a los tres enfoques con los que este modelo debe convivir:
Ante todo consiste en un modelo para componentes. Gran parte del estudio necesario para confeccionar el modelo se centra en conocer a fondo qué es un componente.
El carácter comercial de los componentes COTS ha incurrido en un estudio de los mercados más similares que se dan en la actualidad, mercados de gran éxito abanderados por gigantescas compañías como Google o Apple.
El último aspecto a tener en cuenta es la televisión, desde su técnica de operar en la nueva plataforma digital hasta las leyes que la rigen.

En el punto “Modelo de componente en metalenguaje para TDT” [10] se presenta el modelo detallado. Su justificación se expone en los puntos anteriores a éste bajo los tres puntos de vista comentados.

2. Estudio de la Web semántica

La Web semántica (del inglés semantic web) es la “Web de los datos”. Se basa en la idea de añadir metadatos semánticos y ontológicos a la World Wide Web. Esas informaciones adicionales —que describen el contenido, el significado y la relación de los datos— se deben proporcionar de manera formal, para que así sea posible evaluarlas automáticamente por máquinas de procesamiento. El objetivo es mejorar Internet ampliando la interoperabilidad entre los sistemas informáticos usando “agentes inteligentes”. Agentes inteligentes son programas en las computadoras que buscan información sin operadores humanos.

El precursor de la idea, Tim Berners-Lee, intentó desde el principio incluir información semántica en su creación, la World Wide Web, pero por diferentes causas no fue posible. Por ese motivo introdujo el concepto de semántica con la intención de recuperar dicha omisión.

La Web Semántica se ocuparía de resolver estas deficiencias. Para ello dispone de tecnologías de descripción de los contenidos, como RDF y OWL, además de XML, el lenguaje de marcas diseñado para describir los datos. Estas tecnologías se combinan para aportar descripciones explícitas de los recursos de la Web (ya sean estos catálogos, formularios, mapas u otro tipo de objeto documental). De esta forma el contenido queda desvelado, como los datos de una base de datos accesibles por Web, o las etiquetas inmersas en el documento (normalmente en XHTML, o directamente en XML, y las instrucciones de visualización definidas en una hoja de estilos aparte). Esas etiquetas permiten que los gestores de contenidos interpreten los documentos y realicen procesos inteligentes de captura y tratamiento de información.

Componentes de la Web Semántica

Los principales componentes de la Web Semántica son los metalenguajes (es un lenguaje que se usa para hablar acerca de otro – lenguaje objeto) y los estándares de representación XML, XML Schema, RDF, RDF Schema y OWL

- **XML** [11]
  Aporta la sintaxis superficial para los documentos estructurados, pero sin dotarles de ninguna restricción sobre el significado.

- **XML Schema = XSD** [12]
  Es un lenguaje para definir la estructura de los documentos XML.

También aunque no usados en el presente proyecto existen: RDF (Es un modelo de datos para los recursos), RDF Schema (vocabulario para describir las propiedades y las clases de los recursos) y OWL (describe la función y relación de cada uno de estos componentes)

Usabilidad

La **usabilidad** y aprovechamiento de la Web y sus recursos interconectados puede aumentar con la web semántica gracias a los documentos etiquetados con información semántica (compárese ésta con la etiqueta <meta> de HTML, usada para facilitar el trabajo de los robots) sea interpretada por el ordenador con una capacidad comparable a la del lector humano. El etiquetado puede incluir metadatos descriptivos de otros aspectos documentales o protocolarios.

2.1 Desarrollo de Software Basado en Componentes

La complejidad de los sistemas computacionales actuales nos ha llevado a buscar la reutilización del software existente. El desarrollo de software basado en componentes [13] permite **reutilizar piezas de código pre-elaborado** que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión. Al comparar la evolución del ambiente de IT con el crecimiento de las metrópolis actuales, podemos entender el origen de muchos problemas que se han presentado históricamente en la construcción de software y vislumbrar las posibles y probables soluciones que nos llevarán hacia la industrialización del software moderno utilizando tecnologías de componentes.

Los sistemas de hoy en día son cada vez más complejos, deben ser construidos en tiempo récord y deben cumplir con los estándares de calidad. Para hacer frente a esto, se concibió y perfeccionó lo que hoy conocemos como Ingeniería de Software Basada en Componentes (**ISBC**), la cual se centra en el **diseño y construcción de sistemas computacionales que utilizan componentes de software reutilizables**. Esta ciencia trabaja bajo la filosofía de "comprar, no construir", una idea que ya es común en casi todas las industrias existentes, pero relativamente nueva en lo que a la construcción de software se refiere.

Los componentes son los "ingredientes de las aplicaciones", que se combinan para llevar a cabo una tarea. Esta similitud con el resto de industrias refleja que cada componente de un aparato ha sido diseñado para acoplarse perfectamente con sus pares, las conexiones son estándar y el protocolo de comunicación está ya preestablecido. Al unirse las partes, obtenemos el servicio del aparato.

El paradigma de ensamblar componentes y escribir código para hacer que estos componentes funcionen se conoce como Desarrollo de Software Basado en Componentes. El uso de este paradigma posee algunas ventajas:

1. **Reutilización del software.** Nos lleva a alcanzar un mayor nivel de reutilización de software.
2. **Simplifica las pruebas.** Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
3. **Simplifica el mantenimiento del sistema.** Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.

4. **Mayor calidad.** Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

De la misma manera, el optar por comprar componentes de terceros en lugar de desarrollarlos, posee algunas ventajas:

1. **Ciclos de desarrollo más cortos.** La adición de una pieza dada de funcionalidad tomará días en lugar de meses ó años.
2. **Mejor ROI.** Usando correctamente esta estrategia, el retorno sobre la inversión puede ser más favorable que desarrollando los componentes uno mismo.
3. **Funcionalidad mejorada.** Para usar un componente que contenga una pieza de funcionalidad, solo se necesita entender su naturaleza, más no sus detalles internos. Así, una funcionalidad que sería impráctica de implementar en la empresa, se vuelve ahora completamente asequible.

2.1.1 Características

Aquí se listan las características clave del DSBC, las cuales serán posteriormente identificadas en este proyecto. Una de las características más importantes de los componentes es que son reutilizables. Para ello los componentes deben satisfacer como mínimo un conjunto de características. En próximos capítulos se detalla una correspondencia entre estos conceptos y su adaptación al entorno en el que se diseña nuestro componente. Las características son:

- **Identificable:** un componente debe tener una identificación clara y consistente que facilite su catalogación y búsqueda en repositorios de componentes.

- **Accesible sólo a través de su interfaz:** el componente debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación. Esta característica permite que un componente sea reemplazado por otro que implemente la misma interfaz.

- **Sus servicios son invariantes:** las operaciones que ofrece un componente, a través de su interfaz, no deben variar. La implementación de estos servicios puede ser modificada, pero no deben afectar la interfaz.

- **Documentado:** un componente debe tener una documentación adecuada que facilite su búsqueda en repositorios de componentes, evaluación, adaptación a nuevos entornos, integración con otros componentes y acceso a información de soporte.

- **Genérico:** sus servicios pueden ser usados en una gran variedad de aplicaciones.

- **Auto contenido:** es conveniente que un componente dependa lo menos posible de otros componentes para cumplir su función de forma tal que pueda ser desarrollado, probado, optimizado, utilizado, entendido y modificado individualmente.

- **Mantenido:** es deseable que un componente (como toda pieza de software) esté inmerso en un proceso de mejoramiento continuo que le garantice al integrador nuevas versiones que incluyan correctivos, optimizaciones y nuevas características. Esto contribuye a que dicho componente sea seleccionado con mayor frecuencia para formar parte de sistemas de software.

- **Independiente de la plataforma (hardware y sistema operativo), del lenguaje de programación y de las herramientas de desarrollo:** existen diversas plataformas de cómputo de uso frecuente (Windows/Intel, Solaris/Sparc, OSX/PPC, Linux/Intel) y es deseable que un componente pueda ejecutarse en todas ellas.
  Asimismo, ya que existe una amplia gama de lenguajes de programación y herramientas de desarrollo, es natural que encontremos componentes escritos empleando lenguajes y herramientas de la preferencia del programador, por lo tanto es deseable que dichas preferencias no limiten el uso de los componentes.

- **Puede ser reutilizado dinámicamente:** puede ser cargado en tiempo de ejecución en una aplicación.

- **Certificado:** el componente puede ser certificado por una agencia de software independiente o mediante la aplicación de modelos de auto-certificación que le permiten al comprador del componente determinar la calidad del software adquirido.

- **Accedido uniformemente sin importar su localidad**: la forma de invocar los servicios ofrecidos por los componentes debiese ser independiente de su ubicación (local o remota). Para ello el modelo de componentes debería estar basado en tecnologías de procesamiento distribuido tales como CORBA, RMI y .NET Remoting.

2.1.2 Mecanismos

Mecanismos de composición del software. Bajo el modelo de desarrollo de software basado en componentes, las nuevas aplicaciones se construyen mediante la integración o composición de componentes. Se define la composición de software como “el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de

sus interfaces (de composición)". Nótese que se hace especial énfasis en las interfaces como elementos fundamentales para lograr la composición de componentes. La composición puede concebirse como una relación cliente-servidor entre dos componentes. El componente cliente solicita un servicio (operación) del componente servidor, el cual ejecuta la operación solicitada y devuelve los resultados al cliente. El servidor produce un resultado que es consumido por el cliente.

Además de los componentes, los *frameworks* (la infraestructura de soporte requerida) también se consideran entidades sujetas a composición. En consecuencia, existen tres clases principales de interacción en los sistemas basados en componentes:

- Componente-Componente (C-C): permite la interacción entre componentes. De este tipo de interacción se obtiene la funcionalidad de la aplicación, de forma tal que los contratos que especifican este tipo de interacción pueden ser clasificados como Contratos a Nivel de Aplicación.

- Componente-*Framework* (C-F): posibilita las interacciones entre el *framework* y sus componentes. Dicha interacción permite que el *framework* administre los recursos de los componentes. Los contratos que especifican estas interacciones pueden ser clasificados como Contratos a Nivel de Sistema.

- *Framework*-*Framework* (F-F): posibilita las interacciones entre *framewoks* y permiten la composición de componentes desplegados en *framewoks* heterogéneos. Estos contratos puede ser clasificados como Contratos de Interoperabilidad.

2.2 SOA

SOA. Alineación de la TI (tecnologías de la información) a los negocios y aumento de la agilidad empresarial.

La *Arquitectura Orientada a Servicios* (**SOA**) es un estilo arquitectónico de TI que soporta la transformación de su empresa en un conjunto de servicios vinculados o tareas empresariales repetibles a las cuales se puede acceder en una red cuando sea necesario. Puede ser una red local, Internet o bien una red geográfica y tecnológicamente distinta, que combina servicios en Nueva York, Londres y Hong Kong, aunque estén todos instalados en su desktop local. Esos servicios pueden combinarse para realizar una tarea empresarial específica, para permitir que la empresa se adapte a condiciones y requisitos cambiantes. En el caso que nos ocupa, estos servicios tienen la forma de componentes software que se

combinarán para dar un servicio específico. Su contexto es la TDT, un campo nuevo para SOA.

Cuando la implementación de SOA se guía por objetivos estratégicos empresariales, se asegura la transformación positiva de la empresa. Los beneficios de su uso son:

- Alineación de la TI a los negocios
- Reutilización máxima de los activos de TI

Juntos, esos beneficios ayudan a asegurar que la inversión en proyectos costosos de TI resulte en un valor duradero para la empresa.

La SOA describe un sistema completo de servicios que se buscan dinámicamente los unos a los otros, se unen para realizar alguna aplicación y se recombinan de varias formas.

2.3 Servicios Web

El término "servicios Web" designa una tecnología que permite que las aplicaciones se comuniquen en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio Web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML estandarizada [14]. Usa protocolos basados en el lenguaje XML con el objetivo de describir una operación para ejecutar o datos para intercambiar con otro servicio Web. Un grupo de servicios Web que interactúa de esa forma define la aplicación de un servicio Web específico en una **arquitectura orientada a servicios (SOA).**

La principal razón para usar servicios Web es que se basan en HTTP sobre **TCP** (Transmission Control Protocol) en el puerto 80. Dado que las organizaciones protegen sus redes mediante *firewalls* -que filtran y bloquean gran parte del tráfico de Internet-, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web utilizan este puerto, por la simple razón de que no resultan bloqueados.
Otra razón es que, antes de que existiera **SOAP**, no había buenas interfaces para acceder a las funcionalidades de otros ordenadores en red. Las que había eran *ad hoc* y poco conocidas, tales como **EDI** (Electronic Data Interchange), **RPC** (Remote Procedure Call), u otras APIs.
Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad

será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más utilizada.
Se espera que para los próximos años mejoren la calidad y cantidad de servicios ofrecidos basados en los nuevos estándares.

Los servicios Web usan XML, que puede describir cualquier tipo de dato en una forma independiente de la plataforma para el intercambio entre sistemas, lo que permite el movimiento hacia aplicaciones "flojamente" acopladas. Además, los servicios Web pueden funcionar a un nivel más abstracto que puede reevaluar, modificar o manejar tipos de datos dinámicamente on demand (bajo demanda). Por tanto, en términos técnicos, los servicios Web pueden manejar datos con mucha más facilidad y permiten una comunicación más libre entre los software.

En términos conceptuales más elevados, es posible ver los servicios Web como unidades de trabajo, donde cada una maneja una tarea funcional específica. Las tareas se pueden combinar en tareas orientadas a negocios para manejar tareas operacionales empresariales específicas, esto permite que el personal no-técnico de la empresa piense en aplicaciones que pueden manejar temas empresariales en conjunto en un flujo de trabajo de aplicaciones de servicios Web. Así, una vez que el personal técnico haya diseñado y construido los servicios Web, los arquitectos de procesos empresariales pueden agregarlos para resolver problemas en el ámbito empresarial.

Por tanto, los principales problemas que los servicios Web tratan de resolver son los temas de **integración** de datos y aplicaciones **y** de la **transformación de funciones técnicas en** tareas informáticas **orientadas a negocios**. Esos dos aspectos permiten que las empresas se comuniquen con sus socios en el ámbito de los procesos o aplicaciones y que, al mismo tiempo, dejen un espacio dinámico para adaptarse a nuevas situaciones o trabajar con otros socios mediante solicitud.

2.3.1 Estándares y otras tecnologías W3C

El entramado de tecnologías implicadas en la gestión de servicios Web está aquí representada en la figura 2

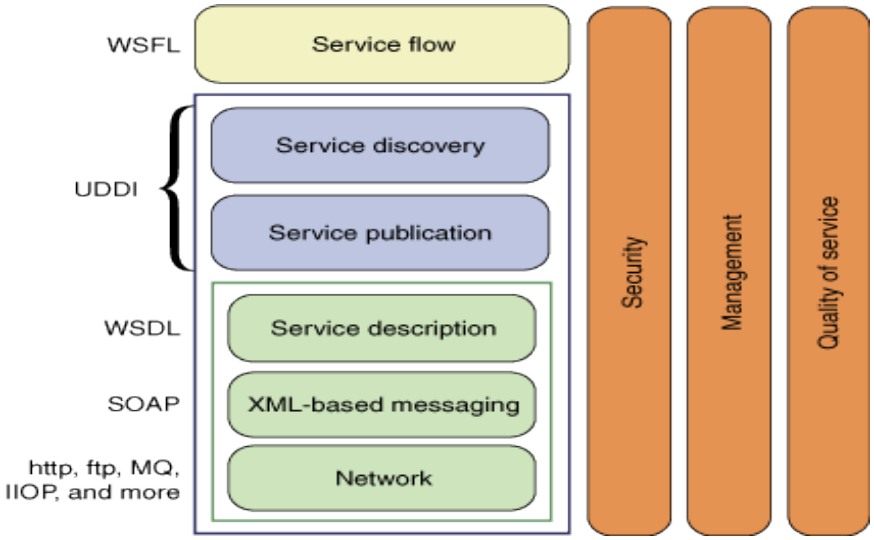


Figura 2: ibm.com

A modo de resumen los principales protocolos y lenguajes involucrados [15] son:

- **Web Services Protocol Stack**
  Así se denomina al conjunto de servicios y protocolos de los servicios Web.

- **XML** (Extensible Markup Language)
  XML (Extensible Markup Language) es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Está diseñado para almacenar de forma estructurada y transportar datos. En el presente proyecto cada componente tiene asociada una plantilla que lo identifica y describe.

  Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).

  No ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

- **SOAP** es un protocolo estándar que define **cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML**. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC

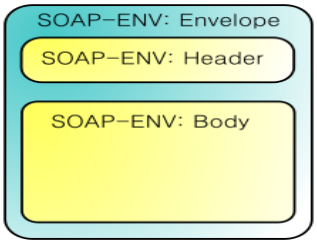


Figura 3

- **WSDL** (Web Services Description Language)
  Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web. No ha sido implementado por requisitos de tiempo.
  En esta línea se encuentran las especificaciones de las interfaces de los propios componentes

- **UDDI** (Universal Description, Discovery and Integration) [16]
  Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles. Cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL.

Los servicios Web son, principalmente, una tecnología de integración independiente de la forma. Las tecnologías de componentes para servicios Web son definidas de forma uniforme e interactúan en XML, según se mencionó anteriormente. Dado que el propio XML es independiente del lenguaje, los servicios Web también lo son. Por tanto, los servicios Web se pueden desarrollar en varios lenguajes de programación, como Java, Python, Perl, C#, Basic y otros.

El concepto de los servicios Web pretende mejorar la interacción entre las aplicaciones Web y la propia arquitectura de Internet.

ODP

El Open Directory Project [17] es el directorio editado por personas, más extenso y más completo del Web. Su construcción y mantenimiento son realizados por una gran comunidad global de editores voluntarios.

El Web continúa creciendo a un ritmo sin precedentes. Los buscadores automatizados son cada vez menos capaces de entregar resultados útiles a sus usuarios. Los pequeños grupos de editores contratados por los directorios comerciales no pueden mantenerse al día catalogando sitios, y la calidad y cantidad de sus índices se han visto deterioradas. Se están llenando de enlaces muertos, y no pueden mantener el ritmo de crecimiento de Internet.
El Open Directory proporciona los medios para que Internet se organice a sí misma. Conforme Internet crece, crece también el número de personas que la usan. Cada una de estas personas puede organizar una pequeña porción del web y presentarla al resto de la población, filtrando lo malo e inútil y conservando sólo los mejores sitios.

Principalmente su clasificación se basa en 3 pilares basados en la descripción de la funcionalidad en términos de atributos y métodos. Éstos son el nombre (identifica la propiedad), tipo de valor (establece el tipo de valores permitidos de la propiedad, por ejemplo, un tipo) y modo (indica si es una propiedad de sólo lectura o lectura-escritura, si es opcional u obligatoria). Además las propiedades pueden ser declaradas como dinámicas, esto es, su valor puede ser actualizado.

Esta clase de servicio ODP es normalmente usado en trader. Aunque la taxonomía que utiliza para la organización de los recursos no es suficiente para el nuevo entorno de servicios basado en componentes para la plataforma TDT. Más adelante se explica el desarrollo de una nueva definición que permita describir los COTS debidamente.

2.3.2 Uso de Servicios Web en nuestra aplicación

Existen varias formas de plantear los servicios Web al crear aplicaciones.

- En el nivel más básico, es una familia de protocolos avanzados de comunicaciones que permite que las aplicaciones se comuniquen. Ese nivel ha progresado mucho en los años recientes con muchas herramientas que permiten que los desarrolladores de software escriban servicios Web que interactúan y creen aplicaciones complejas. Ese nivel frecuentemente se caracteriza por interacciones directas uno a uno entre los servicios o por relativamente pocos servicios que interactúan los unos con los otros.

- Sin embargo, el uso de los servicios Web sólo como un protocolo de comunicaciones no representa su verdadera capacidad: la capacidad de la arquitectura orientada a servicios (SOA).

  La SOA describe un sistema completo de servicios que se buscan dinámicamente los unos a los otros, se unen para realizar alguna aplicación y se recombinan de varias formas. Ese modelo fomenta la reutilización de la tecnología y de software, lo que

produce una evolución en la forma de diseñar, desarrollar y poner en uso las aplicaciones. Acerca a la realidad el mundo de la informática distribuida.

En ese nivel, los desarrolladores de software necesitan pensar en el modelo SOA y diseñar su aplicación distribuida con ese modelo. Este nivel se caracteriza por el uso de tecnologías para permitir las comunicaciones distribuidas de los servicios, como el uso del bus de servicios empresariales (**ESB**), que es una red de distribución común para el trabajo con los servicios.

- El nivel más alto es considerar ese modelo SOA y los varios servicios que lo componen como elementos constructivos, que pueden ser montados en secciones enteras dentro de aplicaciones completas, en lugar del método convencional de escribir el código línea por línea.

  Al examinar las interfaces que se conectan, es posible construir aplicaciones enteras sin escribir código. En realidad, el código directo puede incluso perjudicar, ya que los servicios se pueden escribir en varias plataformas y lenguajes diferentes. Es posible unir los bloques en un flujo de trabajo de operaciones que define el rendimiento de la aplicación, y se pueden usar otras herramientas para supervisar la eficiencia del flujo de trabajo de cada servicio o grupo de servicios.

  En este nivel, los desarrolladores pueden abandonar los lenguajes de programación regulares y trabajar en una arquitectura basada en modelos que les ayuda a crear aplicaciones con más exactitud respecto al diseño. Puesto que los paradigmas entre el diseño del servicio y su implementación son más semejantes. Después, se ejecuta ese diseño sobre la base de un sistema distribuido como un ESB.

3. Estudio previo a la definición de COTS

En un proyecto de esta índole, es clave para su funcionamiento una base tecnológica sólida. Esto se consigue con el uso de estándares que posibiliten la interoperabilidad entre sistemas internos y externos. Además de disponer de un lenguaje que permita una especificación estricta y global de todas características necesarias del proyecto.

El Consorcio World Wide Web (W3C) desarrolla tecnologías inter-operativas (especificaciones, líneas maestras, software y herramientas) para guiar la Red a su potencialidad máxima a modo de foro de información, comercio, comunicación y conocimiento colectivo.

Para que la Web alcance su máximo potencial, las tecnologías Web más importantes deben ser compatibles entre sí y permitir que cualquier hardware y software, utilizado para acceder a la Web, funcione conjuntamente. El W3C hace referencia a este objetivo como "interoperabilidad Web". Al publicar estándares abiertos (no propietarios) para lenguajes Web y protocolos, el W3C trata de evitar la fragmentación del mercado y, por lo tanto, la fragmentación de la Web.

Una vez que se conoce la base tecnológica necesaria, se debe apoyar el conocimiento sobre el proyecto en los trabajos anteriores en la materia. Bajo este aspecto, he seguido el trabajo realizado por el grupo de investigación Informática Aplicada de la Universidad de Almería que junto a la universidad de Málaga comenzó en el 2001 una especificación general sobre conceptos claves en este proyecto como son: el COTS, la interoperabilidad web de un repositorio, la gestión interna del repositorio según un trader, etc.

Así como en el modelo Web actual existen comercios de gran relevancia empresarial, que han demostrado su viabilidad y buen funcionamiento, abriendo un nuevo y lucrativo mercado en lo que a software se refiere. De modo, que este proyecto también se nutre de estos trabajos previos teniendo en cuenta y estudiando desde un punto de vista analítico y crítico sus representaciones de cara al cliente. Más adelante se encuentra una tabla comparativa de los comercios Web de componentes software más conocidos y de mayor éxito comercial que actualmente marcan tendencia en el sector.

Según recorra esta memoria se analizarán las características que diferencian a este proyecto de los ya realizados en el mercado.

A continuación se detalla estos estudios previos en tecnologías consolidadas según estándares internacionales y trabajos previos en la materia, tanto propios – por la Universidad de Almería – como ajenos.

3.1 MHP

Multimedia Home Platform es un sistema intermediario (*middleware*) abierto, diseñado por el proyecto DVB* y estandarizado por la ETSI*[2]. Define una plataforma común para las aplicaciones interactivas de la televisión digital, independiente tanto del proveedor de servicios interactivos como del receptor de televisión utilizado. De este modo, favorece la creación de un mercado horizontal donde aplicaciones, red de transmisión y terminales MHP pueden ser suministrados por proveedores o fabricantes independientes.

* **Digital Video Broadcasting** (**DVB**) es una organización que promueve estándares aceptados internacionalmente de televisión digital.

*[2] **European Telecommunications Standards Institute** es una organización de estandarización de la industria de las telecomunicaciones (fabricantes de equipos y operadores de redes) de Europa, con proyección mundial.

GEM

Globally Executable MHP, consiste en un subconjunto de MHP que ha sido diseñado teniendo en cuenta las diferentes posibles implementaciones del mismo por diferentes standards middleware.

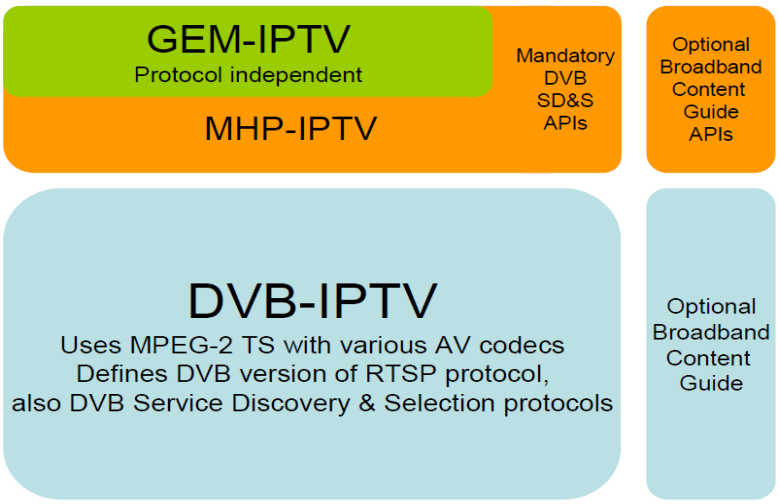


Figura 4: MHP/GEM 1.2 profiles que incluyen las configuraciones para IPTV.

Es un framework orientado a permitir a las distintas organizaciones trabajar en armonía en cuanto a especificaciones técnicas, como por ejemplo la elección de un único runtime de

ejecución y un único conjunto de APIs. El objetivo es que tanto aplicaciones como contenido funcionen en todas las plataformas basadas en GEM.

En resumen esta especificación lista aquellas partes de la especificación MHP que se entiende son específicas de la tecnología o el mercado de DVB. GEM permite la sustitución de estas allí donde sea necesario siempre que la nueva tecnología sea funcionalmente equivalente a la original. [18]

3.2 Qué es y cómo funciona un complemento

Concepto y taxonomías

Un **complemento** es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API. También se lo conoce como **plug-in** (del inglés "enchufable"), **add-on** (agregado), **complemento**, **conector** o **extensión**.
Los complementos permiten:

- A los desarrolladores externos colaborar con la aplicación principal extendiendo sus funciones
- Reducir el tamaño de la aplicación
- Separar el código fuente de la aplicación a causa de la incompatibilidad de las licencias de software

Algunos tipos de aplicaciones que suelen incluir complementos son:

- **Navegadores web**. Es frecuente requerir ciertos complementos que amplían las funciones de las páginas web para ver contenidos interactivos, videos y cosas similares. Un ejemplo conocido es Flash de Adobe un complemento que carga animaciones multimedia interactivas y se usa, por ejemplo, para ver videos.

- **Reproductores de música**. Algunos permiten añadir complementos para reproducir formatos que no son soportados originalmente, producir efectos de sonido o video, mostrar animaciones o visualizaciones que se mueven de acuerdo a la música que se está escuchando, entre otras opciones.

- **Sistemas de gestión de contenidos**. Permiten cambiar la apariencia, añadir botones u otro tipo de contenido a las páginas web que generan.

Y en general, cualquier aplicación puede añadir soporte para complementos.
El primer complemento se diseñó en 1987 para el programa HyperCard de Macintosh.

Funcionamiento mediante APIs

La aplicación principal o *host* proporciona servicios que el complemento puede utilizar, incluyendo un método para que los complementos se registren a sí mismos y un protocolo para el intercambio de datos. Los complementos dependen de los servicios prestados por la aplicación de acogida y no suelen funcionar por sí mismos. Por el contrario, la aplicación principal funciona independientemente de ellos, lo que permite a los usuarios finales añadir y actualizar los complementos de forma dinámica sin necesidad de hacer cambios a la aplicación principal.

Las interfaces de programación de aplicaciones (**APIs**) proporcionan una interfaz estándar, lo que permite a terceros crear complementos que interactúan con la aplicación principal. Un API estable permite que complementos de terceros funcionen como la versión original y amplíen el ciclo de vida de las aplicaciones obsoletas.

Ejemplos de complementos y APIs

El API para complementos de Adobe Photoshop y After Effects se ha convertido en un estándar y las aplicaciones de la competencia como Corel Paint Shop Pro lo han adoptado hasta cierto punto. Otros ejemplos de la APIs son, entre otros, VST y Audio Units.

Las arquitecturas de numerosos juegos y aplicaciones suelen utilizar complementos que permiten a los editores, ya sean los creadores originales o terceros, agregar funcionalidad al software. La serie Microsoft Flight Simulator ha llegado a ser bien conocida por sus complementos de aviones.

Diferencias entre complemento y servicio Web

Los complementos se descargan en el sistema base y allí realiza su función. El servicio Web se ejecuta en la nube por petición, enviándose una entrada y recibiendo la salida con valor de servicio.

Carácter marcadamente colaborador de los servicios Web: Los plugin se diseñan para ser un complemento funcional a un sistema base principal aunque también pueden interactuar entre ellos, incluso necesitar unos de otros. Los servicios Web se diseñan para ser componentes autónomos y además poder combinarse ofreciendo un servicio complejo a partir de otros simples.

3.3 Estudio de trabajos previos

3.3.1. Trading for COTS Components in Open Environments

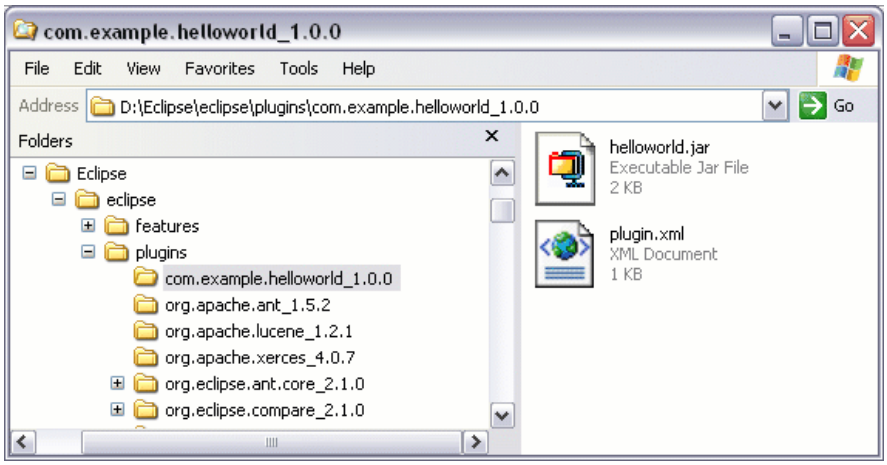
Artículo: Trading for COTS Components in Open Environments
Universidades de Almería y Málaga
http://www.lcc.uma.es/~av/Publicaciones/04/CJ-trader.pdf

3.3.2. Interactuación entre complementos y sistemas base

a. Eclipse

Para instalar plugins [5] en eclipse hay dos formas, dependiendo del plugin concreto. En algunos plugins se facilita una url (dirección web) para que el programa lo instale.

Otros plugins son simplemente un zip. Al descomprimir el zip tiene dentro un directorio **plugins** y otro **features**. En el sitio que tenemos instalado eclipse hay otros dos subdirectorios con los mismos nombres. Basta copiar, con eclipse cerrado, los contenidos de los respectivos directorios, cada uno en su sitio, y arrancar eclipse.



Configuracion del plug-in

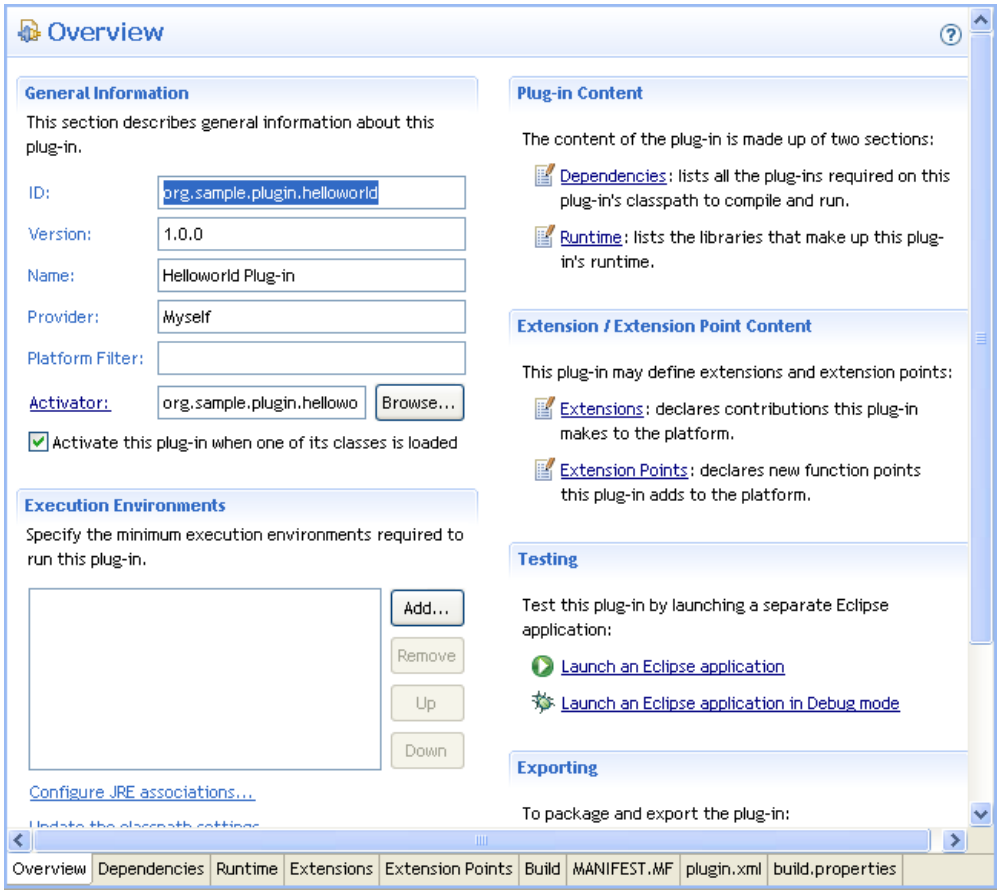
La información almacenada en este archivo le dice a Eclipse **cuándo y dónde** utilizar los nuevos puntos de vista, perspectivas, asistentes, diálogos, etc. que se creó en un plug-in.

Plugin.xml es un editor de varias páginas.

Las primeras seis páginas del editor (información general, dependencias, tiempo de ejecución, las extensiones, los centros de extensión y construcción) tienen por objeto orientar al usuario en la creación de los recursos que quiera con una interfaz de usuario en lugar de directamente editando el XML.

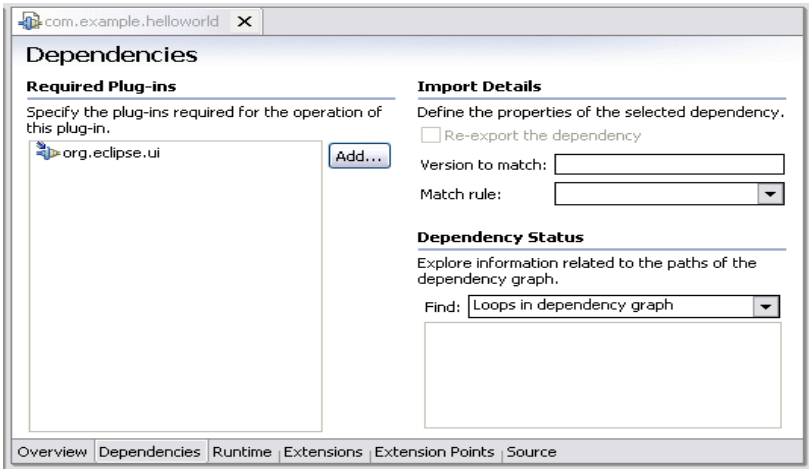
Es recomendable visitar las fichas de la interfaz de usuario proporcionada en lugar de editar el código XML.

- **Overview (Resumen)**
  Esta pestaña proporciona información general y detalles. Hay enlaces para modificar el contenido del plug-in, la forma de ejecución y prueba, y de cómo implementarlo.



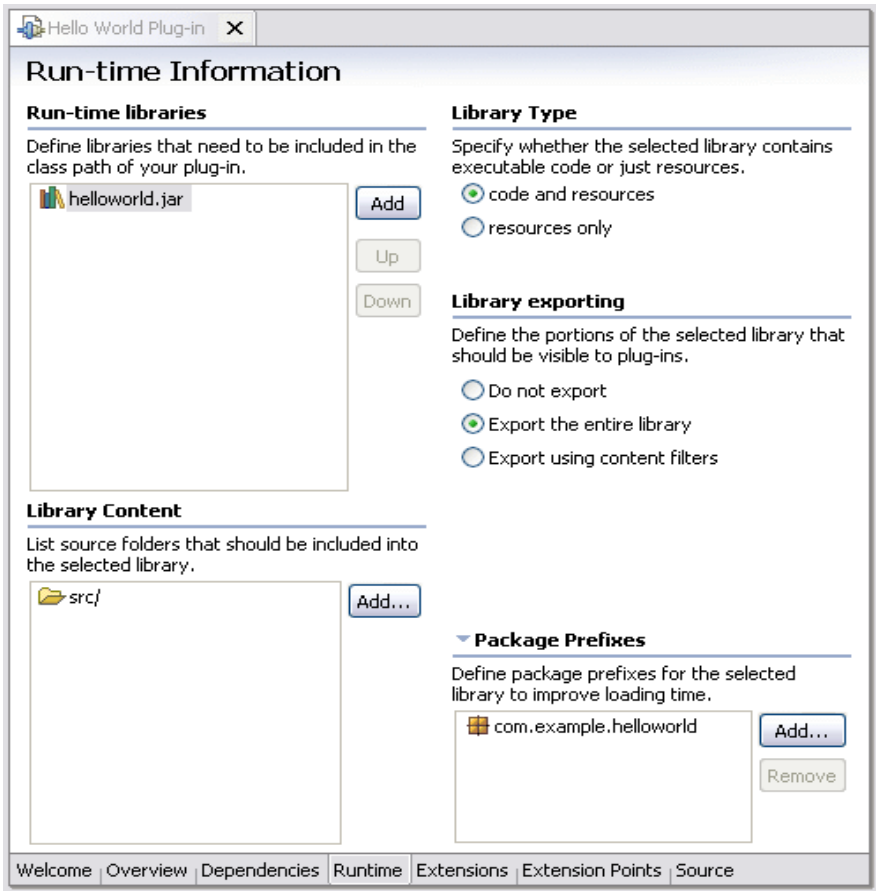
- **Dependencies (Dependencias)**
  Esta pestaña muestra los plug-ins de Eclipse de los que su plug-in depende para ejecutarse. Por defecto tendrá org.eclipse.ui y org.eclipse.core.runtime ya que estos dos plug-ins de Eclipse son el mínimo necesario para ejecutar un plug-in de Eclipse (que son el núcleo del código de Eclipse). Se pueden modificar las

propiedades y analizar las dependencias (por ejemplo, ver si hay ciclos de dependencia, de manera que pueda minimizar la lista).



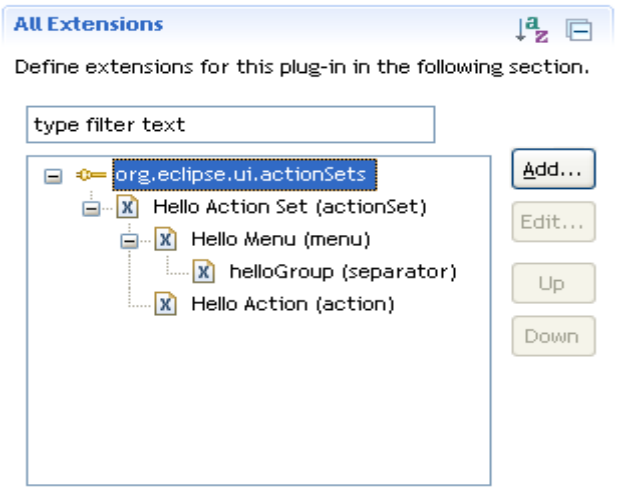
- **Runtime (Tiempo de ejecución)**

  Esta pestaña proporciona información de tiempo acerca del plug-in, en particular, las bibliotecas deben estar accesibles cuando el plug-in se ejecuta. Por lo tanto cualquier biblioteca que necesite (como bibliotecas SQL) debería ser incluida en esta ficha.

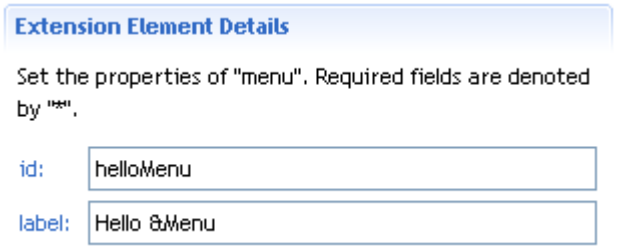


- **Extensions (Extensiones)**

Esta ficha proporciona una lista de extensiones para Eclipse que su plug-in crea. Algunos ejemplos de extensiones son una nueva perspectiva, asistente de diálogo, etc. Se pueden agregar más extensiones.

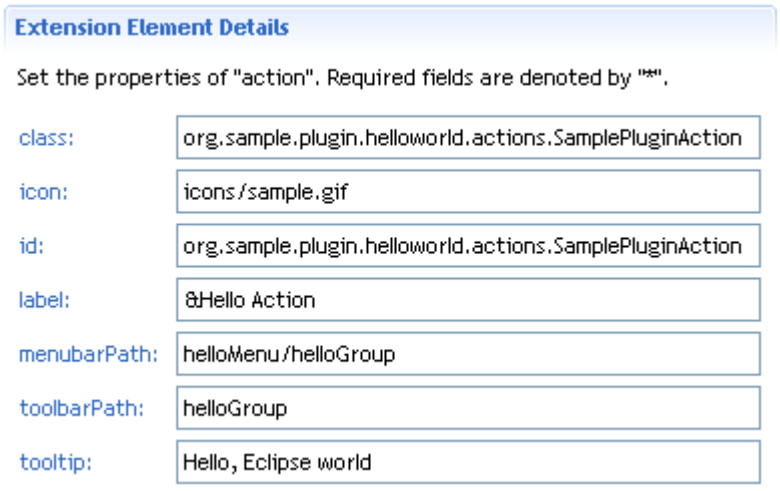


También se puede definir aquí el nombre del plugin por ejemplo.



- **Extension Points (Puntos de Extensión)**

  Esta ficha posibilita extender la funcionalidad actual de las características de Eclipse. Por ejemplo, si desea agregar un botón, su acción o para ver una perspectiva que ya existe en Eclipse que se extendería a través de su punto de extensión.



- **Build (Cuerpo)**
  Determina lo que se incluye en junto con el plug-in. Mínimo el archivo plugin.xml, así como el archivo jar del plug-in de archivos de clase, así como bibliotecas. También se puede incluir otro tipo de información de pruebas, documentación, código fuente, etc.

- **Plugin.xml**
  Esta pestaña contiene el código XML de su plug-in. Su manipulación ha de ser cuidadosa para que case con la especificación esperada por eclipse.

- **Build.properties**
  Esta ficha contiene la versión de texto de la construcción de propiedades que utiliza Eclipse para construir el proyecto. Es recomendable editar las propiedades a través de la construcción en el separador Creador y no a través de esta ficha.

  El plug-in recién configurado puede ser ejecutado sin necesidad de exportarlo e instalarlo. Para hacerlo active la solapa "Overview" y ejecute la opción "Launch an Eclipse application" que se encuentra en el marco llamado "Testing"

En el caso de MOZILLA se detallan un mayor número de parámetros que coinciden con éstos y los amplían. Al ser más completa se usará como base para establecer una correspondencia entre los trabajos previos y la actual especificación

b. Mozilla

Aquí se extraen los pasos para desarrollar una extensión básica para Mozilla, en concreto el preparado necesario del complemento para su compatibilidad con el sistema base. Esta información nos será útil para confeccionar la especificación de COTS de nuestro proyecto, ya que un servicio Web y los complementos tienen algunas semejanzas. Como ya hemos visto en sus definiciones, ambos necesitan interactuar con otro software, ya sea un sistema base en el caso de los complementos u otros componentes en el caso de servicios Web.

Pero veamos primero los tipos de complementos que existen en Mozilla.
Los complementos (agregados) son pequeños programas que modifican o añaden características a la apariencia o funcionalidad de Firefox.
Hay tres tipos de complementos: extensiones, temas y plugins:

- **Extensiones**: Las extensiones añaden nuevas funcionalidades a Firefox. Pueden añadir distintas cosas, desde un botón en la barra de herramientas hasta una característica completamente nueva. Las extensiones permiten que la aplicación que será personalizada se ajuste a las necesidades de cada usuario, y al mismo tiempo, minimizan el tamaño y la apariencia de la aplicación misma.

- **Temas**: Los temas modifican la apariencia de Firefox.

- **Plugins**: los Plugins ayudan a que Firefox realice funciones específicas, como por ejemplo visualizar formatos gráficos especiales o reproducir archivos multimedia. Los Plugins son levemente diferentes de las extensiones, las cuales modifican o añaden alguna característica a las funcionalidades existentes. El objetivo característico de un plugin es añadir herramientas multimedia, tales como la posibilidad de insertar y reproducir videos en una página web.

Preparando el Entorno de Desarrollo para un plug-in en Mozilla
Las extensiones se distribuyen en archivos comprimidos en formato ZIP, o en paquetes, con extensión xpi (*se pronuncia "zippy"*). Los archivos XPI contienen el siguiente código:

```
extensión.xpi:
        /install.rdf
        /components/*
        /components/cmdline.js
        /defaults/
        /defaults/preferences/*.js
        /plugins/*
        /chrome.manifest
        /chrome/icons/default/*
        /chrome/
        /chrome/content/
```

En el fichero install.rdf se encuentra el manifiesto de instalación. Un manifiesto de instalador es un archivo agregado que la aplicación XUL (Administrador-Habilitado) usa para determinar la información acerca de como un *agregado* ó *complemento* (Add-on) ha de ser instalado. En el caso de servicios

Web, podríamos extrapolar algunas características para determinar la información de cómo un servicio es usado. Este aspecto ya está contemplado en trabajos previos como el primer artículo de este capítulo en forma de "Service Access Protocol" principalmente, pero investigaremos como llevan años funcionando y perfeccionándose los plug-in para estudiar si pueden aportar algo interesante a nuestra especificación. Respecto a **XUL** es el lenguaje XML para interfaces de usuario de Mozilla que no es relevante aquí.

Este archivo contiene *metadatos* que identifican el complemento. Y provee información de su proveedor, donde se puede hallar más información, con cuales versiones de qué aplicaciones es compatible, como debería ser actualizado y cosas por el estilo.
El formato de un manifiesto de instalador es RDF/XML, debe ser llamado install.rdf y ubicarse en el nivel superior del archivo XPI de un complemento.

El esqueleto de un manifiesto sería:

```
<?xml version="1.0"?>

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:em="http://www.mozilla.org/2004/em-rdf#">
 <Description about="urn:mozilla:install-manifest">
  <!-- properties -->
 </Description>
</RDF>
```

Las propiedades pueden ser necesarias u opcionales y con cadenas simples o complejas.

Propiedades requeridas

Todo manifiesto las especifica correctamente para poder ser instalado

Id Identificador del complemento

Versión Indica la versión del complemento que está siendo proveído.

Tipo Representa el tipo de complemento: extensiones, temas, plugin, etc. (Mozilla usa un entero)

Aplicación destino (targetApplication)

Especifica una aplicación a la cual se dirige este complemento. Esto significa que el complemento trabajará con la aplicación definida por la propiedad id (<em:id>) especificada (para un rango de IDs de aplicaciones y sus valores máximo y mínimo (min/maxVersions)

Esto permite al autor del plugin especificar, qué complementos han sido probados con qué versiones de Firefox.

Ej. Las extensiones compatibles con Firefox 3.5 deberán especificar un *maxVersion* de 3.5 *, por lo que son compatibles con actualizaciones de seguridad de estabilidad.

Nombre Nombre del complemento

Ejemplos

id

```
<em:id>miextension@misitio.com</em:id>
<em:id>{daf44bf7-a45e-4450-979c-91cf07434c3d}</em:id>
```

Versión

```
<em:version>2.0</em:version>
<em:version>1.0.2</em:version>
<em:version>0.4.1.2005090112</em:version>
```

Tipo

```
<em:type>2</em:type>
```

Aplicación Destino

```
<em:targetApplication>
 <Description>
  <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
  <em:minVersion>1.5</em:minVersion>
  <em:maxVersion>2.0.0.*</em:maxVersion>
 </Description>
</em:targetApplication>
```

Nombre

```
<em:name>Mi Extension</em:name>
```

Propiedades opcionales

Dependiendo de las capacidades del componente es conveniente indicar estas propiedades.

Localización (Localized)

Engloba la información del nombre del complemento, descripción, contribuidores y otros metadatos.

- **name** (nombre)
- **description** (descripción de su funcionalidad)
- **creator** (creador)
- **homepageURL** (URL de la página de inicio)
- **developer** (desarrollador)
- **translator** (traductor. Si está traducida)
- **contributor** (contribuidores adicionales)

Actualización URL (updateURL)

Es un enlace a un archivo de manifiesto para actualización personalizada que especifica la disponibilidad de actualizaciones para el complemento. Si está activado el administrador del complemento periódicamente verificará con este archivo de manifiesto si hay disponibles versiones más recientes.

Seguridad. Se recomienda encarecidamente que el updateURL sea un enlace HTTPS (seguro). Si no es seguro, las updateURL pueden ser secuestradas por un archivo malicioso- update.rdf- permitiendo a malware (código dañino) infiltrarse en la computadora de los usuarios, en tal caso se debe establecer una updateKey.
También es posible delegar las actualizaciones a la Web oficial de Mozilla conocida informalmente como AMO.

Para complementos alojados en addons.mozilla.org no se usa el campo *updateURL*. Por defecto, las aplicaciones Mozilla que usan el administrador de complementos (tales como Firefox y Thunderbird) enviarán peticiones de actualización a addons.mozilla.org usando el servicio web predeterminado. Cada vez que el proveedor suba o publique una nueva versión de su complemento, o cambie los parámetros de compatibilidad mediante la interfaz de autor, su manifiesto de actualización será generado automáticamente.

Este punto es importante para el proyecto principal, ya que el servicio de alojamiento en un repositorio propio conlleva una oportunidad de negocio que se sustenta en servicios como esta ventaja de actualización segura y automática.

Actualmente Apple (tomada como ejemplo de empresa de comercio electrónico entre otras) tiene en una tienda donde se publican aplicaciones (App Store) Los comerciantes deben subir a sus servidores las aplicaciones, así la ubicación física es siempre la de la tienda.

Google sigue la misma forma de operar, no así Softonic.

Opciones URL (optionsURL)

Permite especificar las opciones de la extensión (útil sólo para extensiones)

Acerca de URL (aboutURL)

Permite especificar el campo "acerca de…" (útil sólo para extensiones)

Icono URL (Icon URL)

URL de icono 32x32 para mostrar en la lista de complementos. Es opcional en el sentido en que si no se especifica, se asigna un icono por defecto.

Escondido / Desusado (Hidden)

Valor booleano que cuando es true (verdadero) provoca que el complemento no se muestre en la lista de complementos.

Plataforma destino (targetPlatform)

Especifica la plataforma que el complemento soporta. Se pueden especificar múltiples plataformas por manifiesto.

Para extensiones que incluyan componentes binarios (compilados), no es suficiente indicar sólo el SO, sino que también es necesario el ABI* con el que se compiló.

Existe la posibilidad de usar distintas versiones del componente, una para cada plataforma. Ubicándose en "subdirectorios de plataforma-específica".

*Application binary interface: describe a bajo nivel las interfaces entre una aplicación y el sistema operativo u otra aplicación.

Requisitos (requires)

Requisitos para el funcionamiento del complemento. *Requisitos* software que identifican el id de otros complementos y la *compatibilidad* con su versión.

No es posible actualmente agregar dependencias que sean especificas para un <em:targetApplication>

Archivos por defecto

Los archivos por defecto son utilizados para crear un perfil de usuario predeterminado. Su extensión es .js

Ubicación Indica la ubicación física del complemento.
Para crear valores de atributos ubicables en XUL, se escriben los valores en un archivo .ent (o un .dtd). Donde *window* es el localName del elemento raíz del documento XUL, y el valor de la propiedad SYSTEM es la URI de chrome al archivo entity.

Ejemplos

Descripción
<em:description>Herramienta Avanzada foo.</em:description>

Creador
<em:creator>John Doe</em:creator>
<em:creator>El Equipo Extensión </em:creator>

Desarrollador
<em:developer>Jane Doe</em:developer>
<em:developer>Koos van der Merwe</em:developer>

Traductor
<em:translator>Janez Novak</em:translator>
<em:translator>Kari Nordmann</em:translator>

Contribuidor
<em:contributor>John Doe</em:contributor>

<em:contributor>John Doe</em:contributor>
<em:contributor>Jane Doe</em:contributor>
<em:contributor>Elvis Presley</em:contributor>

homepageURL
<em:homepageURL>http://www.foo.com/</em:homepageURL>

updateURL
<em:updateURL>http://www.foo.com/update.cgi?id=%ITEM_ID%&version=%ITEM_VERSION%</em:updateURL>
<em:updateURL>http://www.foo.com/extension/windows.rdf</em:updateURL>

optionsURL
<em:optionsURL>chrome://miext/content/options.xul</em:optionsURL>

aboutURL

```
<em:aboutURL>chrome://myext/content/about.xul</em:aboutURL>
```

iconURL

```
<em:iconURL>chrome://miext/skin/icon.png</em:iconURL>
```

Hidden

```
<em:hidden>true</em:hidden>
```

targetPlatform

```
<em:targetPlatform>WINNT_x86-msvc</em:targetPlatform>
<em:targetPlatform>Linux</em:targetPlatform>
<em:targetPlatform>Darwin_ppc-gcc3</em:targetPlatform>
<em:targetPlatform>SunOS_sparc-sunc</em:targetPlatform>
```

Requisitos

```
<em:requires>
  <Description>
    <!-- Lightning -->
    <em:id>{e2fda1a4-762b-4020-b5ad-a41df1933103}</em:id>
    <em:minVersion>0.5pre</em:minVersion>
    <em:maxVersion>0.5pre</em:maxVersion>
  </Description>
 </em:requires>
```

***Ejemplo* de manifiesto de instalación**

```
<?xml version="1.0"?>

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">

  <Description about="urn:mozilla:install-manifest">
    <em:id>sample@foo.net</em:id>
    <em:version>1.0</em:version>
    <em:type>2</em:type>

    <!-- Target Application this extension can install into,
         with minimum and maximum supported versions. -->
    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>1.0+</em:minVersion>
        <em:maxVersion>1.5.0.*</em:maxVersion>
      </Description>
```

```
    </em:targetApplication>

    <!-- Front End MetaData -->
    <em:name>Sample!</em:name>
    <em:description>A test extension</em:description>
    <em:creator>Your Name Here</em:creator>
    <em:homepageURL>http://www.foo.com/</em:homepageURL>
  </Description>
</RDF>
```

Ubicación

chrome.manifest

```
locale sample sampleLocale chrome/locale/
```

DTD

```
<!ENTITY  button.label     "Clickeame!">
<!ENTITY  button.accesskey "C">
```

XUL

```
<!DOCTYPE window SYSTEM "chrome://packagename/locale/filename.ent">
```

Validación de extensiones y manifiestos

Mozilla usa un servicio Web para validar los complementos hechos por terceros.
https://addons.mozilla.org/es-ES/developers/addon/validate

Antes de esto se puede probar nuestra propia extensión almacenándola en la carpeta local dedicada a ello http://kb.mozillazine.org/Profile_folder y nombrándolo con el id del manifiesto asociado.

Para empaquetar el complemento y subirlo al servidor oficial para su distribución online, se comprime a .zip y se renombra como .xpi, teniendo el archivo el esqueleto propio de este tipo de archivos, ya expuesto.
Mozilla da a disposición de los desarrolladores una dirección Web donde pueden identificarse y subir sus complementos con el formato y condiciones establecidas. La Web es http://addons.mozilla.org/developers/

3.3.3. Correspondencia entre complementos y COTS para TD

Correspondencia con la especificación de nuestro COTS. Las carencias y particularidades de la especificación del complemento según Mozilla con nuestro COTS (dirigido a componentes para plataforma TDT) son comentadas.

Complemento MOZILLA	COTS (TDT)	Particularidad / Comentario
Id	id URI [19]	
Versión	versión Propiedad no funcional	Recogido como propiedad no funcional del componente o aplicación.
Tipo	Categoría	Taxonomía de los componentes por función. Usado por Google, Apple y Microsoft entre otras.
Aplicación destino	▪ Compatibilidad ▪ WSFL (coreografía)	Los plugin: ▪ Se destinan principalmente a un sistema base. En el caso de componentes, su coreografía debe ser meticulosamente establecida ya que la cooperación entre componentes es primordial. ▪ No contemplan compatibilidad hardware, puesto que no lo necesitan. En TDT sí; Ej. todos los PC tienen un teclado, pero no todas las TV lo tienen.
Nombre	Nombre	Nombre de la app.
Descripción	Múltiples descripciones de cada aspecto del COTS.	La mayoría en metalenguajes recomendados W3C o lenguaje natural, según proceda. Cada descripción viene acompañada de un campo “notation” en el cual se indica la notación usada en la descripción.
Creador	Author	El elemento llamado “Packaging”, empaquetado en español, se usa para la información relativa al paquete. Se indicará aquí este dato entre otros.
Pagina inicio URL	Propiedades no funcionales	Son ilimitadas las propiedades que se pueden especificar. Además existe un elemento complejo de Marketing [20].
Desarrollador	- No es útil -	Esta información no es de importancia para la funcionalidad de nuestro sistema. Si algún usuario desea conocer datos del componente a este nivel, tiene a su disposición
Traductor	- No es útil -	
Contribuidor	- No es útil -	

		información del vendedor y del autor, con datos de contacto y enlaces Web que le proporcionarán cuanta información necesite.
Actualización URL	Propiedades no funcionales	Un servicio Web no se descargará de forma indefinida para ser usado, sino que se usará desde la nube. Su actualización está en un plano no visible al usuario. Si es útil para las aplicaciones o para los componentes de uso reiterado, que se descarguen para una optimización de su uso.
Opciones URL	Preferencias por defecto	Un componente puede ser susceptible de configurarse para enfocar su función. Estos parámetros iniciales pueden estar determinados por defecto y cambiarse adaptándose a las necesidades del servicio concreto. Estos parámetros se recogen en el propio sistema de información de una aplicación MHP.
Acerca de URL	Marketing	Toda esta información y demás relevante de este aspecto se encuentra en el elemento complejo de Marketing.
Icono URL	Icono ▪ Icono Web ▪ Icono TDT ▪ Imagen	Se incluye el elemento de imagen corporativa del componente o entidad que lo crea o suministra. Son necesarios al menos 2 tipos. Se incluyen 3 para abarcar todo el espectro de posibilidades futuras.
Escondido (Hidden)	- No es útil -	Un servicio Web no se instala, es usado cuando es necesario.
Plataforma destino	- No es útil -	Existe una gran inversión por la estandarización al más alto nivel de los mecanismos interactivos de la TDT, con el objetivo de ser una plataforma mundialmente accesible.
Requisitos	Compatibilidad *Software* *+* *Hardware* *(Necesario en el nuevo sector)*	En el sector de los complementos en navegadores, la cuestión de compatibilidad hardware se dejó ya atrás hace años. Pero en el ámbito de la TDT, existen nuevas tecnologías que no todos los televisores o Set top box tienen que tener. Como por ejemplo: visión tridimensional, lector de tarjetas de

		dnie, etc. Por tanto, en este nuevo mercado sí que existe una complejidad en hardware que ha de ser resuelto, teniendo un lugar en la especificación COTS para TDT. Según el tipo de la aplicación, usará unas APIs u otras, esta información si es útil para determinar una compatibilidad software (STB - aplicación)
Archivos por defecto	Perfil de usuario	El perfil de usuario contiene las preferencias por defecto del usuario. Un ejemplo puede ser el idioma o un rango de edades permitidas para la ejecución de componentes. Esta información se encuentra en cada STB y será consultada o recogida por el sistema, no en la especificación COTS que llevamos a cabo en este capítulo.
Ubicación	Localización	Referencia URI a la localización física del componente. Puede estar en un repositorio interno o desde terceros. En cualquier caso on-line.
XPCOM **Múltiples:** **- Idiomas** **- IDL**	 - Idiomas - No es útil -	El trabajo de los distintos consorcios en este sector hace viable el uso de un único estándar que haga innecesario más de una definición de las interfaces. El estándar usado será WSDL (véase su apartado) La forma de contener esta información será mediante un enlace, con lo que, si fuera necesario podría almacenarse otro tipo de documento. El mercado del sector es mundial y es necesario un sistema de múltiples idiomas. Al menos de cara al usuario final, consumidor del servicio.

Tabla 1: Correspondencia entre plugins y componentes para TD

3.3.4. Correspondencia entre las características de componente y nuestro COTS

En un capítulo anterior se definen estas características que hacen a un componente reutilizable. Aquí se encuentran recogidas desde el punto de vista del nuevo entorno en el que se encuentra nuestro COTS.

- **Identificable:** Los componentes son identificados inequívocamente mediante una referencia URI. Además la plantilla asociada a cada uno de ellos registra información útil para su catalogación como son las descripciones de su funcionalidad y categorización de la misma. Esta última es una característica añadida a la especificación previa.

- **Accesible sólo a través de su interfaz:** Dicha interfaz está expuesta en la plantilla de todo componente, expresada en un lenguaje recomendación W3C dedicado a tal objeto, WSDL. En cualquier repositorio al que el sistema de enlace se encuentran únicamente códigos pos-compilados, es decir, componentes preparados para su uso, no depuración o manipulación. De hecho, cada componente tendrá asociado, en su correspondiente plantilla, un certificado.

- **Sus servicios son invariantes:** Un componente se caracteriza de cara al público debido a su interfaz. El descriptor de ésta, el documento WSDL no cambia. De producirse, se trata de otro componente, dándose de baja el anterior.

- **Documentado:** Cada componente tiene asociada una plantilla que contiene la especificación de COTS para plataforma TDT que se define a tal efecto y cuya sólida construcción se persigue en los primeros capítulos de este proyecto.

- **Genérico:** No existe restricción alguna por parte del framework que recorte el abanico de posibilidades de cualquier componente en este sistema.

- **Auto contenido:** Depende directamente del código del componente.

- **Mantenido:** El sistema provee de una Web dirigida a los profesionales proveedores de componentes donde principalmente un profesional, podrá: darse de alta, publicar componentes debidamente documentados, actualizarlos o eliminarlos. Así como publicitarlos y vender su servicio.

- **Independiente de la plataforma (hardware y sistema operativo), del lenguaje de programación y de las herramientas de desarrollo:** En el caso que nos ocupa, la plataforma TDT está siendo objeto de un fuerte esfuerzo por parte de consorcios que persiguen la mayor estandarización de todos los aspectos susceptibles de

presentar un problema futuro, debido al libre albedrio. El lenguaje elegido para la programación en esta plataforma es el MHP y su desarrollo está sustentado en tecnologías abiertas como JAVA y entornos de programación *free* y *open source* como es eclipse.

- **Puede ser reutilizado dinámicamente:** Esta característica es básica en todo CSBD.

- **Certificado** Actualmente las empresas dedicadas a este sector con grandes volúmenes de negocio como son Google y Apple, usan los certificados de forma distinta. Apple exige en sus dispositivos que toda aplicación esté certificada por ellos. Google certifica toda aplicación disponible desde su tienda Web, pero si permite la instalación de contenido desde otros lugares, donde las aplicaciones no está certificadas, basta con indicarlo así en las opciones del sistema.

- **Accedido uniformemente sin importar su localidad**: En trabajos previos se definió el COTS dando lugar a las especificaciones de comunicación de las interfaces bajo CORBA. Sin embargo dado que se usan estándares y se da lugar a especificar un enlace a documento y la notación en la que está escrito, cualquier tipo de descripción puede alojarse aquí.

3.3.5. MHP – Sistem Information

Service Information
Es necesario estudiar el servicio de información de MHP [7] para garantizar la compatibilidad con nuestro modelo. La figura muestra el contenido de un TS*, que son un conjunto de ES*[2].

* Transport Stream es una señal multiplexada en frecuencias.
*[2] Elementary Stream es un flujo continuo de información de un mismo tipo.

Se emiten varios servicios o canales de TV (4 aproximadamente) como una serie de ES multiplexado en un TS.

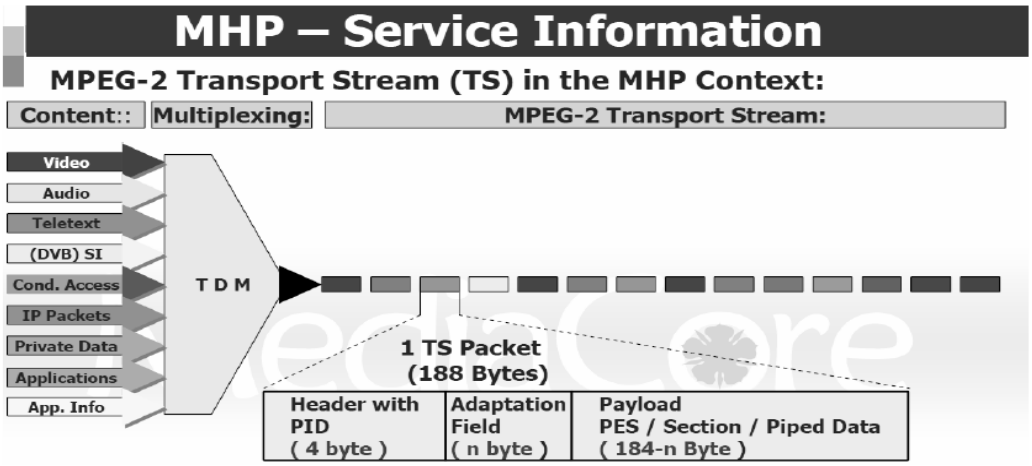


Figura 5: Service Information

El Service Information consiste en información recogida en algunos ES específicos de un TS para indicar a qué canales pertenecen cada ES dentro de un TS. De esta manera **un servicio emitido se identifica por su *Original Network id + TS id + Service id***

Aunque nuestros servicios no se emitan y no se identifiquen por estos datos. Si deben poder identificar a estos servicios por lo que el modelo COTS incluirá esta información para poder dar esta posibilidad.
Esta información es muy útil y de fácil aplicación. Por ejemplo, un componente podría ejecutar alguna funcionalidad automáticamente o sólo cuando un determinado canal o servicio se esté emitiendo.

Sistem Information
Compuesto por tablas de datos.

Estándar	Tabla		Descripción
Estándar MPEG	**PAT**	Program Association Table	Asocia cada ES con su PMT y NIT
	PMT	Program Map Table	Describe para un servicio/canal todos los ES que lo componen indicando el tipo de stream de cada uno
	NIT	Network Information Table	Red física por la que se transportan los TS (satélite, emisor terrestre, cable)
	CAT	Conditional Access Table	Información sobre los sistemas de cifrado que se usan dentro de un TS y cómo decodificarlos
Específica DVB	**SDT**	Service Description Table	Información orientada al usuario: nombre de cada servicio, PIDs, estado, si está codificado (ppv)
	EIT	Event Information Table	Horarios de los eventos (programas/shows) en un Servicio/Canal
	BAT	Bouquet Associantion Table	Agrupación lógica de Servicios. Se usa para poder identificar un grupo de Servicios propiedad de un mismo emisor.

	TDT	Time and Date Table	Tiempos
	TOT	Time Offset Table	

Tabla 2: Tablas del Sistema de Información MHP

Agrupaciones semánticas

- Las 4 tablas del estándar MPEG juntas forman: **PSI**
  Program Specification Information
  La información de PSI permite la configuración automática del receptor para demultiplexar y decodificar los ES de cada TS.

- **BAT**: es útil identificar los componentes / aplicaciones destinados a un determinado emisor, canal o programa.
  De esta manera se hacen reconocibles desde el repositorio las aplicaciones destinadas a un determinado canal o programa. Internamente el componente tiene la información necesaria para actuar sólo en ese determinado programa o canal.

AIT - Aplication Information Table

Contiene toda la información necesaria respecto a las aplicaciones ofertadas en un servicio inlcuyendo todo lo que hace falta para ejecutarlas. Los datos más interesantes son: tipo de aplicación, parámetros de ejecución, nombres de clases y localización de ficheros.

- Tipo de aplicación

application_type	description
0x0000	reserved_future_use
0x0001	DVB-J application
0x0002	DVB-HTML application
0x0003 to 0x7FFF	subject to registration with DVB

(plugins)

Figura 6: Tipos de aplicaciones MHP

DVB-MHP utiliza el lenguaje de programación Java para sus aplicaciones y define la plataforma conocida como **DVB-J**, basada en la Máquina Virtual de Java especificada por Sun Microsystems. DVB-J define un conjunto de APIs genéricas, situadas entre las aplicaciones y el sistema de software, para proporcionar a las distintas aplicaciones acceso a los recursos disponibles en el receptor. El receptor debe contener estas APIs para ser compatible con la aplicación, por ello es de interés incluir esta información en el modelo. Así se evitará comprar o descargar una aplicación no compatible con el STB del cliente.
DVB-HTML provee de acceso a internet a través del televisor.

- Parámetros de ejecución, nombres de clases y localización de ficheros.

  No serán incluidos en la ejecución ya que su información solo es útil a nivel de ejecución, no de búsqueda y clasificación.

- Profiles y versiones. (Compatibilidad)

  MHP, como todo software, sigue un esquema de versionado, y en nuestro caso además se añade un esquema de Perfiles o profiles. Esto significa que una determinada versión de MHP siempre estará enmarcada dentro de un determinado profile, el cual nos indicará los mínimos exigibles de funcionalidades y los opcionales.
  Hoy en día existen 3 profiles y las versiones indicadas en el esquema. Es previsible que se definan nuevas. En nuestro modelo es útil que cada componente indique estos 2 datos a fin de no intentar usarse en un STB que no lo cumpla.
  La forma de mostrarse consistirá en una lista indicando los Profiles y para cada uno la versión de MHP que soporta la App (aplicación o componente)

  En la siguiente figura se muestra un esquema de cruce de Profiles y Versiones:

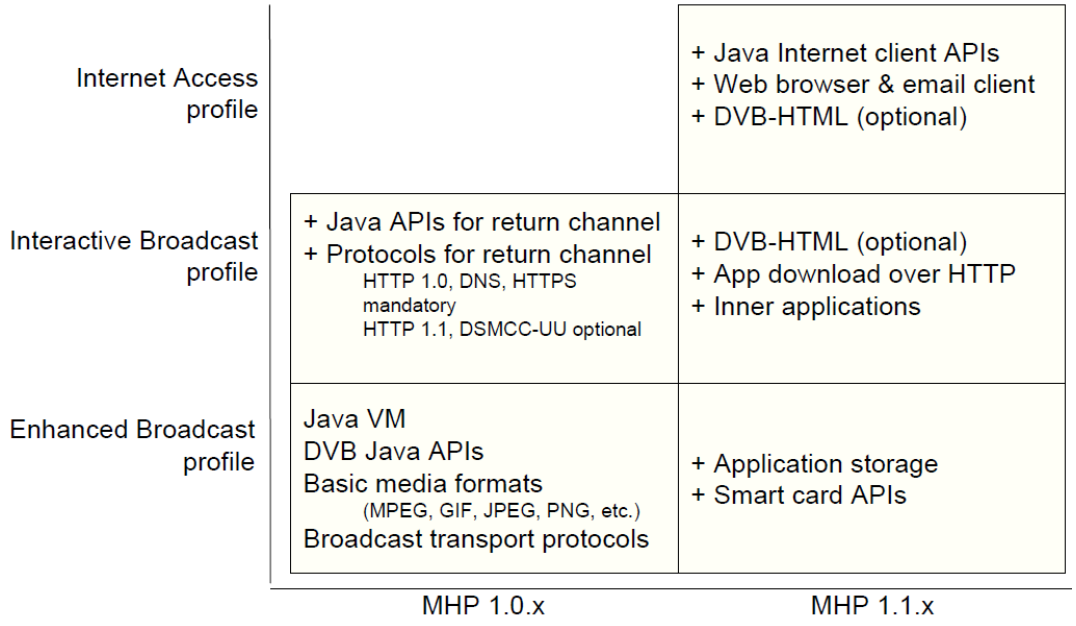


Figura 7: Esquema Profiles y versiones.

Para el tema de las librerías necesarias se usa el **Transport Protocol Descriptor**, que indica de donde descargar las clases necesarias. Esta información está en la aplicación y no tiene interés duplicarla para el modelo COTS. Como nota interesante, se usan direcciones IP ya que el Profile no soporta servicio DNS. Dichas URLs definen directorios o ficheros ZIP.

Además en el descriptor Application Location se usan los términos Length para el tamaño en bytes. Se usará la misma notación para simplificar la comprensión del modelo.

Application icons descriptor

Internamente en la App se encuentra información acerca del icono de la misma. Se indica el formato relevante a la televisión. En el modelo COTS que se presenta aquí no se incluye por tanto éste sino la posibilidad de un icono sin estas restricciones para uso en Web. Así como una imagen sobre el funcionamiento del componente.

Plugins

Al igual que en otros contextos más conocidos, MHP ofrece la posibilidad de incluir dentro de una máquina MHP genérica, bloques de software que saben manejar formatos de aplicaciones que no son cubiertos por la especificación MHP.
Aquellos que deseen implementar este tipo de plugins han de especificar muy claramente dichos formatos, para garantizar la interoperabilidad, de hecho han de ser registrados en DVB para garantizar su consistencia.
Existen 2 tipos de plugin dependiendo del entorno en el que se ejecuten:

- Interoperables: como una aplicación MHP.
- No interoperables: se encuentra en la capa del middleware. Usa código específico de la implementación MHP.

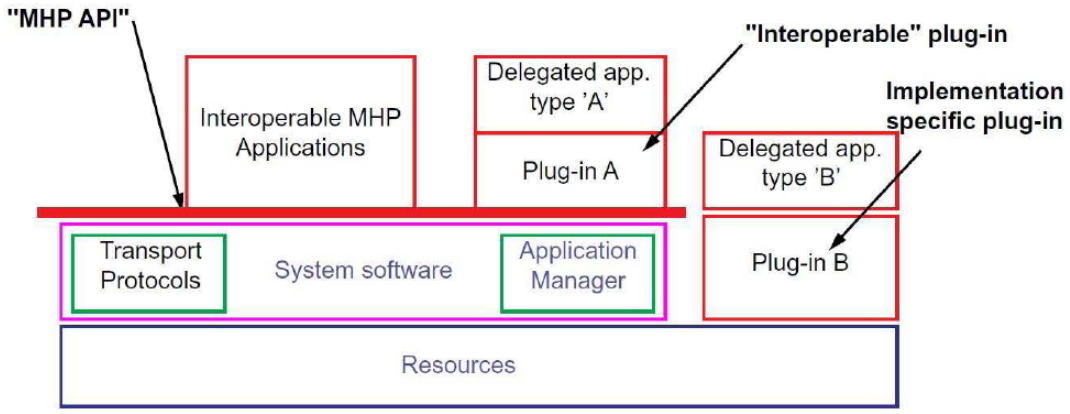


Figura 8: Tipos plugin MHP

En el modelo se da la posibilidad de indicar si se trata de un plugin y su tipo según este baremo.
El mecanismo de perfiles/versiones es el mismo que en apps normales MHP, con sus propios datos, claro está.

3.3.6. Correspondencia entre entornos

a. Perfil de usuario en STB

El perfil de usuario del que se habla anteriormente para los complementos de Mozilla está establecido en la plataforma TDT por el API que se encuentra en org.dvd.user. Son exactamente éstas y sus valores son los de la tabla:

User Language	Código de idioma de 3 letras ISO 639
Parental Rating	Mismo String que el devuelto por javax.tv.service.guide.Content-RatingAdvisory.getDisplayText
User Name	Nombre del usuario
User Address	Dirección del Usuario
User @	Email del Usuario
Country Code	Código de país de 2 letras ISO 3166-1
Default Font Size	Tamaño en puntos para el texto del Body. 26 por defecto
Post Code	Código postal

Figura 9: Perfil de usuario en MHP

Desde el punto de vista de seguridad, las aplicaciones no firmadas (unsigned apps – tienen menos permisos de acceso a recursos) tendrán acceso a las siguientes propiedades:

- User Language
- Parental Rating
- Default Font Size
- Country Code

De hecho, no se permite la comunicación entre aplicaciones firmadas y no firmadas. Esto se indicará en la plantilla del componente, a fin de facilitar la coreografía entre ellos.

b. Multi-idioma

La DVB adopta para la TDT el estándar de códigos de idiomas ISO 639
Define los códigos que representan los lenguajes. Consiste en las siguientes

- *Part 1: Alpha-2 code*
- *Part 2: Alpha-3 code* ← *Es el usado hoy día*
- *Part 3: Alpha-3 code for comprehensive coverage of languages*
- *Part 4: Implementation guidelines and general principles for language coding*

- *Part 5: Alpha-3 code for language families and groups*
- *Part 6: Alpha-4 code for comprehensive coverage of language variants*

[21]

ISO 639-5 Es la quinta parte del código que constituye las normas internacionales de códigos de lengua. Fue publicada en 2008 e incluye **códigos de tres letras** (*alfa-3*) para 114 familias de lenguas y grupos. La Biblioteca del Congreso de Estados Unidos es el agente registrador de esta parte de las normas.

En cualquier caso se trata de una cadena de texto. Cuya longitud ha ido creciendo de 2 a 4 caracteres y no se descartan más.

Trabajos previos

Mozilla usa XPCOM, que es un modelo sencillo de componente multi-plataforma. Lo importante en nuestro caso es que posee, múltiples vínculos idiomáticos.
En otros lenguajes de programación se viene usando archivos de recursos que recogen las cadenas de texto (en múltiples idiomas) que se muestran en la capa de presentación como variables. Se muestran las cadenas pertenecientes a un idioma u otro según el idioma del sistema donde se ejecute. Así funcionan lenguajes de programación como Delphi o Java.

En el caso que nos ocupa el mercado del sector es mundial y es necesario un sistema de múltiples idiomas. Al menos de cara al usuario final, consumidor del servicio. Es imposible predecir cómo evolucionará la forma de tratar este tema, si DVB establecerá unas bases en una próxima versión o se desarrollará un plugin para gestione idiomas. En todo caso desde este modelo se da la posibilidad de incluir de forma independiente un archivo a modo de recurso que no interfiera con futuras soluciones.

Mi propuesta para componentes y aplicaciones en MHP de multilingual automático está explicada tras la especificación del modelo. (Véase punto – Cambios respecto a trabajos previos)

c. Documentación automática

WSDL – Apache Axis, .NET, Eclipse

Una de las tareas más importantes a la hora de crear una SOA es definir su **modelo de servicios:** qué servicios hay y qué tareas en concreto hace cada uno. Este modelo se plasmará en **documentos WSDL** que definan en detalle las interfaces de cada servicio:

operaciones, datos recibidos, datos devueltos y errores que pueden ocurrir. Estos WSDLs son casi imprescindibles a la hora de crear los clientes de un servicio, pues facilitan enormemente la tarea de invocarlo y gestionarlo. No en vano son recomendación W3C.

[22]

La mayoría de las herramientas de creación de servicios web, como **Apache Axis** o **Visual Studio .Net**, facilitan que primero se implemente el servicio (o un esqueleto del mismo), por ejemplo en Java, y a partir de él se genere **automáticamente** el WSDL.

Aunque lo apropiado según expertos es crear primero el documento para que en todo caso sea el servicio el que se adapte al documento y no al revés, creando posibles dependencias. Ya que el beneficio clave de los servicios web es la **interoperabilidad**, que se fundamenta en la **independencia** de las plataformas. Sea como fuere, es claro que, este documento es necesario, y el único que puede hacerlo es el desarrollador de la app. Si bien es interesante facilitar en cuanto sea posible el trabajo de cumplimentar la plantilla del componente.

Eclipse es uno de los entornos de desarrollo más usado para JAVA, dado que el código ejecutable sobre un Set Top Box es MHP y está basado en JAVA, es muy recurrida esta herramienta para desarrollar aplicaciones. Disfruta de una gran variedad de complementos que lo extienden funcionalmente. Todos ellos se encuentran en http://marketplace.eclipse.org la tienda de plugin de eclipse.

Para este proyecto en el que establecemos un modelo de COTS para TDT es muy interesante facilitar a los desarrolladores la tarea de cumplimentar dichos parámetros. La solución pasa por la generación automática de documentación estándar respecto a los que usamos.

En la tienda de eclipse encontramos complementos muy útiles para este cometido:

- **WSDL Extensions Generator 1.0** (License: Free CPL)
  Este plugin facilita la generación de WSDL con soap 1.1 y 1.2.

- **WSDL 2.0 Converter** (License: Free CPL)
  Plugin para convertir entre versiones WSDL 1.1 a 2.0

d. Leyes en televisión

El **control paterno** en los aparatos electrodomésticos, normalmente en aquellos destinados a la reproducción o recepción de imágenes e información; consiste en impedir, o limitar el acceso al manejo de los mismos, o a su contenido a menores de edad.

Muchos países regulan legalmente los contenidos que se pueden difundir. Lo más habitual es que haya una regulación que interrelacione horarios con contenidos evitando que en horarios donde pueda haber presencia infantil los contenidos sean apropiados. Pero en cuestión de aplicaciones a la carta, es necesario otro tipo de sistemas. Es también normal que se indiquen la introducción de avisos sobre la franja de edad a la que está destinado el programa, tanto en televisión como en otros medios.

Las leyes suelen prevenir la utilización de contenidos que puedan perjudicar seriamente el desarrollo físico, mental o moral de los menores, ni programas que fomenten el odio, el desprecio o la discriminación por motivos de nacimiento, raza, sexo, religión, nacionalidad, opinión o cualquier otra circunstancia personal o social, o el desarrollo físico, mental o moral de los menores.

Las nuevas tecnologías de emisión de televisión, la televisión digital, permite que las emisoras faciliten señales de control dependiendo del tipo de programa que se está emitiendo en ese momento, señales que mediante el ajuste previo del receptor, bloquean el acceso a la información. Una práctica habitual, en muchos sitios regulada por ley, es la introducción de un aviso, en forma de texto o logo (como los rombos que se utilizaban en España en la década de los 70 del siglo XX) indicando la edad para la que está recomendado el contenido del programa correspondiente.

Estas señales en el contexto de la TDT consisten en simples cadenas de texto que indican el tipo de contenido que se está emitiendo. Podemos extrapolar este uso a las aplicaciones usando el mismo método. Algunos ejemplos de cadenas usadas en televisión son:

TV-Y
Contenido dirigido expresamente a una audiencia de entre 2 a 6 años.

TV-Y7
Dirigido a menores de 7 años en adelante.

TV-Y7-FV
Dirigido a menores de 7 años en adelante. Advierte de que contiene violencia no explícita o de fantasía.

TV-G
Audiencia en general, es decir, no está dirigida a ningún sector concreto.

Es apropiado para todas las edades pero no está dirigido a menores exclusivamente.

TV-PG
Audiencia en general, pero en el caso de consumidores menores de edad, sugiere la supervisión de un adulto, ya que el contenido puede ser inapropiado para menores.
Este rating puede también incluir las letras:
V para violencia, **S** sexo, **L** lenguaje, **D** dialogo sugestivo

[9]

3.3.7. Tabla comparativa: Análisis de Web comerciales

Los comercios analizados (**13**) pertenecen al mercado de software de reconocido éxito, son las siguientes:

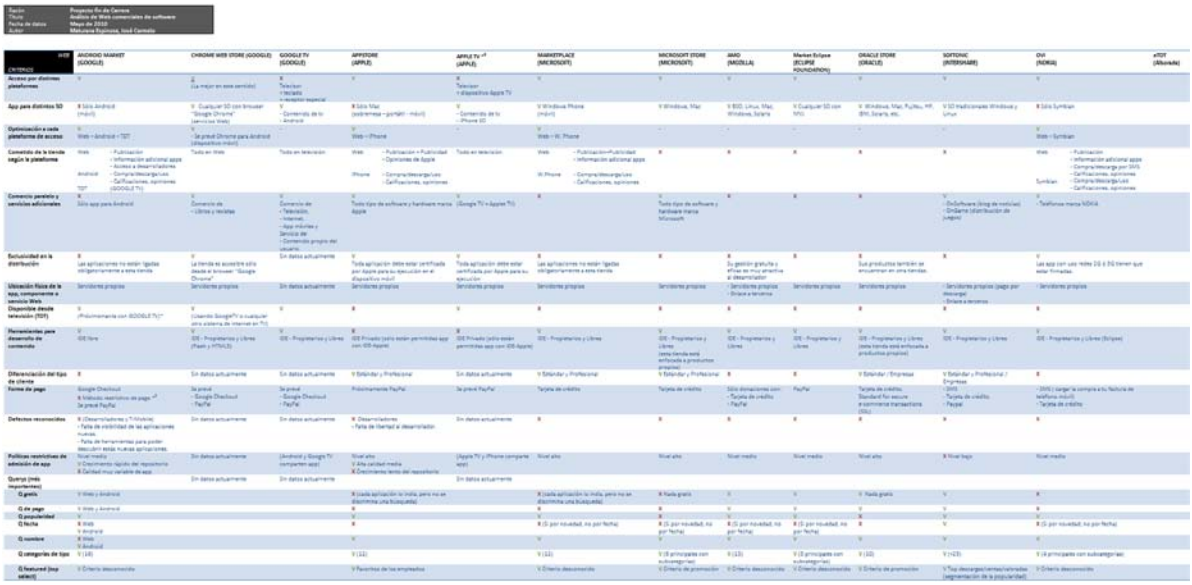
EMPRESA	TIENDA	ENLACE
Google	**Android Market** [23]	http://www.android.com/market/#app=com.epocrates
	Chrome Web Store	https://chrome.google.com/webstore
	Google TV	http://www.google.com/tv/
Apple	**AppStore** [24]	http://www.apple.com/es/iphone/apps-for-iphone/
	Apple TV	http://www.apple.com/es/appletv/
Microsoft	**Market Place**	http://marketplace.windowsphone.com/Default.aspx
	Microsoft Store	http://emea.microsoftstore.com/es/es-ES
Mozilla	**AMO**	https://addons.mozilla.org/es-ES/firefox/
Eclipse Foundation	**Market Eclipse**	http://marketplace.eclipse.org/
ORACLE	**Oracle Store**	https://shop.oracle.com/pls/ostore/f?p=ostore:home:0
INTERSHARE, S.L.	**Softonic**	http://www.softonic.com/
NOKIA	**Store OVI**	http://store.ovi.com/ http://www.ovi.com/services/
SONY	**AppliCast**	http://www.braviawidgets.com/ http://www.braviawidgets.com/descargas/widgets

El análisis consiste en evaluar los criterios (**20**) de:

- Acceso por distintas plataformas
- App para distintos SO
- Optimización a cada plataforma de acceso
- Cometido de la tienda según la plataforma

- Comercio paralelo y servicios adicionales
- Exclusividad en la distribución
- Ubicación física de la app, componente o servicio Web
- Disponible desde televisión (TDT)
- Herramientas para desarrollo de contenido
- Diferenciación del tipo de cliente
- Forma de pago
- Defectos reconocidos
- Políticas restrictivas de admisión de app
- Querys (más importantes)
    - Q gratis
    - Q de pago
    - Q popularidad
    - Q fecha
    - Q nombre
    - Q categorías de tipo
    - Q featured (top select)

Los resultados se encuentran en la siguiente tabla.



En CD-ROM de recursos adjunto

WEB CRITERIOS	ANDROID MARKET (GOOGLE)	CHROME WEB STORE (GOOGLE)	GOOGLE TV (GOOGLE)	APPSTORE (APPLE)	APPLE TV [3] (APPLE)	MARKETPLACE (MICROSOFT)	MICROSOFT STORE (MICROSOFT)	AMO (MOZILLA)
Acceso por distintas plataformas	V	V (La mejor en este sentido)	X Televisor +teclado +receptor especial	V	X Televisor +dispositivo Apple TV	V	V	V
App para distintos SO	X Sólo Android (móvil)	V Cualquier SO con browser "Google Chrome" (servicios Web)	V - Contenido de tv - Android	X Sólo Mac (sobremesa – portátil - móvil)	V - Contenido de tv - iPhone SO	V Windows Phone (móvil)	V Windows, Mac	V BSD, Linux, Mac, Windows, Solaris
Optimización a cada plataforma de acceso	V Web – Android – TDT	V - Se prevé Chrome para Android (dispositivo móvil)	-	V Web – iPhone	-	V Web – W. Phone	-	-
Cometido de la tienda según la plataforma	Web - Publicación - Información adicional apps Android - Acceso a desarrolladores - Compra/descarga/uso TDT - Calificaciones, opiniones (GOOGLE TV)	Todo en Web	Todo en televisión	Web - Publicación + Publicidad - Opiniones de Apple iPhone - Compra/descarga/uso - Calificaciones, opiniones	Todo en televisión	Web - Publicación+Publicidad - Información adicional apps W.Phone - Compra/descarga/uso - Calificaciones, opiniones	X	X
Comercio paralelo y servicios adicionales	X Sólo app para Android	V Comercio de - Libros y revistas	V Comercio de: - Televisión, - Internet, - App móviles y Servicio de: - Contenido propio del usuario.	V Todo tipo de software y hardware marca Apple	V (Google TV = Applet TV)	X	V Todo tipo de software y hardware marca Microsoft	X
Exclusividad en la distribución	X Las aplicaciones no están ligadas obligatoriamente a esta tienda	V La tienda es accesible sólo desde el browser "Google Chrome"	Sin datos actualmente	V Toda aplicación debe estar certificada por Apple para su ejecución en el dispositivo móvil	V Toda aplicación debe estar certificada por Apple para su ejecución	X Las aplicaciones no están ligadas obligatoriamente a esta tienda	X	X Su gestión gratuita y eficaz es muy atractiva al desarrollador
Ubicación física de la app, componente o servicio Web	Servidores propios	Servidores propios	Sin datos actualmente	Servidores propios	Servidores propios	Servidores propios	Servidores propios	- Servidores propios - Enlace a terceros
Disponible desde televisión (TDT)	V *(Próximamente con GOOGLE TV)**	V (Usando GoogleTV o cualquier otro sistema de internet en TV)	V	X	V	X	X	X
Herramientas para desarrollo de contenido	V IDE libre	V IDE - Propietarios y Libres (Flash y HTML5)	V IDE - Propietarios y Libres	X IDE Privado (sólo están permitidas app con IDE-Apple)	X IDE Privado (sólo están permitidas app con IDE-Apple)	V IDE - Propietarios y Libres	V IDE - Propietarios y Libres (esta tienda está enfocada a productos propios)	V IDE - Propietarios y Libres
Diferenciación del tipo de cliente	X	Sin datos actualmente	Sin datos actualmente	V Estándar y Profesional	Sin datos actualmente	V Estándar y Profesional	V Estándar y Profesional	X
Forma de pago	Google Checkout X Método restrictivo de pago *2 Se prevé PayPal	Se prevé - Google Checkout - PayPal	Se prevé - Google Checkout - PayPal	Próximamente PayPal	Se prevé PayPal	Tarjeta de crédito	Tarjeta de crédito	Sólo donaciones con: - Tarjeta de crédito - PayPal
Defectos reconocidos	X (Desarrolladores y T-Mobile) - Falta de visibilidad de las aplicaciones nuevas. - Falta de herramientas para poder descubrir estás nuevas aplicaciones.	Sin datos actualmente	Sin datos actualmente	X Desarrolladores - Falta de libertad al desarrollador.	Sin datos actualmente	X	X	X
Políticas restrictivas de admisión de app	Nivel medio V Crecimiento rápido del repositorio X Calidad muy variable de app	Sin datos actualmente	(Android y Google TV comparten app)	Nivel alto V Alta calidad media X Crecimiento lento del repositorio	(Apple TV y iPhone comparte app)	Nivel alto	Nivel alto	Nivel medio
Querys (más importantes)		Sin datos actualmente	Sin datos actualmente		Sin datos actualmente			
Q gratis	V Web y Android			X (cada aplicación lo indica, pero no se discrimina una búsqueda)		X (cada aplicación lo indica, pero no se discrimina una búsqueda)	X Nada gratis	V
Q de pago	V Web y Android			X		X	X	V
Q popularidad	V			V		V	X	V
Q fecha	X Web V Android			X		X (Si por novedad, no por fecha)	X (Si por novedad, no por fecha)	X (Si por novedad, no por fecha)
Q nombre	X Web V Android			V		V	V	V
Q categorías de tipo	V (16)			V (12)		V (12)	V (6 principales con subcategorías)	V (13)
Q featured (top select)	V Criterio desconocido			V Favoritos de los empleados		V Criterio desconocido	V Criterio de promoción	V Criterio desconocido

Market Eclipse (ECLIPSE FOUNDATION)	ORACLE STORE (ORACLE)	SOFTONIC (INTERSHARE)	OVI (NOKIA)
V	V	V	V
V Cualquier SO con MVJ	V Windows, Mac, Fujitsu, HP, IBM, Solaris, etc.	V SO tradicionales Windows y Linux	X Sólo Symbian
-	-	-	V Web – Symbian
X	X	X	Web - Publicación - Información adicional apps - Compra/descarga por SMS - Calificaciones, opiniones Symbian - Compra/descarga/uso - Calificaciones, opiniones
X	X	V - OnSoftware (blog de noticias) - OnGame (distribución de juegos)	V - Teléfonos marca NOKIA
X	X Sus productos también se encuentran en otra tiendas.	X	V Las app con uso redes 2G ó 3G tienen que estar firmadas.
Servidores propios	Servidores propios	- Servidores propios (pago por descarga) - Enlace a terceros	- Servidores propios
X	X	X	X
V IDE - Propietarios y Libres	V IDE - Propietarios y Libres (esta tienda está enfocada a productos propios)	V IDE - Propietarios y Libres	V IDE - Propietarios y Libres (Eclipse)
X	V Estándar / Empresas	V Estándar y Profesional / Empresas	X
PayPal	Tarjeta de crédito. Standard for secure e-commerce transactions (SSL)	- SMS - Tarjeta de crédito - Paypal	- SMS (cargar la compra a tu factura de teléfono móvil) - Tarjeta de crédito
X	X	X	X
Nivel medio	Nivel alto	X Nivel bajo	Nivel medio
V	V Nada gratis	V	X
V	V	V	X
V	X	V	V
X (Si por novedad, no por fecha)	X	V	X (Si por novedad, no por fecha)
V	V	V	V
V (3 principales con subcategorías)	V (10)	V (+25)	V (4 principales con subcategorías)
V Criterio desconocido	V Criterio de promoción	V Top descargas/ventas/valoradas (segmentación de la popularidad)	V Criterio desconocido

WEB *CRITERIOS*	AppliCast (SONY)
Acceso por distintas plataformas	X Televisor compatible
App para distintos SO	X Sólo AppliCast
Optimización a cada plataforma de acceso	-
Cometido de la tienda según la plataforma	Todo en televisión
Comercio paralelo y servicios adicionales	X Sólo app para AppliCast
Exclusividad en la distribución	X Las aplicaciones no están ligadas obligatoriamente a esta tienda.
Ubicación física de la app, componente o servicio Web	Servidores propios
Disponible desde televisión (TDT)	X Actualmente la forma de obtener los widgets necesita de un PC
Herramientas para desarrollo de contenido	Las aplicaciones son desarrolladas usando el API de AppliCast
Diferenciación del tipo de cliente	X
Forma de pago	Utiliza la pasarela de pagos segura de "La Caixa" para realizar los cobros por tarjeta de crédito
Defectos reconocidos	X Requiere hardware muy específico Escaso repositorio de app
Políticas restrictivas de admisión de app	Nivel medio
Querys (más importantes)	El repositorio es tan pequeño que no necesita búsquedas, sólo se listan todas.
Q gratis	X No hay app gratis
Q de pago	X
Q popularidad	X
Q fecha	X
Q nombre	X
Q categorías de tipo	X
Q featured (top select)	X

Notas

* New York Times: Google, en asociación con Intel y Sony estarían preparando una versión de Android para televisiones.

*2 preparando una versión de Android para televisiones.

*3 Se propone de forma popular un sistema de pago para cargar directamente las apps en la factura del telf. El problema son los acuerdos necesarios entre compañías.

En la nueva versión funcionará con contenidos en "la nube" y correrá con iPhone OS, siendo compatible con las app. (Misma estrategia de funcionamiento que Google TV)

3.3.8. Análisis de Web comerciales

Android Market es un programa informático basado en un sistema abierto de distribución de contenidos desarrollado por Google para dispositivos basados en el sistema operativo Android (también desarrollado por Google), el cual permite a sus usuarios navegar, comprar, instalar y descargar aplicaciones desarrolladas por terceros.

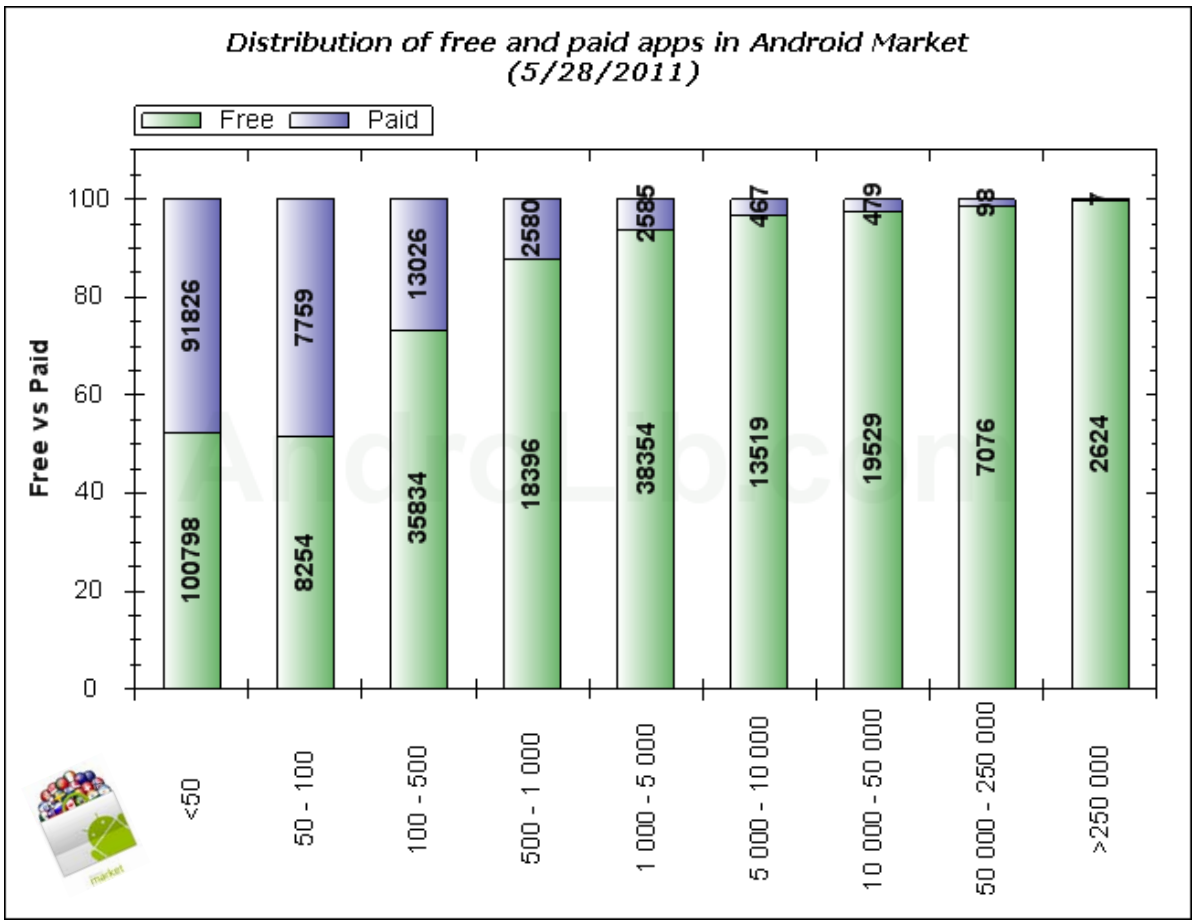


Figura 10: Androlib.com

Las tiendas por Internet están cada vez más en alza y más si se trata de tiendas de aplicaciones. No es de extrañar que las principales plataformas de móviles decidan crear portales donde poder descargar o comprar todas las aplicaciones posibles. Entre estas hablamos de dos de las más importantes, la App Store de Apple y la Android Market de Google pero también hay otras como Symbiam, Blackberry, Windows Mobile o Palm. El desarrollo de aplicaciones para el iPhone no está siendo de fácil acceso por culpa de la política de admisión de aplicaciones de Apple que es muy restrictiva. En cambio Android Market no hace tantas excepciones con las aplicaciones, aceptando todas, propias o de desarrolladores, gracias a su herramienta Android SDK.

Android Market cuenta con el 60% de sus aplicaciones gratuitas. En cambio App Store contiene unas 100.000 aplicaciones de las cuales sólo el 22% son gratuitas. Con lo que se refiere al número de descargas App Store no tiene competencia al menos por ahora, ya que es muy superior a sus competidores. En España la evolución de Android Market parece ser mejor comparado con el nivel mundial.

Para vender aplicaciones debes de crearte una cuenta de comerciante de Google Checkout*, y subir el archivo de la aplicación a los servidores. Para crearte una cuenta de comerciante deberás proporcionar información privada, de contacto y financiera. El precio de la aplicación se puede cambiar en cualquier momento siempre y cuando no la hayas publicado anteriormente como gratuita. Los intervalos de precios permitidos son entre 0,99 y 200 dólares estadounidenses, o entre 0,50 y 100 libras esterlinas.. Los desarrolladores de las aplicaciones de pago reciben un 70% del precio total de la aplicación, mientras que el 30% restante es destinado a las empresas. El beneficio obtenido del 'Android Market' es pagado a los desarrolladores a través sus cuentas en el sistema Google Checkout.

En un primer momento sólo los desarrolladores en Estados Unidos y Reino Unido tenían soporte para publicar aplicaciones de pago. Actualmente Google ha aumentado esa lista con países como Austria, Francia, Alemania, Italia, España y Países Bajos. Por el contrario la lista de países con disponibilidad de desarrolladores que pueden distribuir aplicaciones gratuitas es: Australia, Austria, República Checa, Francia, Alemania, Italia, Países Bajos, Polonia, Singapur, España, Reino Unido y Estados Unidos.

***Google Checkout** es un servicio de pagos en línea seguro que es proporcionado gratuitamente por Google, y que a su vez permite simplificar el proceso de pago por la compras en línea. Los usuarios almacenan su tarjeta de crédito o débito y la información de envío en su cuenta de Google, así que ellos pueden comprar en las tiendas que tienen el servicio clicando un botón. Google Checkout también ofrece protección contra fraudes, así como una página para el seguimiento de las compras y su condición.

Google Checkout fue gratuito para los comerciantes hasta el 1 de febrero de 2008. Desde esa fecha Google carga a los comerciantes un 2.0% más un $0.30 por transacción (1.5% + €0.15 para los comerciantes de UK). También desde esta fecha, los comerciantes que se anuncian con cuentas AdWords* no se les cargara cuotas en transacciones mensuales que sean menores a diez veces su gasto mensual en AdWords.

El alcance de Google Checkout está enfocado en permitir realizar pagos en un solo paso desde un comprador a un mercader. A diferencia de PayPal, Google Checkout no permite el uso de fondos almacenados, ni tampoco de pagos de persona a persona.

***Google AdWords** es el método que utiliza google para hacer publicidad patrocinada, cuenta con enormes cantidades de clientes con webs de todo tipo y de todas partes del mundo. Son anuncios que se muestran de forma relevante en los resultados de la búsqueda del usuario (por ej., si el usuario buscó "coches", a la derecha o arriba de las páginas indexadas por PageRank aparecerán anuncios referentes a "coches"). Google cobra al cliente por cada clic hecho sobre su anuncio. AdWords no solo aparece en el buscador Google, sino también en las patrocinadas por AdSense si el contenido de dichas webs se relacionan con el de la web del cliente y en Gmail.
Podríamos decir que **AdWords** es básicamente el corazón de la facturación de Google, brindando un método de publicidad inteligente para el cliente, puesto a que el costo será "un espejo" del tráfico ganado en la web gracias a Google.

La gran novedad que aporta Android market hace referencia a las desarrolladores, estos serán capaces de hacer su contenido disponible en un servicio abierto el servicio de **Google** que **ofrece una retroalimentación y sistema de calificación similar a YouTube**. Esto se reflejará en la especificación que llevamos a cabo sobre el COTS para componentes especializados en la plataforma TDT, en forma del elemento “QoS” [25] (Quality of Service). Google presenta el siguiente test para evaluar la calidad de los componentes en su tienda Android Market al desinstalar una aplicación.

Indica el motivo por el que has eliminado este elemento:

- No la utilizo o no me interesa.
- Necesito más espacio en el teléfono.
- Es defectuosa.
- Es una aplicación malintencionada.
- Prefiero no indicar el motivo.

Los desarrolladores tienen un entorno abierto y sin obstáculos para hacer su contenido disponible. El contenido puede subirse al mercado en **tres pasos**:

- **registrarse** como comerciante,
- **subir y describir su contenido**,
- **publicarlo** y
- **publicitarlo**.

App Store es un servicio para el iPhone y el iPod Touch, creado por Apple Inc., que permite a los usuarios buscar y descargas aplicaciones informáticas de la iTunes Store, desarrolladas con el iPhone SDK y publicadas por Apple. Estas aplicaciones están disponibles para ser compradas y libres de costo, dependiendo de cada una. Las aplicaciones pueden ser descargadas directamente al iPhone o al iPod Touch por medio de una aplicación del mismo nombre, aunque App Store también está disponible al interior del programa informático iTunes.

App Store: iPhone SDK

El lanzamiento kit de desarrollo de software de iPhone fue anunciado en el evento iPhone Software el 6 de marzo de 2008. El SDK permite a los desarrolladores (que utilicen Mac OS X 10.5.4 o alguna versión superior o una Intel Mac) crear aplicaciones informáticas usando Xcode que correrán nativamente en el iPhone y en el iPod Touch. Una versión beta fue lanzada después del evento y la versión final fue publicada en julio de 2008 junto con el iPhone 3G. Este evento, acompañado con un gran programa de distribución para terceras personas, se convirtió más tarde en el Programa de Desarrolladores de iPhone que ofrece actualmente dos líneas de distribución para los desarrolladores independientes: estándar y empresas.

Las aplicaciones distribuidas por medio del **programa estándar** pueden ser vendidas únicamente a través de la iTunes Store para Mac o para Windows, o bien en la App Store para iPhone o iPod Touch. Los desarrolladores que publiquen sus aplicaciones en la App Store reciben el 70% de los ingresos por ventas y no deben pagar ningún costo de distribución para su aplicación; sin embargo, se requiere pagar una tasa de $99 dólares para utilizar el iPhone SDK y subir aplicaciones a la tienda.

En cambio, las aplicaciones distribuidas por medio del **programa de empresas** son exclusivamente para uso institucional, lo que permite que grandes corporaciones y agencias gubernamentales desarrollen aplicaciones propietarias que no sean para uso público.

Para que una aplicación funcione en el iPhone, este necesita estar registrada, lo que certifica que solo pueda ser concedida por Apple, después que el desarrollado haya creado el software por medio del paquete estándar (99 dólares anuales) o del paquete de empresas (299 dólares por año) y del SDK.

En la actualidad empresas como Adobe se disponen a lanzar una herramienta de desarrollo alternativa al SDK de Apple. Apple ha rechazado esta competencia estableciendo una cláusula por la que hace obligatorio el uso exclusivo de su entorno de desarrollo.

Softonic es una empresa fundada a comienzos del 1997 y perteneciente al Grupo Intercom. Su sede se encuentra en Barcelona, España.
Ofrece más de 100.000 programas, catalogados y explicando en español en qué consiste cada uno de ellos, sirviendo para acercar el software al usuario no especialmente avanzado. Apuntando a este tipo de usuarios, Softonic cobra (mediante el envío de SMS) por permitir bajar programas desde su sitio. Cuando uno busca descargar un juego o programa, aparece la opción de ir a Softonic. Además, también cuentan con una sección orientada a software para profesionales/empresas.

4. Oportunidades de negocio

Dado el carácter comercial que puede desempeñar este sistema, es importante subrayar los elementos que representan una clara oportunidad de negocio. Centrándonos en el objetivo y funcionalidad de cara al usuario final, es la de proveer de forma automática o semiautomática, una lista de componentes efectivos según la necesidad o demanda de servicio del cliente en cada momento.

Es conveniente razonar las necesidades colaterales de los usuarios que lo usen. Y en los beneficios adicionales que podemos ofrecer a los proveedores de componentes. Son las siguientes:

- **Ubicación física del componente**
  Dirección de su ubicación física. De no almacenarse en un repositorio propio, la responsabilidad de su disponibilidad recae en terceros, siendo ésta, una cualidad básica para el conjunto del servicio del sistema trader. Por ello, es conveniente asegurar la accesibilidad a los componentes. Una forma sencilla es usando un repositorio propio, (1ª) oportunidad de negocio, para aquellos proveedores que deseen aportar esta seguridad de accesibilidad a su componente, traducida en un aumento de su fiabilidad y por tanto, de calidad.

- **Publicidad**
  Cada proveedor tiene la oportunidad no sólo de publicar sino también mediante este servicio, publicitar (2ª) oportunidad de negocio su software en televisión, accediendo al gran público. El proveedor podrá usar un sencillo entorno de desarrollo donde crear sus propios anuncios aptos para su emisión en televisión digital terrestre.

- **Calidad** (Referida como QoS en este escrito. *Quality of Service*)
  Datos determinantes para evaluar y conducir la mejora en calidad de servicios, siendo estos proporcionados por componentes y aplicaciones. Estos elementos cambian su valor según los usuarios usen el sistema. Se dividen en dos grupos según el significado de la información que facilitan. Enfocados a la demanda cambiante de los clientes y calidades exigidas.
  En nuestro caso encontramos una nueva (3ªa) oportunidad de negocio en certificar bajo petición del proveedor, los componentes que cumplan una calidad de servicio* determinada.

- **Own Valuation:** Puntuación de la calidad general del componente, por un equipo propio.
  Esta práctica es llevada por numerosos comercios como AppStore o Softonic (Véase – "Tabla comparativa de comercios Web")

***Calidad de Servicio**

Contempla atributos tales como tiempo de respuesta, uso de memoria, precisión, confiabilidad, facilidad de mantenimiento y reutilización, entre otros [25].

Toda esta información se refleja en el uso que la masa de clientes hace de la aplicación. Por ello la valiosa opinión de los clientes acerca de cómo puede mejorar la aplicación también puede ser otra (3ªb) oportunidad de negocio. Si el proveedor lo solicita se le facilitarán estos datos. Consisten en unos porcentajes que indican las deficiencias clave de todo servicio, opinados por los clientes que han dejado de usar un componente o aplicación. *(Véase – "Cambios respecto a trabajos previos")*

- **Idiomas múltiples**

  Es un servicio para el proveedor de componentes que aumenta el valor de los mismos, al poder ofrecer un servicio más personalizado al cliente. Dado el carácter mundial del mercado de software comercial para TDT los clientes potenciales representan todo un abanico multicultural e idiomático. El servicio consiste en la traducción automática (4ª) oportunidad de negocio en el idioma del cliente de la capa de presentación, si la tuviera, del componente. Existen trabajos previos en este tema, véanse los apartados de:

  - "Complementos…" de Eclipse y Mozilla
  - "Correspondencia entre complementos y COTS para TDT"
  - "Cambios respecto a trabajos previos"

5. Modelo de componente en metalenguaje para TD

En nuestro contexto, un tipo de servicio consiste en 4 partes principales [4].

La primera describe los aspectos funcionales, incluyendo información sintáctica (descripción de interfaces, tanto proporcionadas como requeridas; más detallada que en ODP) y semántica. La información semántica puede ser expuesta en dos niveles, dependiendo de si se define el comportamiento de las operaciones (el cual puede ser descrito usando las pre/post condiciones, por operación) o exponiendo el orden relativo en el que el componente espera que sus métodos sean usados (llamados) así como llamadas a otros componentes a través de él. Esto último se encuentra en el elemento Service Access Protocols (también llamado choreography)

La segunda parte describe las propiedades no funcionales por ejemplo QoS, seguridad, etc. Esta parte se corresponde en cierta manera con ODP, esto es, por el significado de las "propiedades" del servicio. Tras la tabla resumen de la especificación se encuentran los cambios acaecidos en este trabajo, entre ellos toma relevancia QoS pasando a ser un nuevo elemento debido al mayor detalle para su uso actual.

La tercera parte contiene información del empaquetado, descarga y despliegue del componente con el servicio demandado. Incluyendo detalles de implementación, contexto y restricciones de arquitectura, etc. Los detalles de implementación se pueden especificar en un documento indicando su notación. Estos datos cumplen el requisito de todo componente al respetar la intimidad del mismo, abstrayendo los detalles que comprometen su portabilidad entre plataformas, tal y como se explicó en el apartado "características de un componente". Para este modelo se extraerán únicamente los datos necesarios, pudiendo ser ampliados, mediante un documento, en la forma dicha.

Y finalmente la cuarta. El marketing, aquí se recoge información no técnica y de negocio. Licencias, precio, vendedor, ofertas especiales, publicidad, etc.

El valor que aporta esta información sobre los componentes del sistema es la de poder hacer búsquedas complejas y eficaces entre ellos por parte de un trader.

La especificación simplificada se compone de los siguientes elementos [3]

COTS Component elements		
Exporting	**Component description (COTS)**	Elementos dedicados a la descripción del COTS.

Importing	**Component query (Querys)**	Elementos dedicados a las búsquedas en el repositorio de COTS.

A continuación se muestra la **tabla detallada del modelo**. Se indican los elementos que la componen, tabulados según su jerarquía. Junto a ellos están las etiquetas que contienen y una breve descripción de su significado. Además las etiquetas destinadas a ser repetibles o unbounded se señalan encuadradas.
Además el modelo permite almacenar la información de las descripciones como cadenas de texto en una etiqueta o como un archivo a parte, referenciado en dicho espacio. La filosofía consiste en proponer una forma de almacenar la información eficaz y fácil, sin ser restrictiva para otras notaciones futuras.

Al dar la posibilidad de usar documentos con una notación internacionalmente reconocida se incentiva la interoperabilidad automática entre componentes o aplicaciones y un entendimiento por parte de desarrolladores. Además al no recoger información sobre ninguna notación, se puede hacer uso de cualquiera, no limitando avances en la materia. Véase – "Documentación automática"

Elemento complejo	Etiqueta	Descripción
Name		Nombre de la app, componente o servicio.
Functional		Listado de funcionalidad/es del componente. *No es posible enumerar todo el abanico funcional.*
Category		Categorización de la funcionalidad del componente. *Propuesta:* *– Compras – Comunicación – Deportes – Sociedad – Salud – Estilo de vida – Finanzas – Herramientas – Multimedia – Noticias y meteorología – Aprendizaje – Productividad – Ocio – Referencia – Viajes – Temas – Variadas.*
Provided Interfaces		*Interfaces que provee (Funcionalidad que aporta)*
	Notation	Lenguaje usado en la definición de las interfaces.
	Description	Descripción de las interfaces. *(Enlace al documento, si procede)* *Por sus características WSDL será el más usado. Pero el modelo permite cualquier otra notación. (Véase punto "Documentación automática")*
Required Interfaces		*Interfaces que necesita (Compatibilidad)*
	Notation	Lenguaje usado en la definición de las interfaces.
	Description	Descripción de las interfaces. *(Enlace al documento, si procede)*
Requirements		*Requisitos software y hardware (versión del set top box – APIs o formatos que soporta-, televisión 3D, smartcard -lector dnie-, teclado, etc.)* *No es posible enumerar todo el abanico de requerimientos futuros. Por ello se usa*

	una lista ilimitada adaptable a notaciones *Notation* — Se indica si se usa una notación determinada. *Description* — Descripción de los requisitos. *(Enlace al documento, si procede)* *Es posible que esta lista se estandarizara en el futuro. Aquí se posibilita usarla.* *SO* — Sistema base donde es ejecutable el componente o aplicación. *Puede ser MHP según el estándar de DVB, Android por Google, etc.* *Requirement name* — Nombre del requisito *Value* — Valor o versión
Service Access Protocol	*Protocolo de uso del componente (Guía de uso - Choreography)* *Describe el orden en que se deben llamar a las interfaces y deben suceder los eventos.* *Notation* — Notación del protocolo. *Description* — Descripción del protocolo. *(Enlace al documento, si procede)* *- Orden relativo de invocación de operaciones del componente, así como llamadas a otros componentes a través del primero.* *Behavior* — Descripción semántica del comportamiento de las operaciones. *(Enlace al documento, si procede)* *- Pre/Post condiciones para cada operación.*
No functional (Properties)	*Propiedades no funcionales del servicio.* *Notation* — Notación de las propiedades. *Description* — Descripción de las propiedades. *(Enlace al documento, si procede)* *Composition* — Conjunto de propiedades. Indica AND/OR, según las propiedades del conjunto se complementen o se excluyan. *Property name* — Nombre de la propiedad. *Value* — Valor. *Property name* — Nombre de la propiedad. *Value* — Valor. *Signed* — Aplicación firmada o no firmada. *Las aplicaciones firmadas y no firmadas no pueden comunicarse.* *Version* — Cadena de texto indicativa de la versión del código *Update URL* — Enlace de actualización de la aplicación. *Sólo si procede.*

	Plugin Indica si el componente es un plugin o no. *(Véase punto MHP – system information)* Type Plugin interoperable o no interoperable. *Profile* Datos sobre el profile *(Véase punto MHP – system information)* *Name* Nombre del profile *Version* Versión del profile *Multi Idiom* Idiomas de la capa de presentación. *Notation* Notación usada *Idioms* Listado de idiomas de la capa de presentación. *Description* Contiene las traducciones de los textos de la capa de presentación del componente. *(Enlace al documento, si procede)* *Parental control* Cadena de texto identificativo del tipo de contenido. *Establecido por DVB.* *(Véase punto – Leyes en televisión)* *BAT* Identifica un determinado emisor, canal o programa, donde está destinada la funcionalidad del componente.
QoS [25] *(Véanse puntos:* *– Análisis Web comerciales* *– Oportunidades de negocio)*	*Información determinante para evaluar y conducir la mejora en calidad de componentes. Estos elementos cambian su valor según los usuarios usen el sistema.* *Enfocados a la demanda cambiante de los clientes y calidades exigidas.* ▪ Use Nº de clientes que usan el componente sin pagar *(Útil para software gratuito o evaluaciones)* ▪ Payment Nº de clientes que usan el componente pagando. *(Útil para software de evaluación)* *Enfocados a la evaluación de calidad y conducción a la mejora de componentes.* ▪ Own Puntuación de la calidad general del componente, por un equipo propio. (unsignedByte) ▪ Coments Enlace a un listado de comentarios sobre el componente o aplicación. ▪ Users Puntuación de la calidad general del componente, por los clientes. (unsignedByte) ▪ Coments Enlace a un listado de comentarios sobre la aplicación. ▪ Disuse Pregunta explícita al cliente de respuesta rápida.

	(Véase punto – Cambios respecto a trabajos previos)
Packaging	*Término típicamente usado para englobar la información respecto al empaquetado de software.* *Notation* — Notación de la descripción del paquete. *Ej. CCM** *Description* — Descripción del paquete. *(Enlace al documento, si procede)* Location — Ubicación física del componente *(Puede existir enlace interno y/o externo)* In Out — Residente en repositorio propio Residente en instalaciones de terceros *(Véase punto – Oportunidades de negocio)* Length — Tamaño del paquete, expresado en bytes. *(Véase punto MHP – system information)*
Marketing	*Datos sobre el marketing del COTS* *License* — Licencia de uso *(Enlace al documento, si procede)* *License type* — Tipo de licencia *(Según los derechos que el autor se reserva sobre su obra - 6 tipos)* *Time slot* — Duración de uso del componente *(Fecha relativa)* *Date* — Fecha de publicación o disponibilidad. *(Fecha absoluta)* *Date of update* — Última fecha de actualización del componente. *(Fecha absoluta)* Icons — Imagen corporativa del componente o entidad que lo crea o suministra. Tipos: ▪ Icon Web: Sus características técnicas de tamaño y resolución no están subyugadas a la especificación application_icons_descriptor, está destinado a la Web. *(Véase punto MHP – system information)* ▪ Image: Imagen del componente en funcionamiento (si procede) ▪ Icon TV: Al ser un modelo no sólo para app MHP, se incluye la posibilidad de insertar un icono destinado a su emisión en televisión. Video — Video que muestra el funcionamiento del componente *Advertising* — Anuncio publicitario apto para su emisión en TDT por terceros, personalizado por el proveedor del componente, haciendo uso de uno de nuestros servicios. *(No se implementará en este proyecto. Enlace al archivo, si procede)*

		(Véase punto – Oportunidades de negocio)
	Price	Precio. *Puede ser cero.*
	Currency	Moneda en la que se tasa la app (Puede ser ficticia)
	Quantity	Importe.
	Certificate	Certificado del componente *(Enlace al documento, si procede)* *(Véase punto – Análisis Web comerciales)*
	Vendor	Información del propietario
	Author	Nombre.
	Company	Compañía, si procede.
	Web	Enlace Web del propietario, si procede.
	Contact	Otros datos de contacto.
	Description	Anotaciones adicionales sobre Marketing.

Tabla 4: Resumen del modelo de componentes

La otra parte de la especificación corresponde a la funcionalidad del sistema en cuanto a consultas para determinar un conjunto de componentes útiles para proporcionar un cierto servicio.

Elemento complejo	Etiqueta	Descripción
Querys (Importing description)	Consultas para encontrar el componente o conjunto de ellos. Según los discriminantes:	
	Description	Descripción
	Functional	Funcionalidad
	Property	Propiedades

Aclaraciones sobre la tabla

* **OPEN CCM. The Software Package Descriptor**. [26]
El archivo suele llamarse “softpkg.csd” o “softpkg.dtd”.
Este descriptor provee de gran cantidad de información sobre el empaquetado de un software.

Parte de ella se refiere a detalles de implementación:

- Referencia IDL (Lenguaje de definición de Interfaces) No es necesario rellenarlo ya que se encuentra en los elementos dedicados a las interfaces, descrito como WSDL.
- Lenguaje de la implementación, que como ya se ha explicado éste será mayoritariamente MHP
- Sistemas operativo en el que se efectuó la compilación y
- El compilador con el que se obtuvo el código máquina. Etc.

Otra parte hace referencia a información sobre marketing, como son el título, licencia, autor, etc.
Estos datos no son usados por la herramienta de despliegue para el paquete, pero son necesarios si se desea compartir el componente. Se han extraído en el modelo los datos estrictamente necesarios para el empaquetado y se ha reservado espacio para especificaciones más detalladas como ésta u cualquier otra.

Exactmatching - Softmatching Coincidencia exacta o aproximada

Las búsquedas muy restrictivas dan un resultado demasiado escueto en comparación al esperado. La holgura en los requisitos de búsqueda permite una ambigüedad necesaria para obtener un abanico suficiente y al menos apto, para solventar la necesidad.

Tipos de coincidencias

Exactmatching	Exacta o nula.
Softmatching	Exacta y parecidas o nula.

Su principal ventaja es la de hacer eficiente las búsquedas de un futuro trader sobre un repositorio de componentes. Por ello se incluirían en el elemento de consultas “Querys”
Estos elementos se encuentran en las etiquetas:

- Interfaces ofrecidas
- Coreografía
- Funcionalidad

5.1 Cambios respecto a trabajos previos

La especificación atrás resumida es resultado de un compendio meditado sobre los trabajos previos en el tema y del intensivo estudio expuesto, basado en los tres pilares sobre los que se sustenta la solución que el modelo quiere aportar; componentes software, comercio de aplicaciones y entorno de televisión digital. Todo ello ha dado lugar a una serie de adaptaciones aquí expuestas. Debido a que son muy numerosas, aquí solo se expondrán las dignas de explicación.

Elementos añadidos

- **Requirements** (Compatibilidad hardware)
  Requisitos hardware (versión del set box, televisión 3D, lector dnie, etc.)
  - Notation Se indica si se usa una notación determinada (enlace opcional)
  - Requirements name Nombre del requisito
  - Value Valor específico o versión

  Además se especifica explícitamente como requisito de carácter software el sistema operativo que ejecuta el componente o aplicación.

- **Propiedades no funcionales**
  En este caso el elemento ha sufrido una profunda modificación. Es evidente que todo software tiene propiedades no funcionales, pero aquí se especifica cuáles son en el entorno de televisión digital principalmente. Tal y como se analiza en el punto – MHP sistema de información, aquí se recoge únicamente la información relevante para su disponibilidad en la plantilla. Los más destacados en este sentido son:

  - **Profile**
    Datos sobre el profile (Nombre y versión)

  - **BAT**
    Identifica un determinado emisor, canal o programa, donde está destinada la funcionalidad del componente.

- **Update URL**
  Enlace a la descarga de la aplicación. Sólo si procede.

- **Signed**
  Indica si el código MHP está firmado o no. Es importante saber esto de cara al choreography ya que no puede existir comunicación entre aplicaciones firmadas y no firmadas.

- **Parental control**

  Muchos países regulan legalmente los contenidos que se pueden difundir. Este campo en el modelo da la oportunidad de que el desarrollador del componente exprese el tipo de contenido de la aplicación.

- **Multi-idiom**

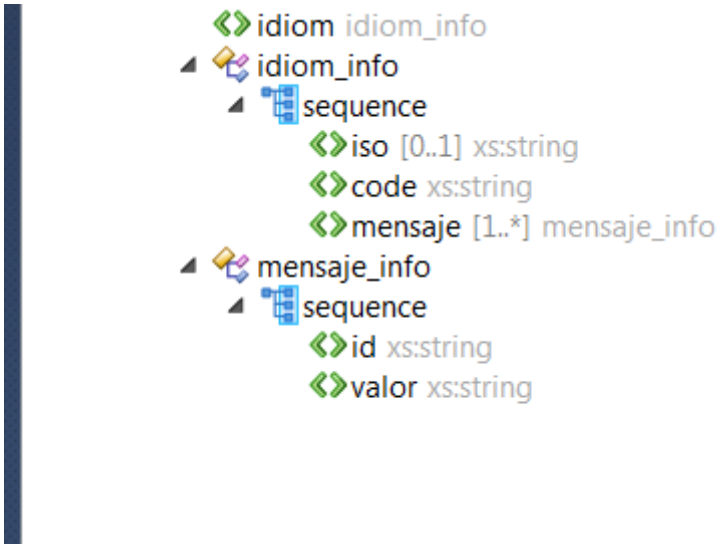
  Enlace al recurso que contiene las traducciones de los textos de la capa de presentación del componente. Se usará uno u otro dependiendo de las preferencias del perfil de usuario que lo esté usando. Véase – "Correspondencia entre entornos – Perfil de usuario en STB"

  Existe gran variedad de entornos de programación que soportan un sistema multilingüe el cual posibilita que el cliente visualice el software en su idioma. Véase – "Complementos…" en Eclipse y Mozilla.

  Planteamiento de la solución

  El recurso consiste en un documento escrito en lenguaje XML cuyo esquema es sencillo. Se compone únicamente de una lista ilimitada de etiqueta de idioma (la cual indica una lengua determinada, según la ISO 639) Y para cada idioma se listan las cadenas de texto referenciadas por un nombre de variable id_cadena.

  La oportunidad de negocio se encuentra en que un desarrollador de aplicaciones no va a traducir su capa de presentación a todos los idiomas donde puede ir a parar su App, gracias a nuestro sistema de distribución, por ello, se propone una traducción automática a partir de un idioma dado en el documento. Y a su vez, incluirlas en él para poder ser mostradas según el perfil de usuario que ejecute la app.
  Es decir, el desarrollador incluye el documento para su idioma y de forma automática, nuestro servicio traduce la capa de presentación al idioma del cliente, incluyendo estas cadenas en dicho documento.



Documento XML de la propuesta como cuarta oportunidad de negocio.
Véanse puntos: "Cambios respecto a trabajos previos" – "Oportunidades de negocio" – "Correspondencia entre complementos y COTS para TDT"

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!-- ********************************************************************
-->
<!--                          XML Schema namespace
-->
<!-- ********************************************************************
-->
<xs:schema  id="multi_idiom"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
            <!--targetNamespace=""-->

    <!--
********************************************************************* -->
    <!--                                 Annotations
-->
    <!--
********************************************************************* -->
    <xs:annotation>
        <xs:documentation>
            XMLSchema para Recurso Multilingüe.
            Grupo de investigación UAL: INFORMÁTICA APLICADA.
            Maturana Espinosa, José Carmelo - Carmelo.Maturana@gmail.com
        </xs:documentation>
    </xs:annotation>

    <!--
********************************************************************* -->
    <!--                           Recurso Multilingüe
-->
    <!--
********************************************************************* -->
```

```
    <xs:element name="idiom" type="idiom_info"/>

    <xs:complexType name="idiom_info">
        <xs:sequence>
            <xs:element name="iso" default="639-5" minOccurs="0"
type="xs:string"/>
            <!-- ISO predeterminada. Se puede indicar otra en un futuro. -
->
            <xs:element name="code" type="xs:string"/>
            <!-- Indica una lengua determinada, según la ISO -->
            <xs:element name="mensaje" type="mensaje_info"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="mensaje_info">
        <!-- Lista de cadenas de la capa de presentación -->
        <xs:sequence>
            <xs:element name="id" type="xs:string"/>
            <xs:element name="valor" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

En CD-ROM de recursos adjunto

Se establece el valor predeterminado para la ISO en la 639-5. La lista de cadenas id – valor corresponde a la lista de mensajes de cara al usuario final que serán traducidas.

Ejemplos de instancias en multi-idioma

```
<idiom>
    <code>en</code>
    <mensaje>
        <id>id_mensaje</id>
        <valor>hola, buenos
días</valor>
    </mensaje>
</idiom>
```

```
<idiom>
    <iso>639-5</iso>
    <code>en</code>
    <mensaje>
        <id>id_mensaje</id>
        <valor>hola, buenos
días</valor>
    </mensaje>
</idiom>
```

En CD-ROM de recursos adjunto

- **License**

  Hace referencia al documento de licencia del software.

- **License type**

  Tipo de licencia (4 tipos principales: shareware, freeware, etc.)

  Discriminante para la búsqueda de componentes según su costo de uso. Empresas como Apple o Google dividen sus aplicaciones entre gratuitas y de pago.

Según los derechos que cada autor se reserva sobre su obra

- **Licencia de software de código abierto permisiva**
  Se puede crear una obra derivada sin que ésta tenga obligación de protección alguna.
- **Licencia de software de código abierto robustas**
  Estas licencias aplican algunas restricciones a las obras derivadas, haciendo que según el grado de aplicación se puedan dividir a su vez en dos sub-categorías:
  - **Licencias de software de código abierto robustas fuertes**
    Las licencias de software de código abierto robustas fuertes o con copyleft fuerte, contienen una cláusula que obliga a que las obras derivadas o modificaciones que se realicen al software original se deban licenciar bajo los mismos términos y condiciones de la licencia original.
  - **Licencias de software de código abierto robustas débiles**
    Las licencias de software de código abierto robustas débiles, con copyleft débil/suave o híbridas, contienen una cláusula que obliga a que las modificaciones que se realicen al software original se deban licenciar bajo los mismos términos y condiciones de la licencia original, pero que las obras derivadas que se puedan realizar de él puedan ser licenciadas bajo otros términos y condiciones distintas.
  - **Licencia de software de código cerrado**
    Estas licencias también se conocen con el nombre de *software propietario* o *privativo*. En ellas los propietarios establecen los derechos de uso, distribución, redistribución, copia, modificación, cesión y en general cualquier otra consideración que se estime necesaria.
  - **Software de dominio público (sin licencia)**
    Se permite uso, copia, modificación o redistribución con o sin fines de lucro.

Según su destinatario

- **Licencia de Usuario Final**
  Es una licencia por la cual el uso de un producto sólo está permitido para un único usuario (el comprador).
- **Licencia de distribuidores**
  En este tipo de contrato, se le asigna derechos restringidos a un comerciante de tipo comisionario para que venda el producto (software) dando una remesa o comisión al fabricante.

  La licencias de uso de software generalmente caen en alguno de estos tipos:
  - **Licencia propietaria**. Uso en una computadora por el pago de un precio.

- **Shareware**. Uso limitado en tiempo o capacidades, después pagar un precio.
- **Freeware**. Usar y copiar ilimitado, precio es cero.
- **Software libre**. Usar, copiar, estudiar, modificar, redistribuir. Código fuente incluído.

[27]

- **Time slot**
  Duración de uso del componente (Fecha relativa)

- **Date**
  Fecha de publicación o disponibilidad

- **Date of update**
  Última fecha de actualización del componente

- **Icon**
  Imagen corporativa del componente o entidad que lo crea o suministra. Tipos:
  Icon Web — Sus características técnicas de tamaño y resolución están destinadas a la Web
  Image — Imagen del componente en funcionamiento (si procede)

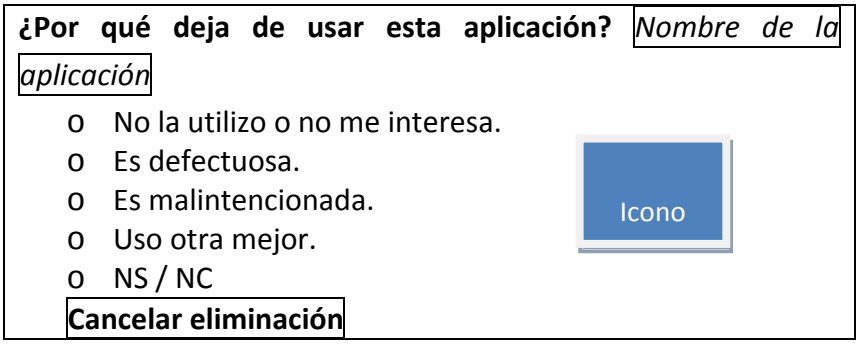
- **Publicidad** - *Advertising* (2ª) oportunidad de negocio
  Cada proveedor tiene ahora la oportunidad de publicitar su software en televisión, accediendo al gran público.

- **QoS**
  Datos determinantes para evaluar y conducir la mejora en calidad de componentes. Estos elementos cambian su valor según los usuarios usen el sistema.

  Se dividen en dos grupos según el significado de la información que facilitan. Enfocados a la demanda cambiante de los clientes y calidades exigidas. Y enfocados a la evaluación de calidad y conducción a la mejora de componentes.

  Se expone su significado en la tabla. Merecen mención los comentarios que pueden hacerse sobre las aplicaciones. Se debe tener en cuenta que estos comentarios no se almacenarán en la plantilla sino sólo estarán referenciados. Es importante considerar el carácter de estos comentarios ya que los clientes opinarán sobre las aplicaciones en su globalidad, ya se compongan de componentes o no, los desarrolladores y un equipo propio si podrían opinar de forma pormenorizada.

  Particularidad del elemento desuso. (3ª b) - oportunidad de negocio
  - **Disuse** Pregunta explícita al cliente de selección de respuesta. Este test de un solo paso sustituye al mensaje típico de confirmación de eliminación o desinstalación.



Este dato será almacenado para cada componente en forma de un número de elecciones de cada opción (Integer), su representación útil será mostrada como un porcentaje de selección para cada respuesta. La información que puede interpretar el proveedor, según los porcentajes de respuesta es:

- No la utilizo o no me interesa: La funcionalidad de la aplicación no es de interés.
- Es defectuosa: La aplicación no realiza correctamente su funcionalidad.
- Es malintencionada: Si la respuesta mayoritaria es ésta, el servicio técnico del sistema debería hacerse cargo de la situación, revisando la aplicación para comprobar si es malintencionada y actuar en consecuencia. Tomando medidas legales incluso, si fueran necesarias.
- Uso otra mejor: La competencia da un mejor servicio.
- NS / NC: El cliente ha preferido no indicar el motivo.

Elementos modificados – ampliados

- **Price**

  Puede ser gratuito. Se indica la moneda y el importe.

- **Location**

  Dirección de la ubicación física del componente (1ª) oportunidad de negocio

  (Puede existir enlace interno y/o externo)

  - Interno Enlace al componente residente en un repositorio propio.
  - Externo Enlace al componente residente en una ubicación externa, accesible online.

  Sea como fuere el componente debe ser accedido uniformemente sin importar su localidad.

  La forma de invocar los servicios ofrecidos por los componentes debiese ser independiente de su ubicación (local o remota).

En caso de los servicios Web: Para ello el modelo de componentes debe estar basado en tecnologías de procesamiento distribuido tales como CORBA, RMI y .NET Remoting. Y en caso de aplicaciones, se deben poder descargar al STB.

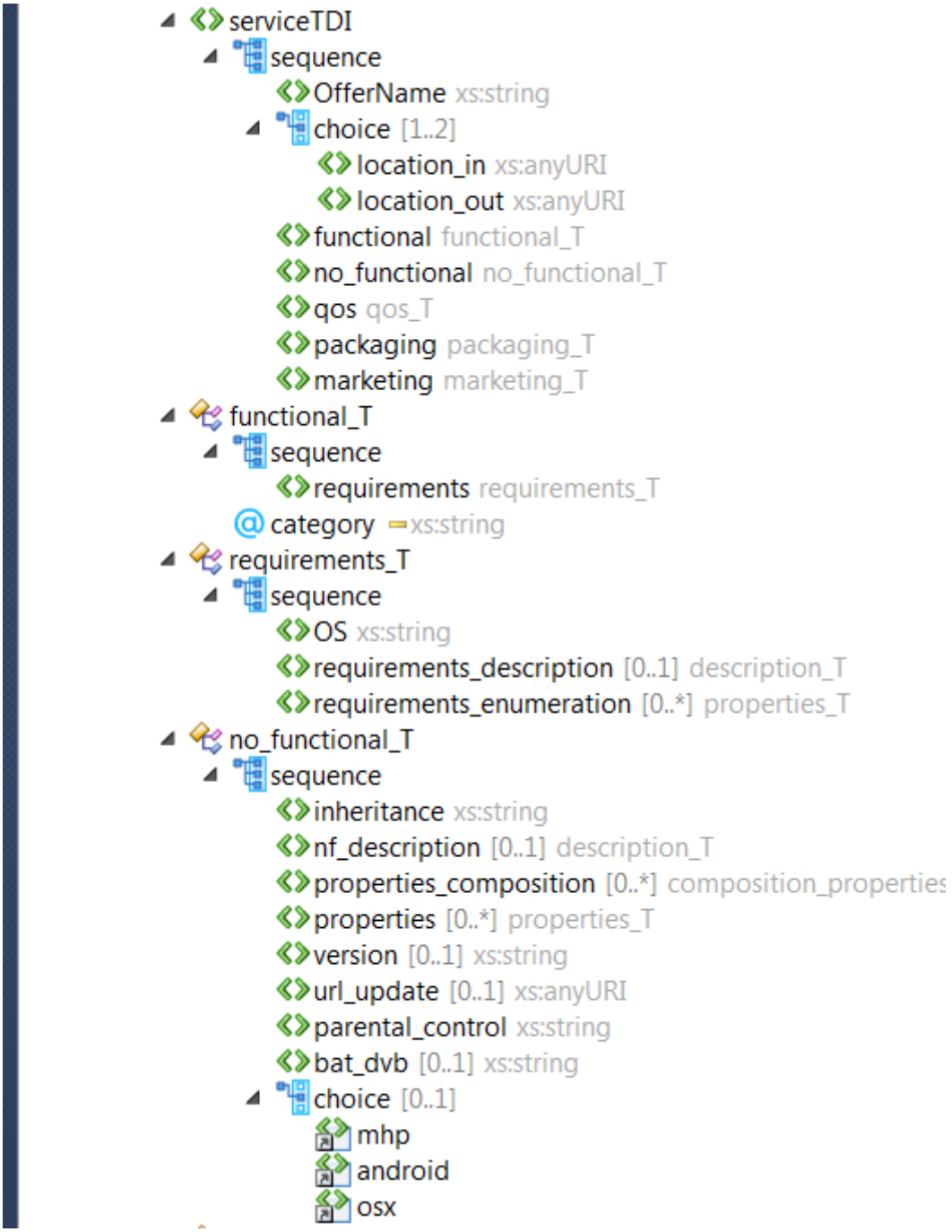
- **Length**
  Tamaño del paquete, expresado en bytes. Contiene el tamaño del paquete completo en su formato de compresión ZIP *(Véase punto MHP – system information)*

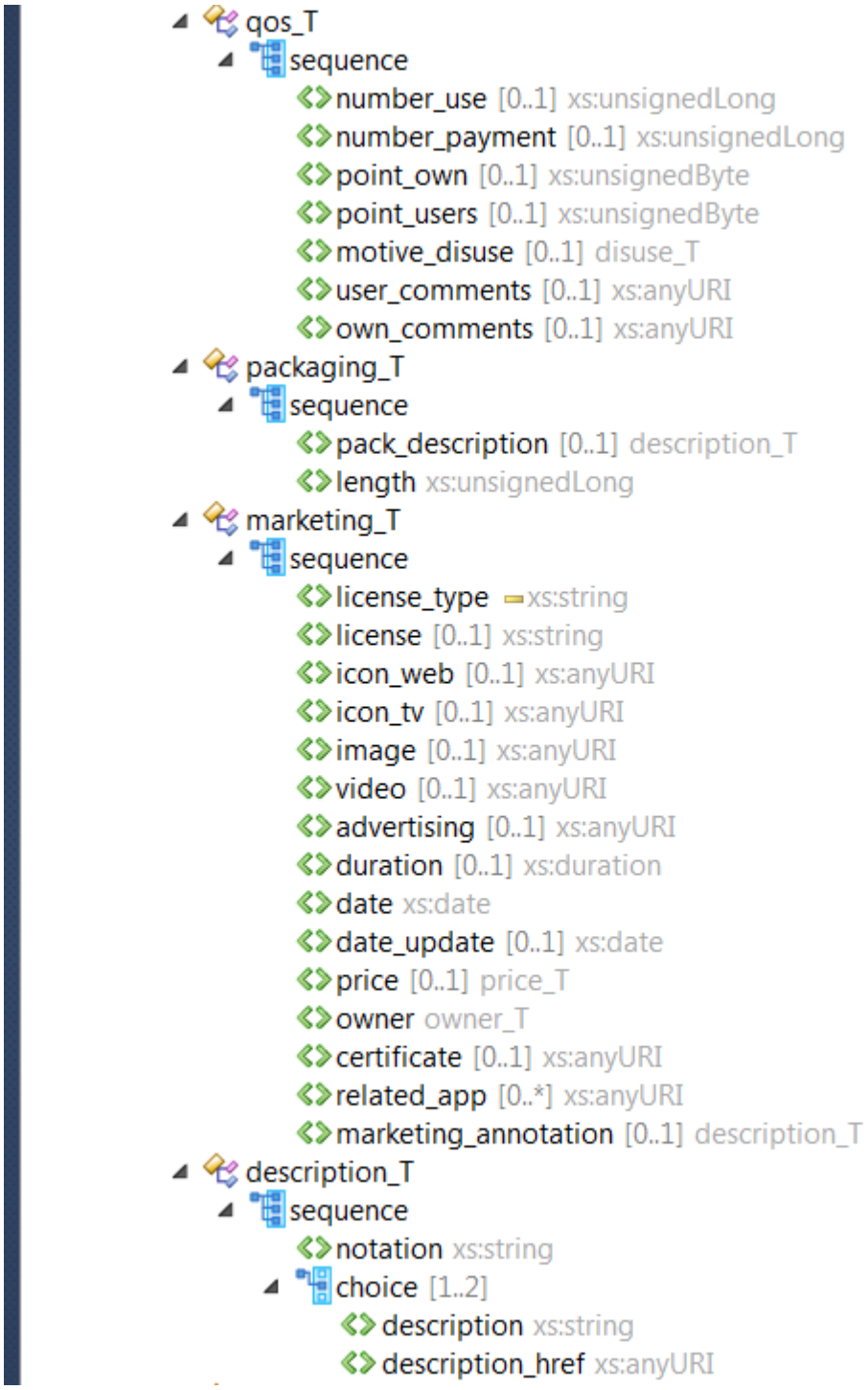
Elementos eliminados

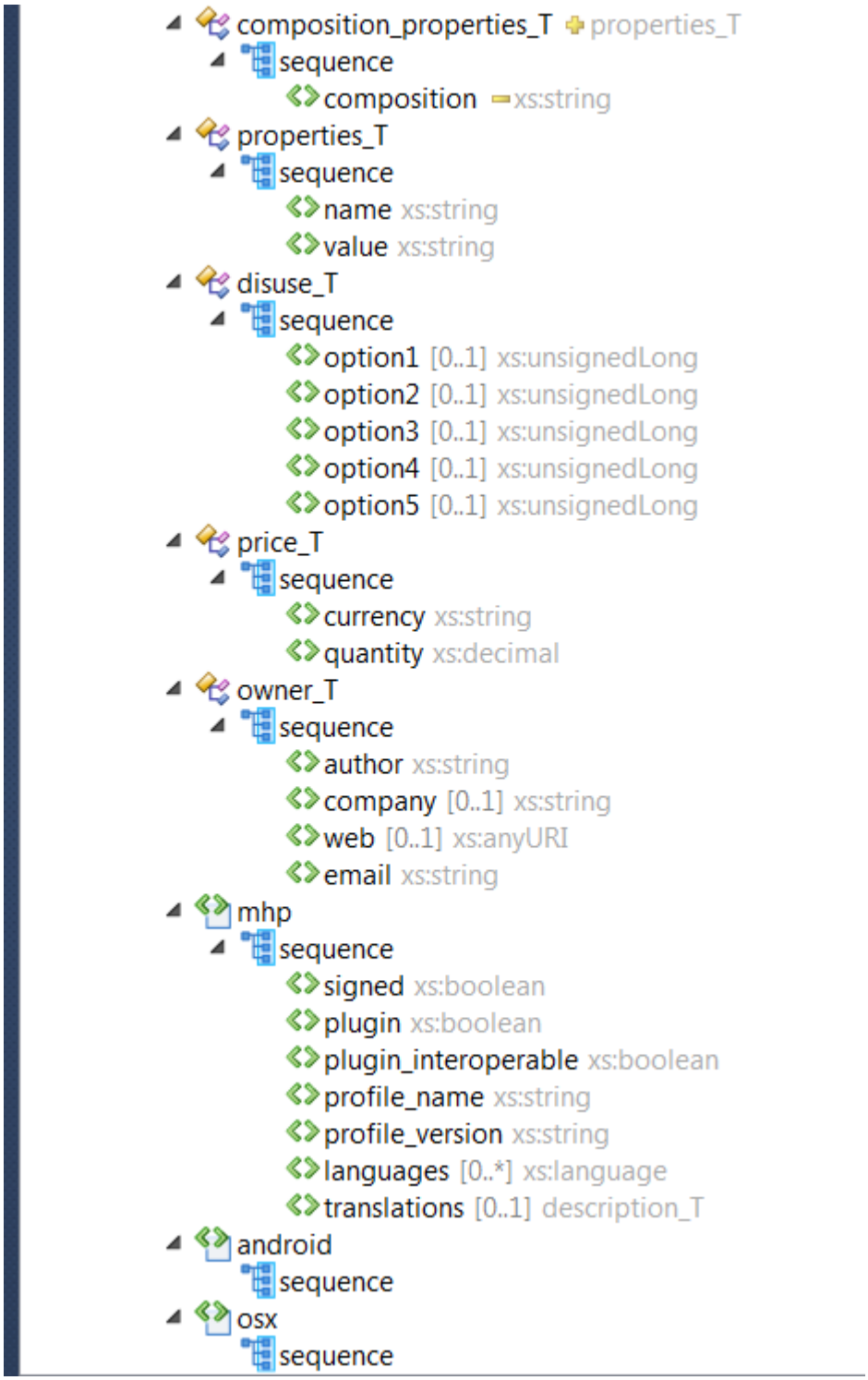
- ***Expiry date*** *(Marketing)*
  Fecha límite de uso del componente (Fecha absoluta)
  Es sustituido por medidas de tiempo relativo de funcionamiento, etiquetado como Time slot.

6. Diseño final de COTS

Este es el diseño del modelo XML para las plantillas de las ofertas, los hasta ahora llamados COTS.




qos_T
sequence
number_use [0..1] xs:unsignedLong
number_payment [0..1] xs:unsignedLong
point_own [0..1] xs:unsignedByte
point_users [0..1] xs:unsignedByte
motive_disuse [0..1] disuse_T
user_comments [0..1] xs:anyURI
own_comments [0..1] xs:anyURI
packaging_T
sequence
pack_description [0..1] description_T
length xs:unsignedLong
marketing_T
sequence
license_type xs:string
license [0..1] xs:string
icon_web [0..1] xs:anyURI
icon_tv [0..1] xs:anyURI
image [0..1] xs:anyURI
video [0..1] xs:anyURI
advertising [0..1] xs:anyURI
duration [0..1] xs:duration
date xs:date
date_update [0..1] xs:date
price [0..1] price_T
owner owner_T
certificate [0..1] xs:anyURI
related_app [0..*] xs:anyURI
marketing_annotation [0..1] description_T
description_T
sequence
notation xs:string
choice [1..2]
description xs:string
description_href xs:anyURI


composition_properties_T properties_T
sequence
composition xs:string
properties_T
sequence
name xs:string
value xs:string
disuse_T
sequence
option1 [0..1] xs:unsignedLong
option2 [0..1] xs:unsignedLong
option3 [0..1] xs:unsignedLong
option4 [0..1] xs:unsignedLong
option5 [0..1] xs:unsignedLong
price_T
sequence
currency xs:string
quantity xs:decimal
owner_T
sequence
author xs:string
company [0..1] xs:string
web [0..1] xs:anyURI
email xs:string
mhp
sequence
signed xs:boolean
plugin xs:boolean
plugin_interoperable xs:boolean
profile_name xs:string
profile_version xs:string
languages [0..*] xs:language
translations [0..1] description_T
android
sequence
osx
sequence

7. Gramática de las ofertas

XMLSchema para el modelo de ofertas en un trader para COTS en TDI.
Los tipos de servicio están almacenados en STR (Service Type Repository). Un componente, expresado como una oferta en el contexto del Trader hereda de un tipo de servicio.

```
<?xml version="1.0" encoding="utf-16"?>

<!-- ********************************************************************* -->
<!--                          XML Schema namespace                         -->
<!-- ********************************************************************* -->

<xs:schema
    id="RO_cots_tdi"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

    <!--
<xs:schema
    id="cots_tdt"
    targetNamespace="http://tempuri.org/XMLSchema.xsd"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/XMLSchema.xsd"
    xmlns:mstns="http://tempuri.org/XMLSchema.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    >
    -->


    <!-- ********************************************************************* -->
    <!--                                Annotations                            -->
    <!-- ********************************************************************* -->
    <xs:annotation>
        <xs:documentation>
            RO: Repository of Offers.
            XMLSchema para modelo de ofertas en un trader para COTS en TDI.
            Grupo de investigación UAL: INFORMÁTICA APLICADA.
            Maturana Espinosa, José Carmelo - Carmelo.Maturana@gmail.com
        </xs:documentation>
    </xs:annotation>


    <!-- ********************************************************************* -->
    <!--                                COTS TDI                               -->
    <!-- ********************************************************************* -->

    <xs:element name="serviceTDI">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="OfferName"        type="xs:string"/>
            <xs:choice minOccurs="1" maxOccurs="2">
                <xs:element name="location_in"      type="xs:anyURI"
default="null"/>
```

```
                <xs:element name="location_out"     type="xs:anyURI"
default="null"/>
            </xs:choice>
```

Información del servicio

```
            <xs:element name="functional"       type="functional_T"/>
            <xs:element name="no_functional"    type="no_functional_T"/>
            <xs:element name="qos"              type="qos_T"/>
            <xs:element name="packaging"        type="packaging_T"/>
            <xs:element name="marketing"        type="marketing_T"/>
        </xs:sequence>
    </xs:complexType>
    </xs:element>

    <!-- ********************************************************************* -->
    <!--                                 Types                                 -->
    <!-- ********************************************************************* -->

    <xs:complexType name="functional_T">
        <xs:sequence>
            <xs:element name="requirements" type="requirements_T"/>
            <!--<xs:element name="ports"    type="ports_T"/>-->

        </xs:sequence>

        <xs:attribute   name="category" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="Shop"/>
                    <xs:enumeration value="Communication"/>
                    <xs:enumeration value="Sports"/>
                    <xs:enumeration value="People"/>
                    <xs:enumeration value="Health"/>
                    <xs:enumeration value="Lifestyle"/>
                    <xs:enumeration value="Finance"/>
                    <xs:enumeration value="Tools"/>
                    <xs:enumeration value="Multimedia"/>
                    <xs:enumeration value="News and meteorology"/>
                    <xs:enumeration value="Learning"/>
                    <xs:enumeration value="Productivity"/>
                    <xs:enumeration value="Free time"/>
                    <xs:enumeration value="Reference"/>
                    <xs:enumeration value="Trips"/>
                    <xs:enumeration value="Themes"/>
                    <xs:enumeration value="Others"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>


    <xs:complexType name="requirements_T">
        <xs:sequence>
            <xs:element name="OS"                       type="xs:string"/>
            <xs:element name="requirements_description" type="description_T"
minOccurs="0"/>
```

```
            <xs:element name="requirements_enumeration" type="properties_T"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- ports en IR -->

    <xs:complexType name="no_functional_T">
        <xs:sequence>
            <xs:element name="inheritance"              type="xs:string"/>
            <!-- Específico para uso por el trader.-->
            <xs:element name="nf_description"           type="description_T"
minOccurs="0"/>
            <xs:element name="properties_composition"
type="composition_properties_T"     minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="properties"               type="properties_T"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="version"                  type="xs:string"
minOccurs="0"/>
            <xs:element name="url_update"               type="xs:anyURI"
minOccurs="0"/>
            <xs:element name="parental_control"         type="xs:string"/>
            <xs:element name="bat_dvb"                  type="xs:string"
minOccurs="0"/>
            <xs:choice minOccurs="0" maxOccurs="1">
                <xs:group ref="mhp"/>
                <xs:group ref="android"/>
                <xs:group ref="osx"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
```

La anterior secuencia se puede ampliar con cualquier sistema operativo con el que se puede completar la gramática

```
    <xs:complexType name="qos_T">
        <xs:sequence>
            <xs:element name="number_use"       type="xs:unsignedLong"
minOccurs="0"/>
            <xs:element name="number_payment"   type="xs:unsignedLong"
minOccurs="0"/>
            <xs:element name="point_own"        type="xs:unsignedByte"
minOccurs="0"/>
            <xs:element name="point_users"      type="xs:unsignedByte"
minOccurs="0"/>
            <xs:element name="motive_disuse"    type="disuse_T"
minOccurs="0"/>
            <xs:element name="user_comments"    type="xs:anyURI"
minOccurs="0"/>
            <xs:element name="own_comments"     type="xs:anyURI"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="packaging_T">
        <xs:sequence>
```

```
            <xs:element name="pack_description"     type="description_T"
minOccurs="0"/>
            <xs:element name="length"               type="xs:unsignedLong"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="marketing_T">
        <xs:sequence>
            <xs:element name="license_type">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="Proprietary|Shareware|Freeware|Free
Software"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="license"      type="xs:string" minOccurs="0"/>
            <xs:element name="icon_web"     type="xs:anyURI" minOccurs="0"/>
            <xs:element name="icon_tv"      type="xs:anyURI" minOccurs="0"/>
            <xs:element name="image"        type="xs:anyURI" minOccurs="0"/>
            <xs:element name="video"        type="xs:anyURI" minOccurs="0"/>
            <xs:element name="advertising"  type="xs:anyURI" minOccurs="0"/>

            <xs:element name="duration"     type="xs:duration"  minOccurs="0"/>
            <xs:element name="date"         type="xs:date"/>
            <xs:element name="date_update"  type="xs:date"      minOccurs="0"/>
            <xs:element name="price"        type="price_T"      minOccurs="0"/>
            <xs:element name="owner"        type="owner_T"/>
            <xs:element name="certificate"  type="xs:anyURI"    minOccurs="0"/>
            <xs:element name="related_app"  type="xs:anyURI"    minOccurs="0"
maxOccurs="unbounded"/>
            <!--type="xs:IDREFS" no aceptado por XML tipado de SQL Server-->
            <xs:element name="marketing_annotation" type="description_T"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>


    <!-- ********************************************************************* -->
    <!--                                Aux Types                              -->
    <!-- ********************************************************************* -->


    <xs:complexType name="description_T">
        <xs:sequence>
            <xs:element name="notation"             type="xs:string"/>
            <xs:choice minOccurs="1" maxOccurs="2">
                <xs:element name="description"      type="xs:string"/>
                <xs:element name="description_href" type="xs:anyURI"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="composition_properties_T">
        <xs:complexContent>
            <xs:extension base="properties_T">
                <xs:sequence>
```

```
                    <xs:element name="composition">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:pattern value="AND|OR"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>


    <xs:complexType name="properties_T">
        <xs:sequence>
            <xs:element name="name"     type="xs:string"/>
            <xs:element name="value"    type="xs:string"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="disuse_T">
        <xs:sequence>
            <xs:element name="option1"     type="xs:unsignedLong" minOccurs="0"
default="0"/>
            <xs:element name="option2"     type="xs:unsignedLong" minOccurs="0"
default="0"/>
            <xs:element name="option3"     type="xs:unsignedLong" minOccurs="0"
default="0"/>
            <xs:element name="option4"     type="xs:unsignedLong" minOccurs="0"
default="0"/>
            <xs:element name="option5"     type="xs:unsignedLong" minOccurs="0"
default="0"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="price_T">
        <xs:sequence>
            <xs:element name="currency" type="xs:string"/>
            <xs:element name="quantity" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="owner_T">
        <xs:sequence>
            <xs:element name="author"   type="xs:string"/>
            <xs:element name="company"  type="xs:string" minOccurs="0"/>
            <xs:element name="web"      type="xs:anyURI" minOccurs="0"/>
            <xs:element name="email"    type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
```

Código reservado para aplicaciones estándar MHP, de Apple, Google, Microsoft, etc para TV. Según el sistema operativo que usan y por tanto, su compatibilidad con receptores e interoperabilidad entre componentes.

```
    <!-- ********************************************************************** -->
    <!--                                Aux Group                               -->
    <!-- ********************************************************************** -->



    <xs:group name="mhp">
        <xs:annotation>
            <xs:documentation>
                Datos relevantes en el repositorio para los componentes con SO =
MHP.
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="signed"               type="xs:boolean"/>

            <xs:element name="plugin"               type="xs:boolean"/>
            <xs:element name="plugin_interoperable" type="xs:boolean"/>

            <xs:element name="profile_name"         type="xs:string"/>
            <xs:element name="profile_version"      type="xs:string"/>

            <xs:element name="languages"            type="xs:language"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="translations"         type="description_T"
minOccurs="0"/>
        </xs:sequence>
    </xs:group>
```

El siguiente código está reservado para aplicaciones de Apple, Google, Microsoft, etc para TV.

```
    <xs:group name="android">
        <xs:annotation>
            <xs:documentation>
                Datos relevantes en el repositorio para los componentes con OS =
ANDROID.
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
        </xs:sequence>
    </xs:group>


    <xs:group name="osx">
        <xs:annotation>
            <xs:documentation>
                Datos relevantes en el repositorio para los componentes con OS =
OSX (APPLE).
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>
        </xs:sequence>
    </xs:group>
</xs:schema>
```

8. Ejemplo de instanciación del modelo COTS

Aquí se escribe una instancia completa a modo de ejemplo para todo el modelo de ofertas desarrollado.

```
<?xml version="1.0" encoding="utf-8"?>
<serviceTDI>
  <OfferName>OfferName1</OfferName>
  <location_in>null</location_in>
  <location_out>null</location_out>
  <functional category="Shop">
    <requirements>
      <OS>OS1</OS>
      <requirements_description>
        <notation>notation1</notation>
        <description>description1</description>
        <description_href>http://uri1</description_href>
      </requirements_description>
      <requirements_enumeration>
        <name>name1</name>
        <value>value1</value>
      </requirements_enumeration>
      <requirements_enumeration>
        <name>name2</name>
        <value>value2</value>
      </requirements_enumeration>
      <requirements_enumeration>
        <name>name3</name>
        <value>value3</value>
      </requirements_enumeration>
    </requirements>
  </functional>
  <no_functional>
    <inheritance>inheritance1</inheritance>
    <nf_description>
      <notation>notation1</notation>
      <description>description1</description>
      <description_href>http://uri1</description_href>
    </nf_description>
    <properties_composition>
      <name>name1</name>
      <value>value1</value>
      <composition>composition1</composition>
    </properties_composition>
    <properties_composition>
      <name>name2</name>
      <value>value2</value>
      <composition>composition2</composition>
    </properties_composition>
    <properties_composition>
      <name>name3</name>
      <value>value3</value>
      <composition>composition3</composition>
    </properties_composition>
    <properties>
```

```
      <name>name1</name>
      <value>value1</value>
    </properties>
    <properties>
      <name>name2</name>
      <value>value2</value>
    </properties>
    <properties>
      <name>name3</name>
      <value>value3</value>
    </properties>
    <version>version1</version>
    <url_update>http://uri1</url_update>
    <parental_control>parental_control1</parental_control>
    <bat_dvb>bat_dvb1</bat_dvb>
    <signed>true</signed>
    <plugin>true</plugin>
    <plugin_interoperable>true</plugin_interoperable>
    <profile_name>profile_name1</profile_name>
    <profile_version>profile_version1</profile_version>
    <languages>en</languages>
    <languages>fr</languages>
    <languages>de</languages>
    <translations>
      <notation>notation1</notation>
      <description>description1</description>
      <description_href>http://uri1</description_href>
    </translations>
  </no_functional>
  <qos>
    <number_use>0</number_use>
    <number_payment>0</number_payment>
    <point_own>0</point_own>
    <point_users>0</point_users>
    <motive_disuse>
      <option1>0</option1>
      <option2>0</option2>
      <option3>0</option3>
      <option4>0</option4>
      <option5>0</option5>
    </motive_disuse>
    <user_comments>http://uri1</user_comments>
    <own_comments>http://uri1</own_comments>
  </qos>
  <packaging>
    <pack_description>
      <notation>notation1</notation>
      <description>description1</description>
      <description_href>http://uri1</description_href>
    </pack_description>
    <length>0</length>
  </packaging>
  <marketing>
    <license_type>license_type1</license_type>
    <license>license1</license>
    <icon_web>http://uri1</icon_web>
    <icon_tv>http://uri1</icon_tv>
    <image>http://uri1</image>
    <video>http://uri1</video>
```

```
    <advertising>http://uri1</advertising>
    <duration>P396DT1H1M1S</duration>
    <date>1900-01-01</date>
    <date_update>1900-01-01</date_update>
    <price>
      <currency>currency1</currency>
      <quantity>1</quantity>
    </price>
    <owner>
      <author>author1</author>
      <company>company1</company>
      <web>http://uri1</web>
      <email>email1</email>
    </owner>
    <certificate>http://uri1</certificate>
    <related_app>http://uri1</related_app>
    <related_app>http://uri2</related_app>
    <related_app>http://uri3</related_app>
    <marketing_annotation>
      <notation>notation1</notation>
      <description>description1</description>
      <description_href>http://uri1</description_href>
    </marketing_annotation>
  </marketing>
</serviceTDI>
```

Para la implementación del repositorio se barajaron las siguientes posibilidades:

- En primer lugar estudio como llevar a cabo una importación automatizada de datos provenientes de documentos escritos en XML según la gramática ideada en el capítulo 1 para este proyecto.
- El trato directamente con documentos XML usando XQuery. Se tomó la decisión a favor de ésta última.

El sistema gestor de bases de datos usado era en un principio MySQL [28]. Junto con un plugin para manejar consultas XQuery. Posteriormente trabajando con MySQL descarté su uso a favor de herramientas como SQL Server 2008 Enterprise gracias a un soporte nativo y más completo hacia XML. Concretamente puede ver parte de dicho soporte en las librerias de aprendizaje de Microsoft.

9. Software usado

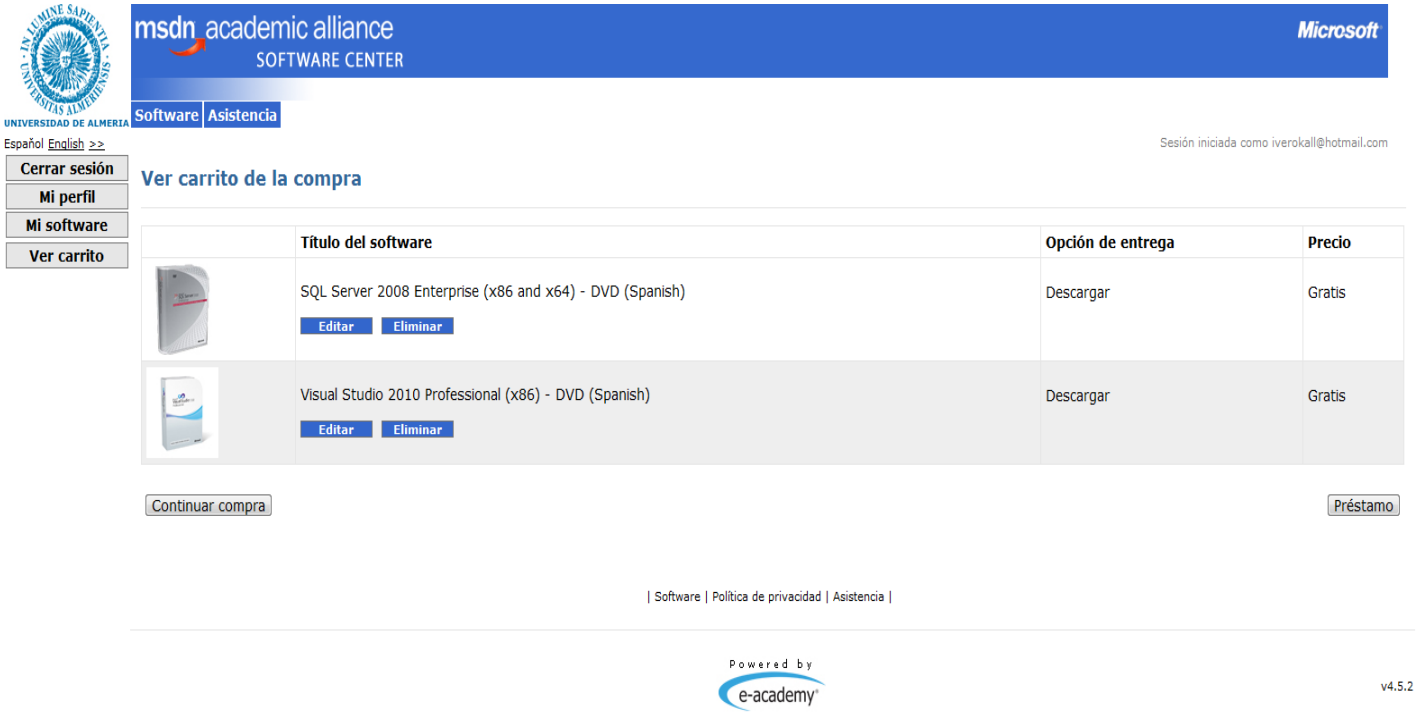
La implementación se ha llevado a cabo exclusivamente con tecnología .NET de Microsoft, haciendo uso de Alliance Microsoft-UAL (uso gratuito de software con razón docente o de investigación)

El software usado para el desarrollo ha sido: Visual Studio 2010 (C# y ASP.NET), SQL Server 2008 Enterprise (SQL y Transact-SQL), es necesaria la versión de empresas para hacer uso de funciones avanzadas de organización para una mayor optimización. FrameWork .NET 3.5 (la versión 4 era una beta en el momento de la implementación). Para el servidor: Windows Server 2008 (IIS incluido).

Se elegió el Visual Studio 2010 Professional porque está más indicado para desarrolladores individuales.

”Discover how Visual Studio 2010 Professional helps individual developers build, test, and debug software solutions“

El software necesario se consiguió en:
http://msdn30.e-academy.com/elms/Storefront/Home.aspx?campus=ual_stic



10. Diseño del repositorio

Para crear los repositorios se atendieron medidas de eficiencia y consistencia.

- **Eficiencia**: Las particiones permiten dividir de forma física una tabla en archivos. Los registros se van repartiendo entre los archivos según un valor del propio registro, por el criterio que se define en una función de partición y entre los archivos que indica un esquema de partición.
  Además se han indexado los documentos XML, bajo un índice primario, los índices secundarios son opcionales.

- **Consistencia**: XML tipados. Consiste en asociar una instancia XML con una gramática, de forma que el motor de BD siempre comprueba la consistente entre instancia y gramática en inserciones y modificaciones. En nuestro caso los XML se almacenan en columnas del mismo tipo y a éstas se asocia su gramática respectivamente.

El repositorio se ha creado en 4 pasos.

- Creación de la BD y archivos de partición [29]
- Creación de la colección de esquemas [30]

- Creación de los repositorios y
- Asociaciones para particiones, colecciones de esquemas e índices [31] usando la función de partición definida anteriormente y el esquema de partición.

10.1 Creación de la BD

El siguiente script reinicia la base de datos: eliminándola, creándola e iniciándola con la creación de archivos.

```
-- ============================================
-- Create database with filestream filegroups
-- ============================================

IF EXISTS (
  SELECT *
    FROM sys.databases
   WHERE name = 'pfc_repositorio'
)
  DROP DATABASE pfc_repositorio
GO


CREATE DATABASE pfc_repositorio

ON PRIMARY
       (NAME = repositorio_dat1,
         FILENAME = 'C:\REPOSITORIO\repositorio_dat1.mdf',
          SIZE = 10MB,
          MAXSIZE = 50MB,
          FILEGROWTH = 10%),

       ( NAME = repositorio_dat2,
         FILENAME = 'C:\REPOSITORIO\repositorio_dat2.ndf',
          SIZE = 10MB,
          MAXSIZE = 50MB,
          FILEGROWTH = 10%),
```

Grupos de archivos -fileGroup- para el repositorio RO (OFERTAS)

```
FILEGROUP RO_1
       ( NAME = RO1_dat1,
         FILENAME = 'C:\REPOSITORIO\RO1_dat1.ndf'),

FILEGROUP RO_2
       ( NAME = RO2_dat1,
         FILENAME = 'C:\REPOSITORIO\RO2_dat1.ndf'),

FILEGROUP RO_3
       ( NAME = RO3_dat1,
         FILENAME = 'C:\REPOSITORIO\RO3_dat1.ndf'),
```

```
FILEGROUP RO_4
       ( NAME = RO4_dat1,
         FILENAME = 'C:\REPOSITORIO\RO4_dat1.ndf'),

FILEGROUP RO_5
       ( NAME = RO5_dat1,
         FILENAME = 'C:\REPOSITORIO\RO5_dat1.ndf'),

FILEGROUP RO_6
       ( NAME = RO6_dat1,
         FILENAME = 'C:\REPOSITORIO\RO6_dat1.ndf'),

FILEGROUP RO_7
       ( NAME = RO7_dat1,
         FILENAME = 'C:\REPOSITORIO\RO7_dat1.ndf'),

FILEGROUP RO_8
       ( NAME = RO8_dat1,
         FILENAME = 'C:\REPOSITORIO\RO8_dat1.ndf'),

FILEGROUP RO_9
       ( NAME = RO9_dat1,
         FILENAME = 'C:\REPOSITORIO\RO9_dat1.ndf'),

FILEGROUP RO_10
       ( NAME = RO10_dat1,
         FILENAME = 'C:\REPOSITORIO\RO10_dat1.ndf'),

FILEGROUP RO_11
       ( NAME = RO11_dat1,
         FILENAME = 'C:\REPOSITORIO\RO11_dat1.ndf'),

FILEGROUP RO_12
       ( NAME = RO12_dat1,
         FILENAME = 'C:\REPOSITORIO\RO12_dat1.ndf'),

FILEGROUP RO_13
       ( NAME = RO13_dat1,
         FILENAME = 'C:\REPOSITORIO\RO13_dat1.ndf'),

FILEGROUP RO_14
       ( NAME = RO14_dat1,
         FILENAME = 'C:\REPOSITORIO\RO14_dat1.ndf'),

FILEGROUP RO_15
       ( NAME = RO15_dat1,
         FILENAME = 'C:\REPOSITORIO\RO15_dat1.ndf'),

FILEGROUP RO_16
       ( NAME = RO16_dat1,
         FILENAME = 'C:\REPOSITORIO\RO16_dat1.ndf'),

FILEGROUP RO_17
       ( NAME = RO17_dat1,
         FILENAME = 'C:\REPOSITORIO\RO17_dat1.ndf'),

FILEGROUP RO_18
       ( NAME = RO18_dat1,
         FILENAME = 'C:\REPOSITORIO\RO18_dat1.ndf'),
```

Grupos de archivos para DATOS SECUNDARIOS DE OFERTAS.

```
FILEGROUP fileGroup_101
       ( NAME = fileGroup101_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup101_dat1.ndf'),

FILEGROUP fileGroup_102
       ( NAME = fileGroup102_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup102_dat1.ndf'),

FILEGROUP fileGroup_103
       ( NAME = fileGroup103_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup103_dat1.ndf'),

FILEGROUP fileGroup_104
       ( NAME = fileGroup104_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup104_dat1.ndf'),

FILEGROUP fileGroup_105
       ( NAME = fileGroup105_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup105_dat1.ndf'),

FILEGROUP fileGroup_106
       ( NAME = fileGroup106_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup106_dat1.ndf'),

FILEGROUP fileGroup_107
       ( NAME = fileGroup107_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup107_dat1.ndf'),

FILEGROUP fileGroup_108
       ( NAME = fileGroup108_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup108_dat1.ndf'),

FILEGROUP fileGroup_109
       ( NAME = fileGroup109_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup109_dat1.ndf'),

FILEGROUP fileGroup_110
       ( NAME = fileGroup110_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup110_dat1.ndf'),

FILEGROUP fileGroup_111
       ( NAME = fileGroup111_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup111_dat1.ndf'),

FILEGROUP fileGroup_112
       ( NAME = fileGroup112_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup112_dat1.ndf'),

FILEGROUP fileGroup_113
       ( NAME = fileGroup113_dat1,
         FILENAME = 'C:\REPOSITORIO\fileGroup113_dat1.ndf'),

FILEGROUP fileGroup_114
       ( NAME = fileGroup114_dat1,
```

```
        FILENAME = 'C:\REPOSITORIO\fileGroup114_dat1.ndf'),

FILEGROUP fileGroup_115
      ( NAME = fileGroup115_dat1,
        FILENAME = 'C:\REPOSITORIO\fileGroup115_dat1.ndf'),

FILEGROUP fileGroup_116
      ( NAME = fileGroup116_dat1,
        FILENAME = 'C:\REPOSITORIO\fileGroup116_dat1.ndf'),

FILEGROUP fileGroup_117
      ( NAME = fileGroup117_dat1,
        FILENAME = 'C:\REPOSITORIO\fileGroup117_dat1.ndf'),

FILEGROUP fileGroup_118
      ( NAME = fileGroup118_dat1,
        FILENAME = 'C:\REPOSITORIO\fileGroup118_dat1.ndf'),
```

Grupos de archivos para el repositorio IR (Interfaces de componentes)

```
FILEGROUP IR_201
      ( NAME = IR201_dat1,
        FILENAME = 'C:\REPOSITORIO\IR201_dat1.ndf'),

FILEGROUP IR_202
      ( NAME = IR202_dat1,
        FILENAME = 'C:\REPOSITORIO\IR202_dat1.ndf'),

FILEGROUP IR_203
      ( NAME = IR203_dat1,
        FILENAME = 'C:\REPOSITORIO\IR203_dat1.ndf'),

FILEGROUP IR_204
      ( NAME = IR204_dat1,
        FILENAME = 'C:\REPOSITORIO\IR204_dat1.ndf'),

FILEGROUP IR_205
      ( NAME = IR205_dat1,
        FILENAME = 'C:\REPOSITORIO\IR205_dat1.ndf'),

FILEGROUP IR_206
      ( NAME = IR206_dat1,
        FILENAME = 'C:\REPOSITORIO\IR206_dat1.ndf'),

FILEGROUP IR_207
      ( NAME = IR207_dat1,
        FILENAME = 'C:\REPOSITORIO\IR207_dat1.ndf'),

FILEGROUP IR_208
      ( NAME = IR208_dat1,
        FILENAME = 'C:\REPOSITORIO\IR208_dat1.ndf'),

FILEGROUP IR_209
      ( NAME = IR209_dat1,
        FILENAME = 'C:\REPOSITORIO\IR209_dat1.ndf'),
```

```
FILEGROUP IR_210
       ( NAME = IR210_dat1,
         FILENAME = 'C:\REPOSITORIO\IR210_dat1.ndf'),

FILEGROUP IR_211
       ( NAME = IR211_dat1,
         FILENAME = 'C:\REPOSITORIO\IR211_dat1.ndf'),

FILEGROUP IR_212
       ( NAME = IR212_dat1,
         FILENAME = 'C:\REPOSITORIO\IR212_dat1.ndf'),

FILEGROUP IR_213
       ( NAME = IR213_dat1,
         FILENAME = 'C:\REPOSITORIO\IR213_dat1.ndf'),

FILEGROUP IR_214
       ( NAME = IR214_dat1,
         FILENAME = 'C:\REPOSITORIO\IR214_dat1.ndf'),

FILEGROUP IR_215
       ( NAME = IR215_dat1,
         FILENAME = 'C:\REPOSITORIO\IR215_dat1.ndf'),

FILEGROUP IR_216
       ( NAME = IR216_dat1,
         FILENAME = 'C:\REPOSITORIO\IR216_dat1.ndf'),

FILEGROUP IR_217
       ( NAME = IR217_dat1,
         FILENAME = 'C:\REPOSITORIO\IR217_dat1.ndf'),

FILEGROUP IR_218
       ( NAME = IR218_dat1,
         FILENAME = 'C:\REPOSITORIO\IR218_dat1.ndf')



-- Archivos de log.
LOG ON
       ( NAME = repositorio_log,
         FILENAME = 'C:\REPOSITORIO\repositorio_log1.ldf',
          SIZE = 10MB,
          MAXSIZE = 50MB,
          FILEGROWTH = 10%)
GO
```

Resulta el siguiente conglomerado de archivos para la Base de Datos de todo el sistema.

Se crean 18 archivos para 17 categorías posibles (uno de más para futuros usos) para cada repositorio donde compete (no para STR). La BD está compuesta por 57 elementos totales: 2 organización de la BD + 1 log + 18 archivos físicos para RO + 18 DataRO + 18 IR.

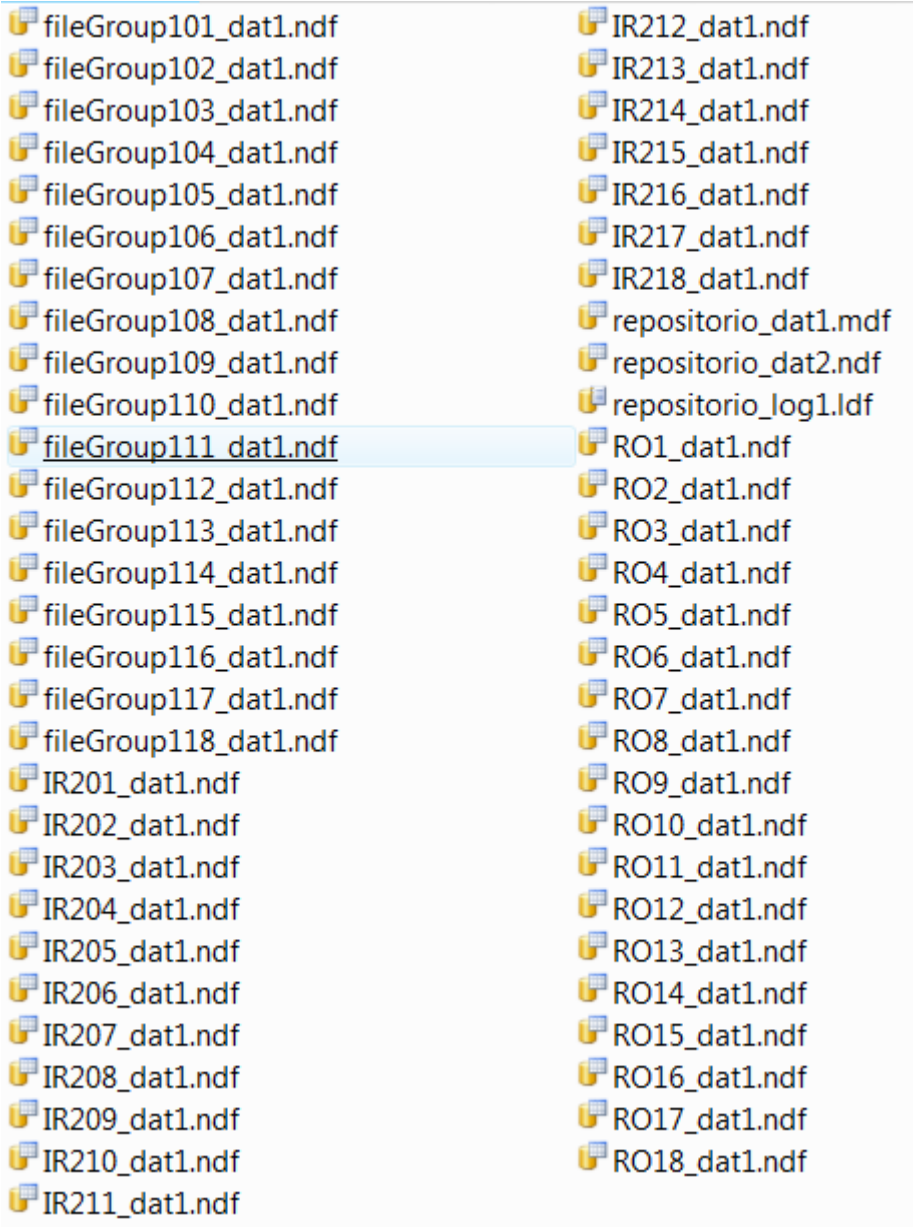


Figura 11: Particiones físicas del repositorio

10.2 Colección de gramáticas

Registro de la gramática XML que define su contenido en cada repositorio. Aseguran la consistencia de los datos en los documentos XML independientemente de alteraciones externas al conjunto de repositorios.

```
-- ============================================
-- Create XSD Collection of schemes
-- ============================================

USE pfc_repositorio
```

```
GO

IF EXISTS (SELECT *
             FROM sys.xml_schema_collections
            WHERE name = 'XSD_RO')
DROP XML SCHEMA COLLECTION XSD_RO
GO

IF EXISTS (SELECT *
             FROM sys.xml_schema_collections
            WHERE name = 'XSD_IR')
DROP XML SCHEMA COLLECTION XSD_IR
GO

IF EXISTS (SELECT *
             FROM sys.xml_schema_collections
            WHERE name = 'XSD_STR')
DROP XML SCHEMA COLLECTION XSD_STR
GO

IF EXISTS (SELECT *
             FROM sys.xml_schema_collections
            WHERE name = 'XSD_comments_ro')
DROP XML SCHEMA COLLECTION XSD_comments_ro
GO



CREATE XML SCHEMA COLLECTION XSD_RO AS
N'<?xml version="1.0" encoding="utf-16"?>
…
</xs:schema>';
GO

CREATE XML SCHEMA COLLECTION XSD_IR AS
…

CREATE XML SCHEMA COLLECTION XSD_STR AS
…

CREATE XML SCHEMA COLLECTION XSD_comments_ro AS
…
```

Verificación de la colección de esquemas en la BD. Y verificación de espacios de nombres en la BD.

```
select *
from sys.xml_schema_collections

select name
from sys.xml_schema_namespaces
```

10.3 Asignación lógica de particiones en archivos físicos

El siguiente script particiona físicamente (en archivos distintos) un repositorio. El usuario interactuará con un sólo repositorio lógico.
La ventaja consiste en el aumento de rendimiento al organizar de forma física los datos.

Creación de la función de partición y esquemas de partición para los repositorios (excepto STR que no compete). El discriminante para la función de partición es el valor del elemento "categoría" de la plantilla del componente (agrupaciones de funcionalidad). Puesto que se usa en todos los repositorios el mismo, sólo hay una función de partición. En cada repositorio se usa dicha función para asociar con el esquema de partición cada registro de cada repositorio a un archivo específico.

```
-- ========================================
-- Create partitions
-- ========================================

/*     CATEGORY
       ------------------
       1      'Shop',
       2      'Communication',
       3      'Sports',
       4      'People',
       5      'Health',
       6      'Lifestyle',
       7      'Finance',
       8      'Tools',
       9      'Multimedia',
       10     'News and meteorology',
       11     'Learning',
       12     'Productivity',
       13     'Free time',
       14     'Reference',
       15     'Trips',
       16     'Themes',
       17     'Others'
       ------------------
*/
-- ========================================

USE pfc_repositorio
GO
```

Función de partición según CATEGORÍA.

```
CREATE PARTITION FUNCTION PF_category (int)
AS RANGE LEFT
FOR VALUES (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17);
GO
```

Esquema de partición para el repositorio de ofertas.

```
CREATE PARTITION SCHEME RO_categoryPS
AS PARTITION PF_category
TO (
	RO_1,
	RO_2,
	RO_3,
	RO_4,
	RO_5,
	RO_6,
	RO_7,
	RO_8,
	RO_9,
	RO_10,
	RO_11,
	RO_12,
	RO_13,
	RO_14,
	RO_15,
	RO_16,
	RO_17,
	RO_18
);
```

Esquema de partición para el repositorio de interfaces.

```
CREATE PARTITION SCHEME IR_categoryPS
AS PARTITION PF_category
TO (
	IR_201,
	IR_202,
	IR_203,
	IR_204,
	IR_205,
	IR_206,
	IR_207,
	IR_208,
	IR_209,
	IR_210,
	IR_211,
	IR_212,
	IR_213,
	IR_214,
	IR_215,
	IR_216,
	IR_217,
	IR_218
);
```

Esquema de partición para el repositorio de datos secundarios de ofertas.

```
CREATE PARTITION SCHEME DataRO_categoryPS
AS PARTITION PF_category
```

```
TO (
      fileGroup_101,
      fileGroup_102,
      fileGroup_103,
      fileGroup_104,
      fileGroup_105,
      fileGroup_106,
      fileGroup_107,
      fileGroup_108,
      fileGroup_109,
      fileGroup_110,
      fileGroup_111,
      fileGroup_112,
      fileGroup_113,
      fileGroup_114,
      fileGroup_115,
      fileGroup_116,
      fileGroup_117,
      fileGroup_118
);
```

10.4 Creación de los repositorios

Creación de los 4 repositorios totales en los que se estructura el diseño del presente proyecto para una mediación de un trader sobre ofertas de componentes en televisión digital.

Según el planteamiento expuesto (vease gráfico en la memoria) la retaila de tecnologías usadas para aumentar el rendimiento del sistema distribuido entre varios repositorios con distintas características y necesidades, gestionados por el trader, a su vez encapsulado en un servicio web xml, son aplicadas en consonancia. Estas tecnologías para favorecer el rendimiento son la partición e indexación. En auge de otras ventajas está el uso de XML tipado o con tipo, ahora sí, en todos los repositorios (todos los repositorios contienen documentos XML).

Existe una documentación adicional en la que me he apoyado para la escritura del siguiente código, se encuentra en documentación MSDN
http://msdn.microsoft.com/es-es/library/ms174979.aspx

```
-- ========================================
-- Create table
-- ========================================

USE pfc_repositorio
GO
```

El siguiente código elimina los repositorios existentes anteriormente con motivo de crear un entorno inicial limpio.

```
IF OBJECT_ID('DataRO', 'U') IS NOT NULL
  DROP TABLE DataRO
GO

IF OBJECT_ID('IR', 'U') IS NOT NULL
  DROP TABLE IR
GO

IF OBJECT_ID('RO', 'U') IS NOT NULL
  DROP TABLE RO
GO

IF OBJECT_ID('STR', 'U') IS NOT NULL
  DROP TABLE "STR"
GO
```

10.4.1 RO: Repositorio de ofertas

El repositorio de las ofertas se constituye por un identificador secuencial, la categoría del componente y el campo available (indica si la oferta está disponible o deshabilitada). Estos elementos pertenecen al SGBD y no al componente en sí, por ello es un dato del repositorio y no un elemento del documento XML.
Es reseñable que dichos documentos XML se encuentran tipados como aparece en el código.
Ademas se crea un indice de tipo xml exigiendo que su discriminador sea parte de la clave en la tabla que indexa.

El código “clustered” indica que se ha creado un índice agrupado para la restricción PRIMARY KEY.
Usando la función de partición definida anteriormente y el esquema de partición almacena las particiones del repositorio (elemento lógico) en archivos físicos distintos. Por último, se indexan las ofertas escritas en documentos XML.

```
----------------------------------
-- Crea repositorio de PLANTILLAS.
----------------------------------
CREATE TABLE RO(
      id_service          int IDENTITY not null,
      category_range      int not null,
      available           int,
      template            XML (XSD_RO) not null,
      CONSTRAINT k_RO     PRIMARY KEY CLUSTERED (id_service, category_range)

)
```

```
      ON RO_categoryPS (category_range);
      GO

      CREATE PRIMARY XML INDEX index_XML_RO
      ON RO(template);
      GO
```

10.4.2 IR: Repositorio de interfaces

El siguiente código crea el repositorio dedicado al almacenaje de las interfaces de los componentes que serán tratados por el Trader en forma de ofertas.

```
-------------------------
-- Crea repositorio de IR.
-------------------------
CREATE TABLE IR(
      id_service_ref     int not null,
      ranges_ref         int not null,
      available          int,
      interface          XML (XSD_IR) not null,
      FOREIGN KEY (id_service_ref, ranges_ref) REFERENCES RO(id_service,
category_range)
)

      ON IR_categoryPS (ranges_ref);
      GO

      CREATE INDEX index_IR
    ON IR (id_service_ref)
    ON IR_categoryPS (ranges_ref);     GO
```

Los índices se encuentran particionados según el mismo esquema de partición que su tabla.
El presente repositorio se encuentra relacionado con el repositorio de ofertas por medio de una clave externa para referencia.
El campo available indica si la interfaz está disponible. Pertenece al SGBD y no al elemento en sí, por ello es un dato del repositorio y no un elemento del documento XML.

10.4.3 STR: Repositorio de tipos de servicio

La estructura del repositorio de tipos de servicio no requiere de particiones. Si es conveniente una indexación por nombre de tipos de servicios.

```
-------------------------
-- Crea repositorio de STR.
-------------------------
CREATE TABLE "STR"(
```

```
      nameServiceType           nvarchar(90) PRIMARY KEY not null,
      available                 int,
      ServiceType               XML (XSD_STR) not null,
      -- En argot de SQL Server se dice: XML tipado.
)

      CREATE INDEX index_STR
    ON "STR" (nameServiceType);
    GO
```

10.4.4 DATA RO: Repositorio de la capa de negocio de las ofertas

Es un repositorio que contiene datos no necesarios para la búsqueda o ejecución del componente. Cada conjunto de esta información conforma una entidad relacionada a la plantilla de su componente mediante una llave externa al ID de éste.

Los elementos en la tabla secundaria no estarán almacenados en XML ya que en su mayoría se trata de archivos binarios (imágenes, videos, códigos,…) por lo tanto, tendría que almacenar una dirección URI para referenciarlos pero no almacenarlos directamente. Por ello, la tabla contiene ya los campos a almacenar en caso de que se disponga de ellos. La gramática inicial no cambia.

El código se puede encontrar alojado en repositorio propio (<location_in>) o en algún sitio externo. Por otro lado también se define una clave externa para referencia a posibles datos secundarios de cada plantilla.

```
-----------------------------------------------------
-- Crea repositorio de DATOS SECUNDARIOS DE OFERTAS.
-----------------------------------------------------
CREATE TABLE DataRO(
      id_service_ref      int not null,
      ranges_ref          int not null,
      available           int,
      comments            XML (XSD_comments_ro) not null,
      license             nvarchar(max),
      icon_web            image,
      icon_tv             image,
      image_service       image,
      video               varbinary(8000),
      advertising         varbinary(8000),
      code                varbinary(8000),
      FOREIGN KEY (id_service_ref, ranges_ref) REFERENCES RO(id_service,
category_range)

)

      ON DataRO_categoryPS (ranges_ref);
      GO


      CREATE INDEX index_DataRO
```

```
    ON DataRO (id_service_ref)
    ON DataRO_categoryPS (ranges_ref);
      GO
```

El sistema de comentarios almacenados en el almacenamiento secundario llamado Data_RO se encuentra especificado por la siguiente gramática:

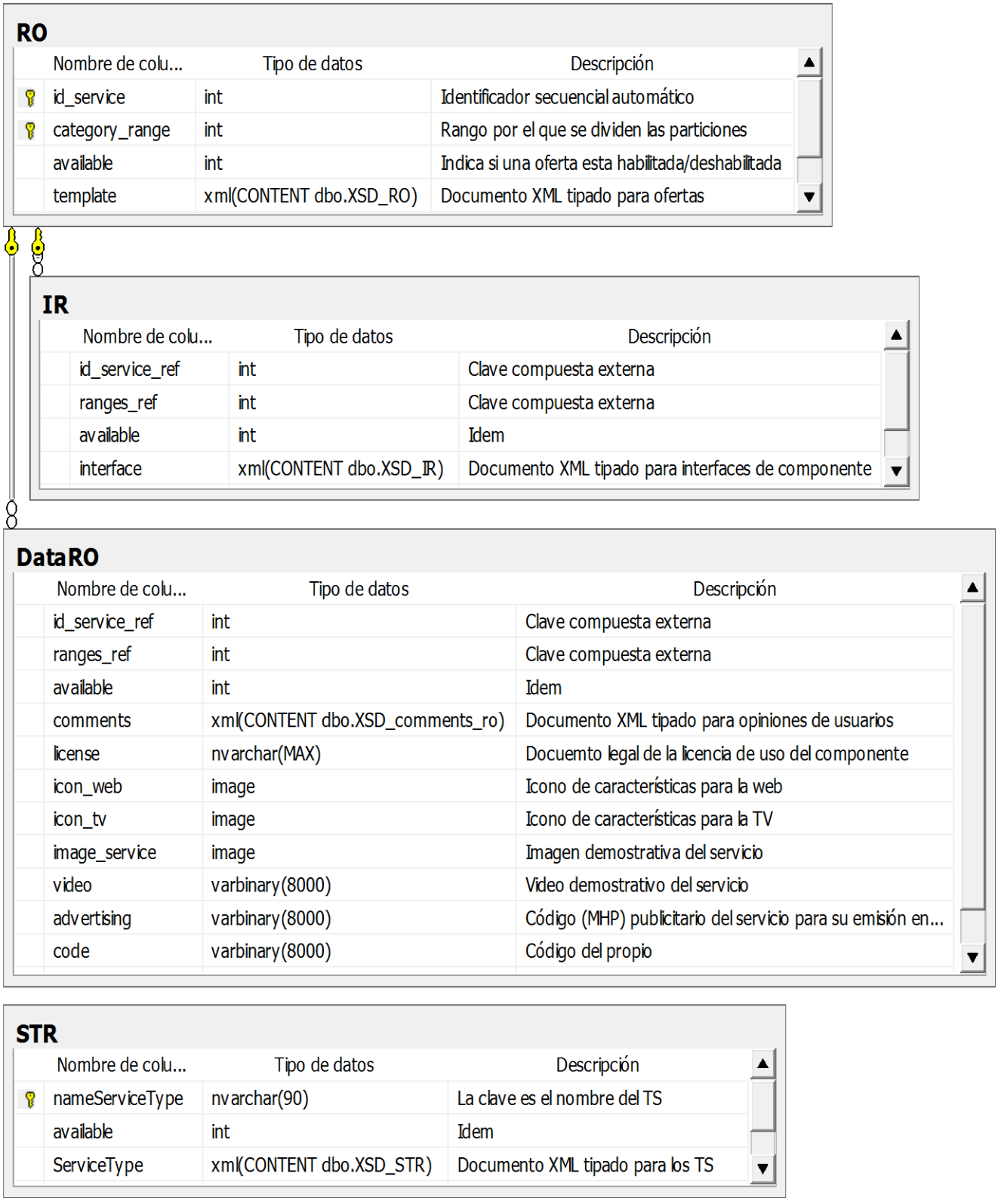
La implementación en código XML del sistema de comentarios es el siguiente:

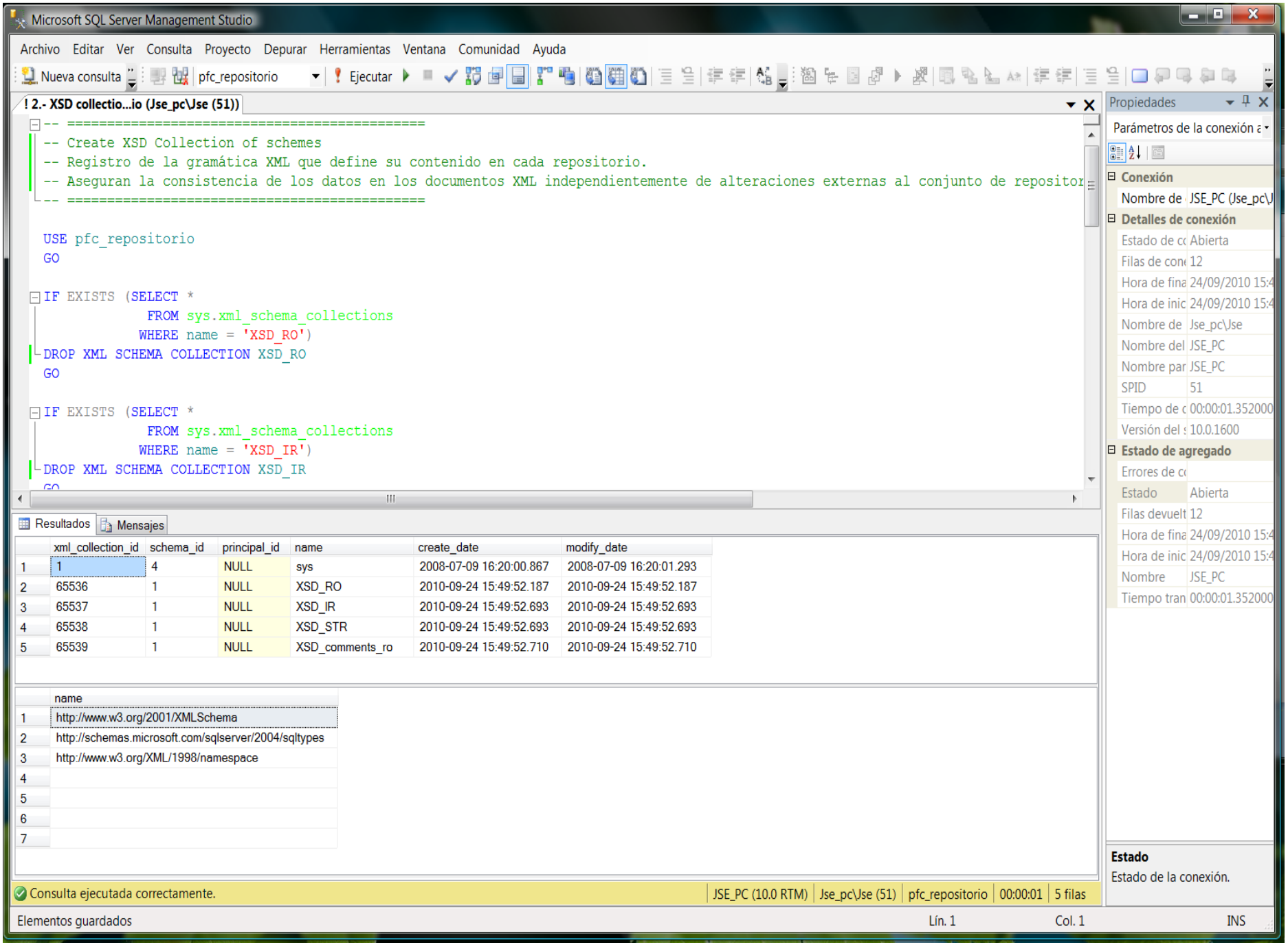
```
<?xml version="1.0" encoding="utf-16"?>

<!-- ********************************************************************* -->
<!--                           XML Schema namespace                        -->
<!-- ********************************************************************* -->

<xs:schema
    id="comments"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

    <!--
<xs:schema
    id="comments"
    targetNamespace="http://tempuri.org/XMLSchema.xsd"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/XMLSchema.xsd"
    xmlns:mstns="http://tempuri.org/XMLSchema.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
>
-->


    <!-- ********************************************************************* -->
    <!--                                Annotations                            -->
```

```
    <!-- ********************************************************************* -->
    <xs:annotation>
        <xs:documentation>
            XMLSchema para modelo de comentarios.
            Grupo de investigación UAL: INFORMÁTICA APLICADA.
            Maturana Espinosa, José Carmelo - Carmelo.Maturana@gmail.com
        </xs:documentation>
    </xs:annotation>


    <!-- ********************************************************************* -->
    <!--                                COMMENTS                               -->
    <!-- ********************************************************************* -->
    <xs:element name="comments" type="comments_T"/>

    <xs:complexType name="comments_T">
        <xs:sequence>
            <xs:element name="own_comments"     minOccurs="0"
maxOccurs="unbounded" type="comment_T"/>
            <xs:element name="user_comments"    minOccurs="0"
maxOccurs="unbounded" type="comment_T"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="comment_T">
        <xs:sequence>
            <xs:element name="author"   type="xs:string"    default="Anonymous"/>
            <xs:element name="contact"  type="xs:anyURI"    minOccurs="0"/>
            <xs:element name="date"     type="xs:date"/>
            <xs:element name="points"   type="xs:int"       default="0"/>
            <xs:element name="comment"  type="xs:string"/>
        </xs:sequence>
    </xs:complexType>


</xs:schema>
```

10.5 Resultado final de los repositorios

El resultado se representa en la siguiente figura número 12:





Captura del proceso

¿Porqué no se ha usado una columna calculada para el campo category_range?

La clave del repositorio RO es compuesta porque por definición la columna que es usada en una función de partición tiene que ser parte de la clave, ya que toda columna que participa en un esquema de partición tiene que ser parte de la clave, como es lógico. Si no se hubieran usado particiones sino simplemente crear tablas manualmente bajo el mismo criterio, el valor de "categoría" seguiría usándose para la organización pero de forma implícita, y para las consultas que no tuvieran en cuenta ese discriminante sería necesario crear vistas globales de todas las tablas con la consiguiente carga computacional. En este proyecto el uso de las particiones (como del resto de medidas) no complica el diseño, ni el desempeño, obteniendo sólo beneficios.

El siguiente código muestra la respuesta del compilador al ejecutar el codigo

```
]----------------------------------
 -- Crea repositorio de PLANTILLAS.
----------------------------------
]CREATE TABLE RO(
     id_service      int IDENTITY PRIMARY KEY,    -- identificador secuencial.
     category_range  as dbo.my_category(template), -- Cálculo de columna mediante el usc
     available       int,                         -- Indica si la oferta está disponible
]    template        XML (XSD_RO) not null,       -- XML tipado.
     --CONSTRAINT k_RO   PRIMARY KEY (id_service, category_range)
```

Mens. 7724, Nivel 16, Estado 1, Línea 6
No se puede usar una columna calculada como clave de partición si no es persistente. La columna de clave de partición 'category_range' de la tabla 'RO' no es persistente.

No se puede usar código DML sobre la instancia XML para extraer el valor del elemento "categoría" haciéndola una columna calculada, ya que es parte de la clave y sus datos tienen que ser persistentes. Por lo tanto, esta tarea se llevará a cabo por código Linq [32] desde el objeto cliente (liberando al servidor de carga innecesaria). La función hubiera podido ser:

```
-------------------------------------------------
-- Crea columna calculada a partir de columna XML.
-------------------------------------------------

USE pfc_repositorio
GO

--------------------------------
-- Elimina funciones existentes.
--------------------------------
IF  EXISTS (
    SELECT *
    FROM sys.objects
    WHERE object_id = OBJECT_ID(N'[dbo].[my_category]')
)
    DROP FUNCTION [dbo].[my_category]
GO

-------------------
-- Crea la función
```

```
--------------------
CREATE FUNCTION [my_category](@var xml) returns int
AS BEGIN
RETURN @var.value('(/servicetv/functional/@category)[1]' , 'int')
END
GO

------------------
-- VERIFICACIONES.
------------------
SELECT  *
FROM    sys.objects
GO
```

11. Gramática de tipos de servicio

XMLSchema para modelo de tipos de servicio en un trader para COTS en TDI. Los usuarios a los que se refiere este apartado son usuarios profesionales.

La herencia entre las 3 gramáticas principales (RO, IR y STR) no está justificada porque sólo heredarían elementos de bajo nivel de abstracción, es decir, se crearía un espacio de nombres carente de sentido semántico para el propósito de los mismos. Ej: "description_T".

El repositorio STR no se relaciona con la clave primaria de la base de datos ya que un componente puede tener herencia múltiple. Esto repercute en que no sea suficiente sacar de la plantilla xml de RO un campo "tipo de servicio" relacionado con STR. Entonces, los ST serán tratados extrapolando a la POO como clases, cuya eliminación del repositorio no debe comprometer la consistencia del RO.

En la especificación OMG trader, nombre de la interfaz IDL [1] que describe la firma de cómputo de los servicios anunciados recibe la denominación de "IDL".
El presente proyecto permite cualquier definición de interfaz en un servicio, pero no en un tipo de servicio. Sirva esto para homogeneizar los tipos de servicio.

La herencia entre tipos de servicio representa el mayor exponente de intermediación entre componentes en el trader, ya que gracias a ella se pueden combinar construyendo nuevos servicios más complejos gracias a la colaboración entre componentes.

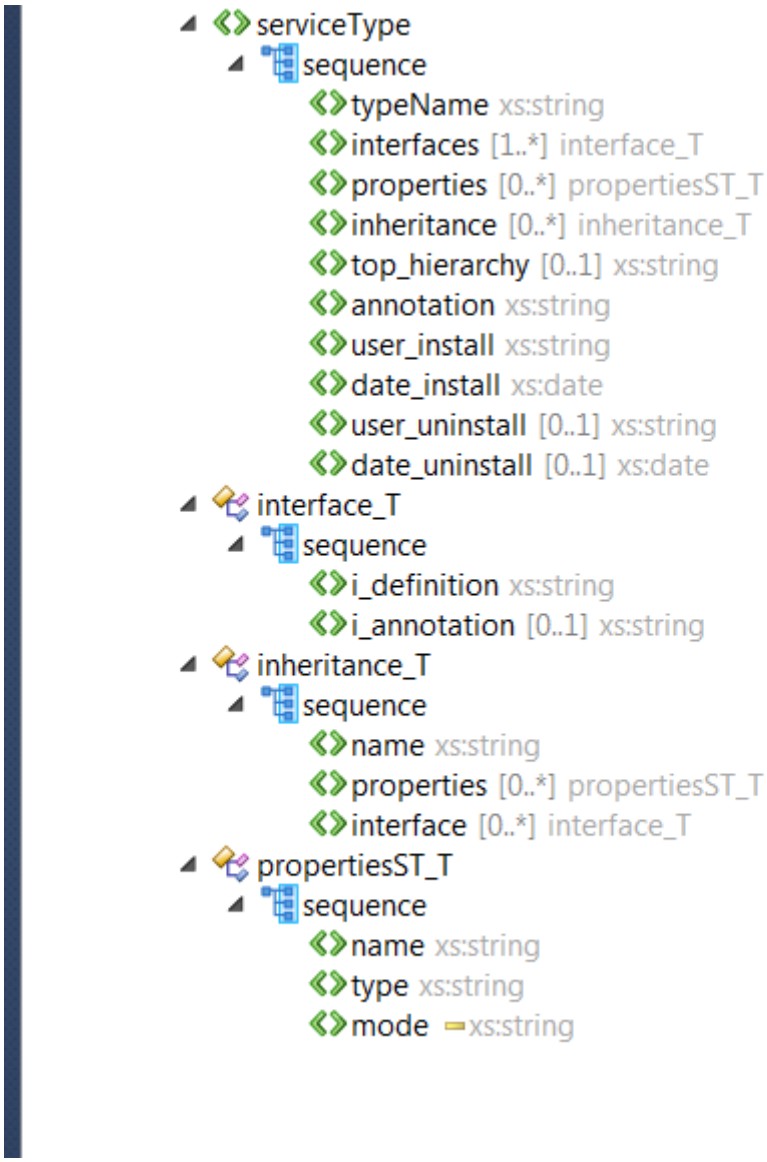
En la jerarquía de tipos de servicio "top_hierarchy" indica un tipo de servicio superior en la jerarquía. Esto representa la otra mitad de la funcionalidad del trader, al poder ofrecer resultados alternativos a una consulta explícita. Esta jerarquía de carácter semántico es establecida por los profesionales que registran un tipo de servicio en el STR.

El elemento "annotation" de valor semántico para comprender el tipo de servicio. Es una descripción en lenguaje natural sobre el tipo de servicio, que escribe el propio usuario que lo registra**. "i_annotation"** de valor semántico para comprender la interfaz del tipo de servicio

Quién lo ha creado, en qué fecha y **cuándo** se ha desinstalado. Estos últimos elementos constituyen la información para la gestión de calidad de los tipos de servicio.
El presente proyecto no contempla una típica gestión de usuarios, centrándose en el objetivo de crear un trader competente.

Por último las propiedades se establecen según la declaración oficial de **OMG Trader**: "*Each property declaration is marked by the keyword “property” and may be preceded by mode attributes “mandatory” and/or “readonly*".

A continuación se presenta de forma gráfica la gramática desarrollada:



La implementación para conseguir este resultado es la siguiente:

```
<?xml version="1.0" encoding="utf-16"?>

<!-- ********************************************************************** -->
<!--                          XML Schema namespace                          -->
<!-- ********************************************************************** -->

<xs:schema
    id="STR_cots_tdi"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

    <!--
<xs:schema
    id="st_cots_tdt"
    targetNamespace="http://tempuri.org/XMLSchema.xsd"
```

```
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/XMLSchema.xsd"
    xmlns:mstns="http://tempuri.org/XMLSchema.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    >
    -->

    <!-- ********************************************************************** -->
    <!--                                Annotations                             -->
    <!-- ********************************************************************** -->
    <xs:annotation>
        <xs:documentation>
            STR: Service Type Repository.
            XMLSchema para modelo de tipos de servicio en un trader para COTS en
TDI.
            Grupo de investigación UAL: INFORMÁTICA APLICADA.
            Maturana Espinosa, José Carmelo - Carmelo.Maturana@gmail.com
        </xs:documentation>
    </xs:annotation>


    <!-- ********************************************************************** -->
    <!--                              ST COTS TDI                               -->
    <!-- ********************************************************************** -->

    <!--

    -->

    <xs:element name="serviceType">
    <xs:complexType>
        <xs:sequence>
```

Específicos del STR según OMG, definición de trader.

```
            <xs:element name="typeName"         type="xs:string"/>
            <xs:element name="interfaces"       type="interface_T"
minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="properties"       type="propertiesST_T"
minOccurs="0" maxOccurs="unbounded"/>
```

No específicos del STR según OMG, definición de trader.

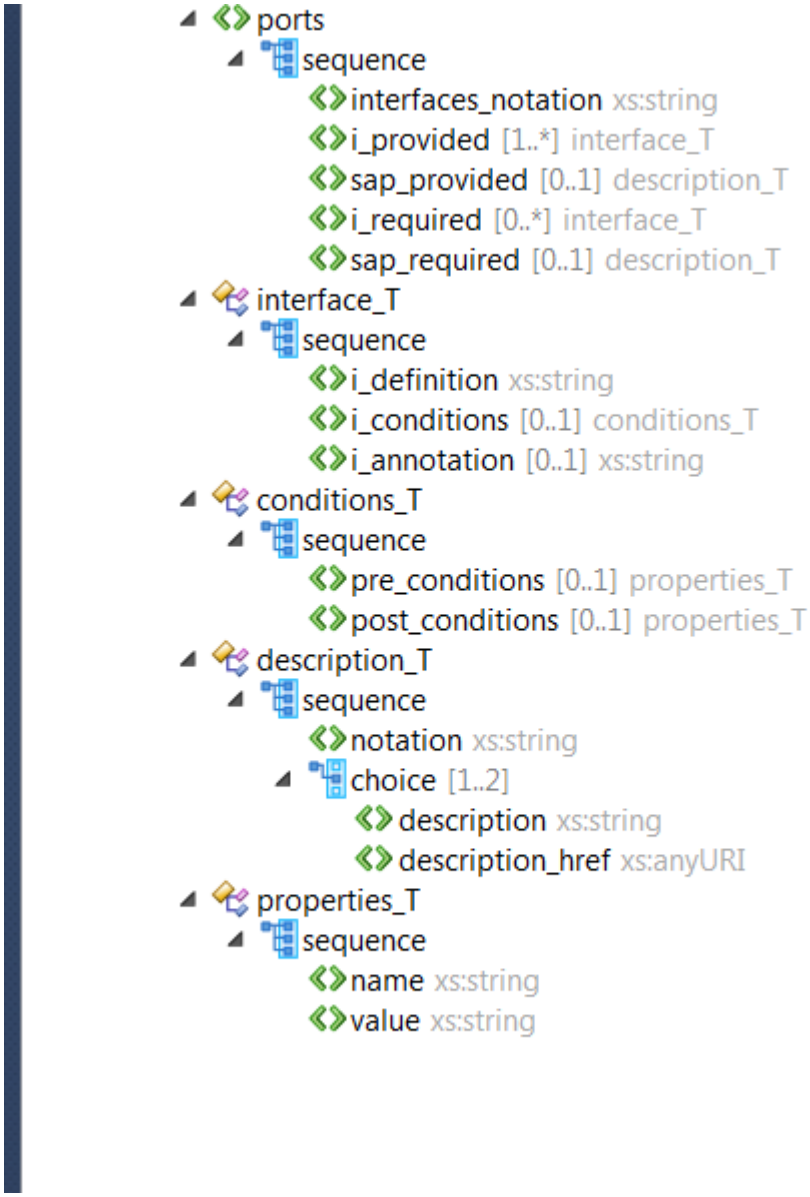
```
            <xs:element name="inheritance"      type="inheritance_T"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="top_hierarchy"    type="xs:string"
minOccurs="0" maxOccurs="1"/>
            <xs:element name="annotation"       type="xs:string"
minOccurs="1"/>
            <xs:element name="user_install"     type="xs:string"/>
            <xs:element name="date_install"     type="xs:date"/>
            <xs:element name="user_uninstall"   type="xs:string"
minOccurs="0"/>
            <xs:element name="date_uninstall"   type="xs:date"
minOccurs="0"/>
```

```
        </xs:sequence>
    </xs:complexType>
    </xs:element>


    <!-- ********************************************************************* -->
    <!--                                Types                                  -->
    <!-- ********************************************************************* -->

    <xs:complexType name="interface_T">
        <xs:sequence>
            <xs:element name="i_definition"     type="xs:string"/>
            <xs:element name="i_annotation"     type="xs:string"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="inheritance_T">
        <xs:sequence>
            <xs:element name="name"         type="xs:string"/>
            <xs:element name="properties"   type="propertiesST_T"   minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="interface"    type="interface_T"      minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="propertiesST_T">
        <xs:sequence>

            <xs:element name="name"     type="xs:string"/>
            <xs:element name="type"     type="xs:string"/>
            <xs:element name="mode">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="mandatory|readonly|mandatory and
readonly"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>


</xs:schema>
```

12. Gramática de las interfaces

XMLSchema para modelo de interfaces en un trader para COTS en TDI.

Las interfaces se enumeran en orden, se puede especificar unas restricciones determinadas con un documento SAP = service_access_protocol. Además es compatible con la especificación OMG de TRADER en el apartado que enuncia: “The trading function specification accepts the definition of interfaces in OMG IDL.”



La implementación en código XML para conseguir la gramática diseñada es:

```
<?xml version="1.0" encoding="utf-16"?>

<!-- ********************************************************************* -->
<!--                          XML Schema namespace                         -->
<!-- ********************************************************************* -->

<xs:schema
    id="IR_cots_tdi"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

    <!--
<xs:schema
    id="cots_tdt"
    targetNamespace="http://tempuri.org/XMLSchema.xsd"
    elementFormDefault="qualified"
    xmlns="http://tempuri.org/XMLSchema.xsd"
    xmlns:mstns="http://tempuri.org/XMLSchema.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    >
    -->

    <!-- ********************************************************************* -->
    <!--                                Annotations                            -->
    <!-- ********************************************************************* -->
    <xs:annotation>
        <xs:documentation>
            IR: Interface Repository.
            XMLSchema para modelo de interfaces en un trader para COTS en TDI.
            Grupo de investigación UAL: INFORMÁTICA APLICADA.
            Maturana Espinosa, José Carmelo - Carmelo.Maturana@gmail.com
        </xs:documentation>
    </xs:annotation>


    <!-- ********************************************************************* -->
    <!--                           PORTS COTS TDI                              -->
    <!-- ********************************************************************* -->

    <xs:element name="ports">
    <xs:complexType>

        <xs:sequence>

            <xs:element name="interfaces_notation"  type="xs:string"
default="IDL"/>

            <xs:element name="i_provided"           type="interface_T"
minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="sap_provided"         type="description_T"
minOccurs="0"/>

            <xs:element name="i_required"           type="interface_T"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="sap_required"         type="description_T"
minOccurs="0"/>
```

```
        </xs:sequence>
    </xs:complexType>
    </xs:element>



    <!-- ********************************************************************* -->
    <!--                                Types                                  -->
    <!-- ********************************************************************* -->


    <xs:complexType name="interface_T">
        <xs:sequence>
            <xs:element name="i_definition" type="xs:string"/>
            <xs:element name="i_conditions" type="conditions_T" minOccurs="0"/>
            <xs:element name="i_annotation" type="xs:string"    minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="conditions_T">
        <xs:sequence>
            <xs:element name="pre_conditions"    type="properties_T"
minOccurs="0"/>
            <xs:element name="post_conditions"   type="properties_T"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="description_T">
        <xs:sequence>
            <xs:element name="notation"             type="xs:string"/>
            <xs:choice minOccurs="1" maxOccurs="2">
                <xs:element name="description"      type="xs:string"/>
                <xs:element name="description_href" type="xs:anyURI"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="properties_T">
        <xs:sequence>
            <xs:element name="name"     type="xs:string"/>
            <xs:element name="value"    type="xs:string"/>
        </xs:sequence>
    </xs:complexType>


</xs:schema>
```

13. Instancias de ejemplo de componentes

Ejemplos de tipos de servicio instanciados y registrados usando la interfaz REGISTER del Trader desde el servicio Web. Estas instancias servirán para la comprobación del funcionamiento del Trader, así como de las premisas implementadas de herencias multiples y simples, jerarquía, etc.

Los ejemplos son los siguientes para tipos de servicio:

- Calculadora naturales.

```
<serviceType>
    <typeName>calculadora naturales</typeName>
    <interfaces>
        <i_definition>definición de interfaz del tipo de servicio</i_definition>
    </interfaces>
    <properties>
        <name>propiedad_naturales</name>
        <type>tipo</type>
        <mode>readonly</mode>
    </properties>
    <properties>
        <name>propiedad_naturales2</name>
        <type>tipo</type>
        <mode>readonly</mode>
    </properties>

    <annotation>Es una descripción en lenguaje natural sobre el tipo de servicio
"calculadora de naturales"</annotation>

    <user_install>usuario que registra el ST</user_install>
    <date_install>2010-09-12</date_install>

</serviceType>
```

- Calculadora enteros

```
<serviceType>
    <typeName>calculadora enteros</typeName>
    <interfaces>
        <i_definition>definición de interfaz del tipo de servicio</i_definition>
    </interfaces>
    <properties>
        <name>propiedad_enteros</name>
        <type>tipo</type>
        <mode>readonly</mode>
    </properties>
    <properties>
        <name>propiedad2_enteros</name>
        <type>tipo</type>
        <mode>readonly</mode>
    </properties>
```

```
    <inheritance>
        <name>calculadora naturales</name>
    </inheritance>
    <annotation>Es una descripción en lenguaje natural sobre el tipo de servicio
"calculadora de enteros"</annotation>

    <user_install>usuario que registra el ST</user_install>
    <date_install>2010-09-12</date_install>

</serviceType>
```

- Calculadora enteros 8bits

```
<serviceType>
    <typeName>calculadora enteros 8bits</typeName>
    <interfaces>
        <i_definition>definición de interfaz del tipo de servicio</i_definition>
    </interfaces>
    <properties>
        <name>propiedad_enteros8bits</name>
        <type>tipo</type>
        <mode>readonly</mode>
    </properties>
    <properties>
        <name>propiedad2_enteros8bits</name>
        <type>tipo</type>
        <mode>readonly</mode>
    </properties>

    <top_hierarchy>calculadora enteros</top_hierarchy>

    <annotation>Es una descripción en lenguaje natural sobre el tipo de servicio
"calculadora de enteros de 8 bits"</annotation>

    <user_install>usuario que registra el ST</user_install>
    <date_install>2010-09-12</date_install>

</serviceType>
```

Los ejemplos son los siguientes para las ofertas de dichos tipos de servicio:

- Tu calculadora enteros

```
<serviceTDI>
    <OfferName>tu calculadora de enteros</OfferName>
    <location_out>http:\\servidorexterno.com</location_out>
    <functional category="Tools">
        <requirements>
            <OS>mhp</OS>
        </requirements>
    </functional>

    <no_functional>
        <inheritance>calculadora enteros</inheritance>
```

```
        <parental_control>TV-Y7</parental_control>
    </no_functional>

    <qos>
    </qos>

    <packaging>
        <length>800</length>
    </packaging>

    <marketing>
        <license_type>Proprietary</license_type>
        <date>2010-08-01</date>
        <date_update>2010-08-01</date_update>
        <owner>
            <author>Carmelo</author>
            <email>Carmelo.Maturana@gmail.com</email>
        </owner>
    </marketing>

</serviceTDI>
```

- Tu calculadora enteros 8 bits

```
<serviceTDI>
    <OfferName>tu calculadora de enteros 8 bits</OfferName>
    <location_out>http:\\servidorexterno.com</location_out>
    <functional category="Tools">
        <requirements>
            <OS>mhp</OS>
        </requirements>
    </functional>

    <no_functional>
        <inheritance>calculadora enteros 8 bits</inheritance>
        <properties>
            <name>propiedad2_enteros 8 bits</name>
            <value>un valor</value>
        </properties>

        <parental_control>TV-Y7</parental_control>
    </no_functional>

    <qos>
    </qos>

    <packaging>
        <length>800</length>
    </packaging>

    <marketing>
        <license_type>Proprietary</license_type>
        <date>2010-08-01</date>
        <date_update>2010-08-01</date_update>
        <owner>
            <author>Carmelo</author>
            <email>Carmelo.Maturana@gmail.com</email>
```

```
        </owner>
    </marketing>

</serviceTDI>
```

- Tu otra calculadora enteros 8 bits

Esta oferta con el mismo nombre en una de sus propiedades pero distinto valor. Con objeto de jugar con la búsqueda de ofertas a través de la interfaz LOOKUP.

```
<serviceTDI>
    <OfferName>tu otra calculadora de enteros 8 bits</OfferName>
    <location_out>http:\\servidorexterno.com</location_out>
    <functional category="Tools">
        <requirements>
            <OS>mhp</OS>
        </requirements>
    </functional>

    <no_functional>
        <inheritance>calculadora enteros 8 bits</inheritance>
        <properties>
            <name>propiedad2_enteros 8 bits</name>
            <value>otro valor</value>
        </properties>

        <parental_control>TV-Y7</parental_control>
    </no_functional>

    <qos>
    </qos>

    <packaging>
        <length>800</length>
    </packaging>

    <marketing>
        <license_type>Proprietary</license_type>
        <date>2010-08-01</date>
        <date_update>2010-08-01</date_update>
        <owner>
            <author>Carmelo</author>
            <email>Carmelo.Maturana@gmail.com</email>
        </owner>
    </marketing>

</serviceTDI>
```

- Tu calculadora de naturales

```
<serviceTDI>
    <OfferName>tu calculadora de naturales</OfferName>
    <location_out>http:\\servidorexterno.com</location_out>
    <functional category="Tools">
        <requirements>
            <OS>mhp</OS>
```

```
        </requirements>
    </functional>

    <no_functional>
        <inheritance>calculadora naturales</inheritance>
        <parental_control>TV-Y7</parental_control>
    </no_functional>

    <qos>
    </qos>

    <packaging>
        <length>800</length>
    </packaging>

    <marketing>
        <license_type>Proprietary</license_type>
        <date>2010-08-01</date>
        <date_update>2010-08-01</date_update>
        <owner>
            <author>Carmelo</author>
            <email>Carmelo.Maturana@gmail.com</email>
        </owner>
    </marketing>

</serviceTDI>
```

PROPIEDAD EXTRAÑA. Es usada una propiedad que no es heredada de ningún tipo de servicio con el fin de comprobar el funcionamiento de la consistencia de la herencia al registrar ofertas.

```
<?xml version="1.0"?>

<serviceTDI>
    <OfferName>tu calculadora de enteros</OfferName>
    <location_out>http:\\servidorexterno.com</location_out>
    <functional category="Tools">
        <requirements>
            <OS>mhp</OS>
        </requirements>
    </functional>

    <no_functional>
        <inheritance>calculadora enteros</inheritance>
        <properties>
            <name>propiedad_extraña</name>
            <type>tipo</type>
            <mode>readonly</mode>
        </properties>
        <parental_control>TV-Y7</parental_control>
    </no_functional>

    <qos>
    </qos>

    <packaging>
```

```
        <length>800</length>
    </packaging>

    <marketing>
        <license_type>Proprietary</license_type>
        <date>2010-08-01</date>
        <date_update>2010-08-01</date_update>
        <owner>
            <author>Carmelo</author>
            <email>Carmelo.Maturana@gmail.com</email>
        </owner>
    </marketing>

</serviceTDI>
```

14. Trader básico

La especificación oficial de Trader tomada como base de estudio en el presente proyecto puede ser consultada en la biblioteca de OMG [33]. Además del artículo que concreta de forma sencilla algunos aspectos del mismo [4].

Interfaces del trader: register y lookup

A modo de resumen las dos interfaces básicas en un Trader consisten en que un cliente puede tener en el proceso de negociación uno de los siguientes papeles:

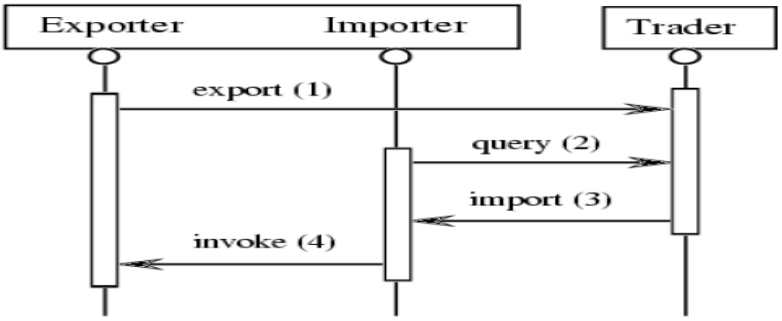


Figura 13: Procesos de negociación en un Trader

a) El papel de exportador (Register). Un objeto adopta el papel de exportador cuando se debe anunciar un servicio al sistema de objetos. Esto se logra por medio del método de la exportación() del objeto comerciante, estableciendo los parámetros para que se puedan llevar a cabo el almacenamiento del servicio en su repositorio.

b) El papel de importador (Lookup). Un objeto adopta el papel de importador, si requiere ciertos servicios almacenados en el repositorio del comerciante. Esto se logra por medio de la query() método del objeto comerciante, consultando los parámetros establecidos en su registro.

15. Trader modificado

A partir de ahora al hablar sobre trader, el texto se refiere a nuestra interpretación de la especificación OMG Trader. Para comprender cómo funciona hay que saber qué es un tipo de servicio (TS). Un TS es una entidad, con propiedades, que representa a un conjunto de funcionalidades muy similares. Un TS puede heredar de uno o varios TS.

Aquí radica una parte de la potencia del trader: creación de tipos de TS a partir de otros, o lo que es lo mismo, cooperación de componentes, es decir, la mediación entre componentes para conseguir un servicio mayor por colaboración. Finalmente un componente en el trader es una "oferta de servicio". Toda oferta hereda de un TS, su representación de funcionalidad y sus propiedades. Por ejemplo, un componente-A que suma números reales, podría heredar de un TS llamado "sumador de reales". Otro B que los resta, heredaría de "restador de reales". Ambos tipos de servicio podrían unirse en otro TS llamado "calculadora simple de reales" heredando de los TS anteriores. El código en este TS estaría formado por ambos componentes dichos.
La granularidad de los TS se va definiendo conforme se crean por parte de los usuarios.

La otra parte de la potencia del trader consiste en la jerarquía entre TS. Esto es, al registrar un TS se puede especificar otro que semánticamente hablando sea de funcionalidad similar y más amplia. Por ejemplo, un desarrollador crea un componente con la funcionalidad (especializada por cualquier motivo) de sumar y restar números enteros. Lo registra en el trader bajo el TS de "calculadora simple de enteros", sería conveniente que indicara al TS "calculadora simple de reales" como superior en la jerarquía. De este modo, cuando un cliente pregunte al trader si tiene algún servicio de calculadora de enteros, si la oferta de componentes del TS "…enteros" no se puede usar porque ya no esté disponible o sea muy cara, el sistema puede ofrecer la alternativa de las ofertas disponibles del TS "…reales" con un porcentaje de certeza elevado, de que este otro tipo de servicio le puede ser útil al cliente.

Como resumen, la potencia del trader estriba en la organización de los componentes software entre los que media, a través de relacionar componentes con TS y organizar estos últimos, estableciéndose relaciones entre ellos que permiten al trader manejarlos. Por último se comprueba la interoperabilidad entre dos componentes para que sus códigos puedan ejecutarse usando la información resultante del anterior obteniendo progresivamente un resultado de mayor riqueza. La información de interoperabilidad está asociada a cada oferta y se encuentra en el repositorio IR.

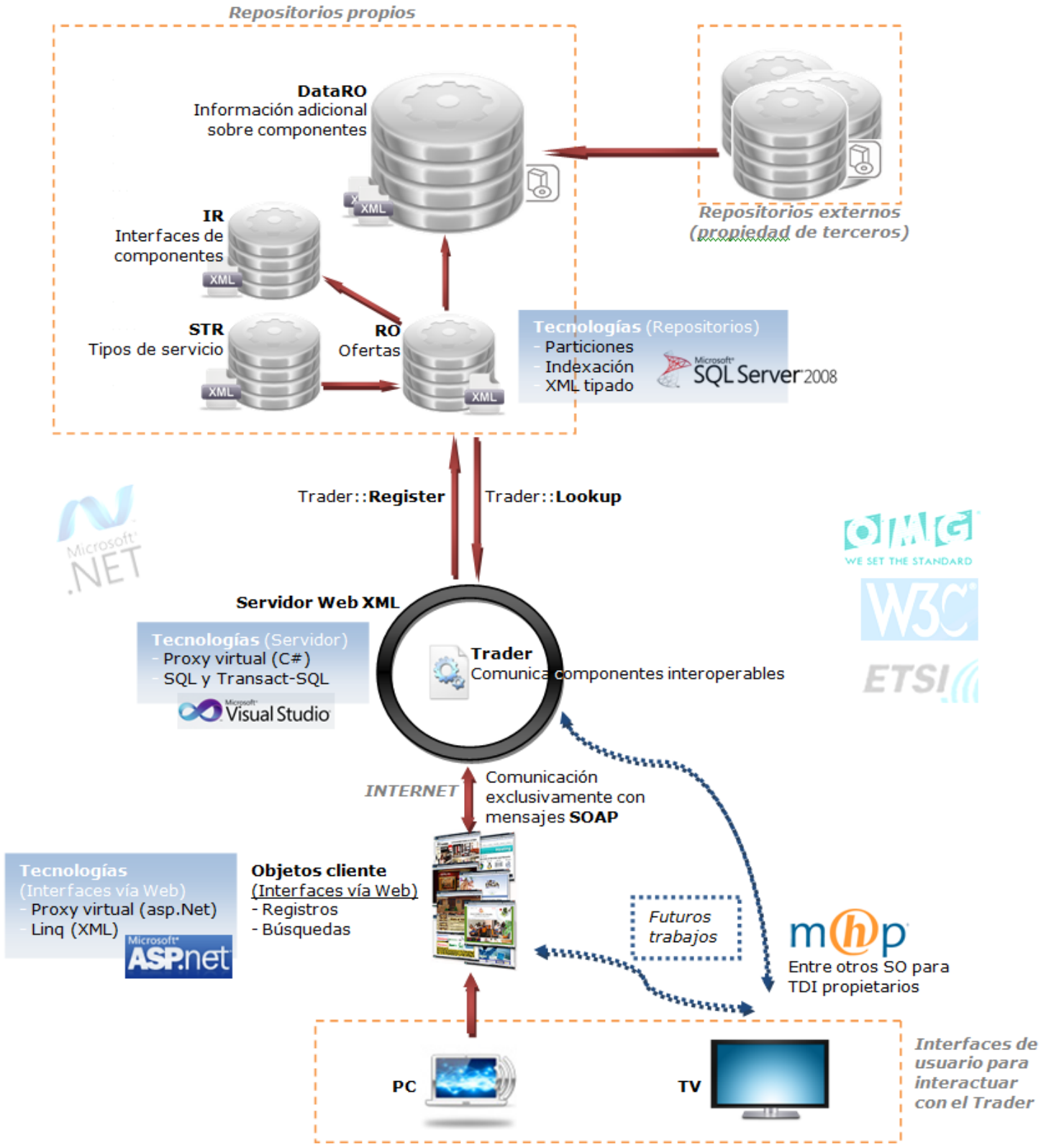
Un objeto trader trabaja con tres tipos de repositorios independientes:

- **RO** (Offers of Repository). Depósito de las ofertas donde los desarrolladores registran sus componentes como instancias de servicios de los TS.

- **IR** (Interfaz Repository). Repositorio que almacena las interfaces de los componentes ofertados. Una capacidad o servicio exportado por un objeto cliente debe tener asociado una interfaz computacional que responde a la funcionalidad del servicio anunciado. La especificación trader acepta la definición de interfaces en OMG IDL.

- **STR** (Service Type Repository). Depósito para los tipos de servicio.

Se ha añadido DataRO con la información adicional de las ofertas, que no es necesaria para el funcionamiento del trader pero si para el mercado de negocio. Esta clase de información se nombra en la especificación general sobre trader, pero no se específica donde guardarse. Por ejemplo, el propio código del componente (que puede estar en el repositorio propio o en alguno externo, ajeno a nuestro sistema), licencias de uso, etc.
(Véase el diagrama del sistema completo en la siguiente figura)



La figura muestra la estructura de todo el sistema junto con las herramientas y estándares usados. Las flechas indican el flujo habitual de información.

Políticas del trader

La principal responsabilidad de un trader es satisfacer las búsquedas de ofertas de los clientes, las cuales se centran en TS y 3 temas:

- Constraints. Expresión booleana escrita en lenguaje OCL. Las propiedades de un tipo de servicio tienen un valor en las ofertas que lo heredan. De esta forma la oferta especifica al TS.

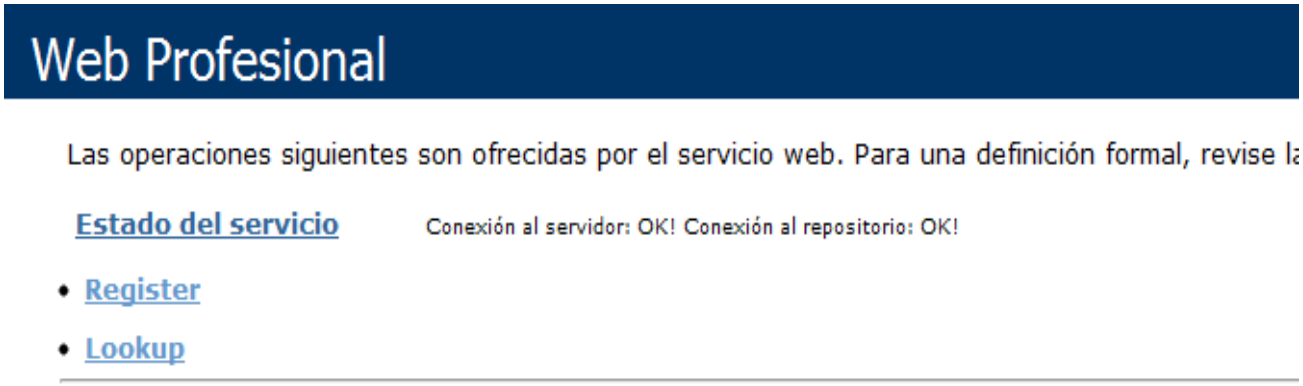
- Preference. Indica el **orden** en el que el cliente quiere que se le devuelvan las ofertas: max (ascendente), min (descendiente), with (bajo condición), random (al azar) y first (en el orden natural). Por ejemplo, considerando una oferta, con una preferencia "with (rango == coronel)" significa que el comerciante devolverá en primer lugar las ofertas con esa propiedad a tal valor, después de evaluar la búsqueda con las condiciones impuestas.
En la definición de Trader, el orden lo establece él, sin embargo en nuestra implementación hemos decidido que algunas preferencias (max y min) las realice el objeto cliente, ya que de esta forma el usuario puede interactuar en tiempo real para tener varias vistas de los resultados de la búsqueda. La paginación de los resultados también se lleva a cabo en el obejto cliente.

- Policies. Son las políticas que adaptan la búsqueda. Por ejemplo, hay políticas que limitan la **cardinalidad** de las ofertas involucradas en el proceso de búsqueda del trader. De manera que el usuario puede elegir que se le devuelvan sólo los primeros X resultados encontrados.

El usuario tiene que beneficiarse y percibir las ventajas que aporta el Trader. Para ello se ha implementado especialmente unos controles que le permiten usar la jerarquía como “búsqueda ampliada”. Hace uso de la jerarquía entre tipos de servicio haciendo ver al usuario que se hace una búsqueda según sus criterios originales pero obteniendo un mayor resultado.

16. Estado del servicio

Es una interfaz del servidor Web que permite comprobar la operatividad del sistema. Lleva a cabo los mecanismos de comunicación (protocolos y mensajes SOAP) entre el objeto cliente, el servidor Web donde se encuentra el Trader y el repositorio. De forma que se comprueba que todo está comunicado correctamente.



17. Register

Comprende el registro de tipos de servicio y ofertas en el repositorio que maneja el trader.

17.1 XML tipado

Se lleva a cabo al registrar un tipo de servicio u oferta una comprobación de la sintáxis del documento XML que se pretende registrar. Desde el repositorio se envía a través del servico Web un mensaje de corrección específico al XML enviado en caso de que sea necesario, por errores, omisiones, etc.

17.2 Integridad al registrar una oferta (HERENCIA SIMPLE)

Se lleva a cabo la comprobación del uso correcto de las posibles propiedades heredadas desde el tipo de servicio designado en la propia oferta. No se lleva a cabo una herencia automátizada ya que el usuario tiene que especificar valores concretos para la oferta y no tiene porqué hacerlo para todas las propiedades que puede llegar a heredar. El lector puede comprobarlo mediante el ejemplo de “propiedad extraña” anteriormente explicado.

Comprueba la consistencia en la HERENCIA SIMPLE del TS para una oferta. (Código preparado para multiherencia) Se puede llevar a cabo gracias al uso del lenguaje LINQ [32], embebido en ASP.NET, embebido a su vez en el HTML.
Por si le sirviera de aclaración al lector, LINQ posibilita operaciones del carácter de XML DOM [34].

No se lleva a cabo la herencia del TS de una oferta por la sencilla razón de que el usuario tiene que darle un valor.

```
            try
            {
                System.Collections.Generic.IEnumerable<XElement> enumeracion_st =
xmldoc.Element("serviceTDI").Elements("no_functional");
                foreach (XElement st in enumeracion_st)
                {
                    var ServiceType =
                        from c in st.Elements("inheritance")
                        select c.Value;
                    if (!listadoNombresTS.Contains(ServiceType.Single()))
                    {
                        ReturnError.Text = "El tipo de servicio: \"" +
ServiceType.Single() + "\" no está registrado.";
                        return;
                    }
                }
            }
            catch {
                ReturnError.Text = "Toda oferta debe heredar de un tipo de
servicio mediante el elemento 'inheritance' en las propiedades no funcionales.";
                return;
            }
```

17.3 Integridad al registrar un tipo de servicio (MULTI-HERENCIA)

Como ya se ha visto, un tipo de servicio puede heredar de uno o más tipos de servicios, es necesario que se realice una implementación de dicha herencia. Consiste en la copia de propiedades desde los tipos de servicio padres, al hijo. Para ello se usará código LINQ (véase el capítulo “Implementación”)

Comprueba la consistencia en la MULTIHERENCIA de tipos de sercivio para un nuevo tipo de servicio. Es decir, se comprueba que todo tipo de servicio herede de tipos de servicio existentes, en caso de que lo haga. Además se comprueba que al establecerse una jerarquía entre el presente tipo de servicio y otro, este último realmente exista en el catálogo del Trader.

```
            System.Collections.Generic.IEnumerable<XElement> enumeracion_st =
xmldoc.Element("serviceType").Elements("inheritance");
```

```
            foreach (XElement st in enumeracion_st)
            {
                var ServiceType =
                    from c in st.Elements("name")
                    select c.Value;
                if (!listadoNombresTS.Contains(ServiceType.Single()))
                {
                    ReturnError.Text = "No se puede heredar del tipo de servicio
\"" + ServiceType.Single() + "\" porque no se encuentra registrado.";
                    return;
                }
            }
            XElement a = xmldoc.Element("serviceType");
            XElement existe = a.Element("top_hierarchy");
            if (existe != null)
            {
                var ServiceType_padre =
                    from c in xmldoc.Elements("serviceType")
                    select c.Element("top_hierarchy").Value;
                if (!listadoNombresTS.Contains(ServiceType_padre.Single()))
                {
                    ReturnError.Text = "El tipo de servicio 'padre' indicado no
está registrado. Indique uno válido.";
                    return;
                }
            }
```

18. Lookup

La recuperación de ofertas solicitada desde el objeto cliente, se lleva a cabo en el Trader, quien lanza consultas en lenguaje **XQuery** [35] contra los repositorios.

Las consultas se describen según una serie de parámetros que intervienen antes de la consulta, con motivo de realizarla y obtener unos resultados deacuerdo a ellos. Y otros parámetros cuyo fin es clasificar o aclarar de cara al cliente los resultados que hemos recibido del Trader y ahora se encuentran en el objeto cliente. Así, se pueden clasificar en parámetros o criterios de pre y post consulta. Siendo los parámetros de post-consulta referidos únicamente a un subconjunto de ofertas cuya información se encuentra ya en el objeto cliente, sin necesidad de interactuar con el Trader, o éste con el repositorio.

Para más detalle

- "Parámetros **PRE-CONSULTA"** añade criterios de búsqueda que serán enviados al TRADER, quien realiza las consultas.
  Dichos criterios son: buscar por nombre, categoría, valor de propiedades, cardinalidad y búsqueda ampliada.

  - La cardinalidad consiste en dejar de buscar servicios cuando durante la búsqueda ya se dispone de un determinado nº de resultados.

  - La búsqueda ampliada consiste en realizar la consulta relajando los criterios de búsqueda especificados en caso de que no se obtenga ningún resultado con ellos.

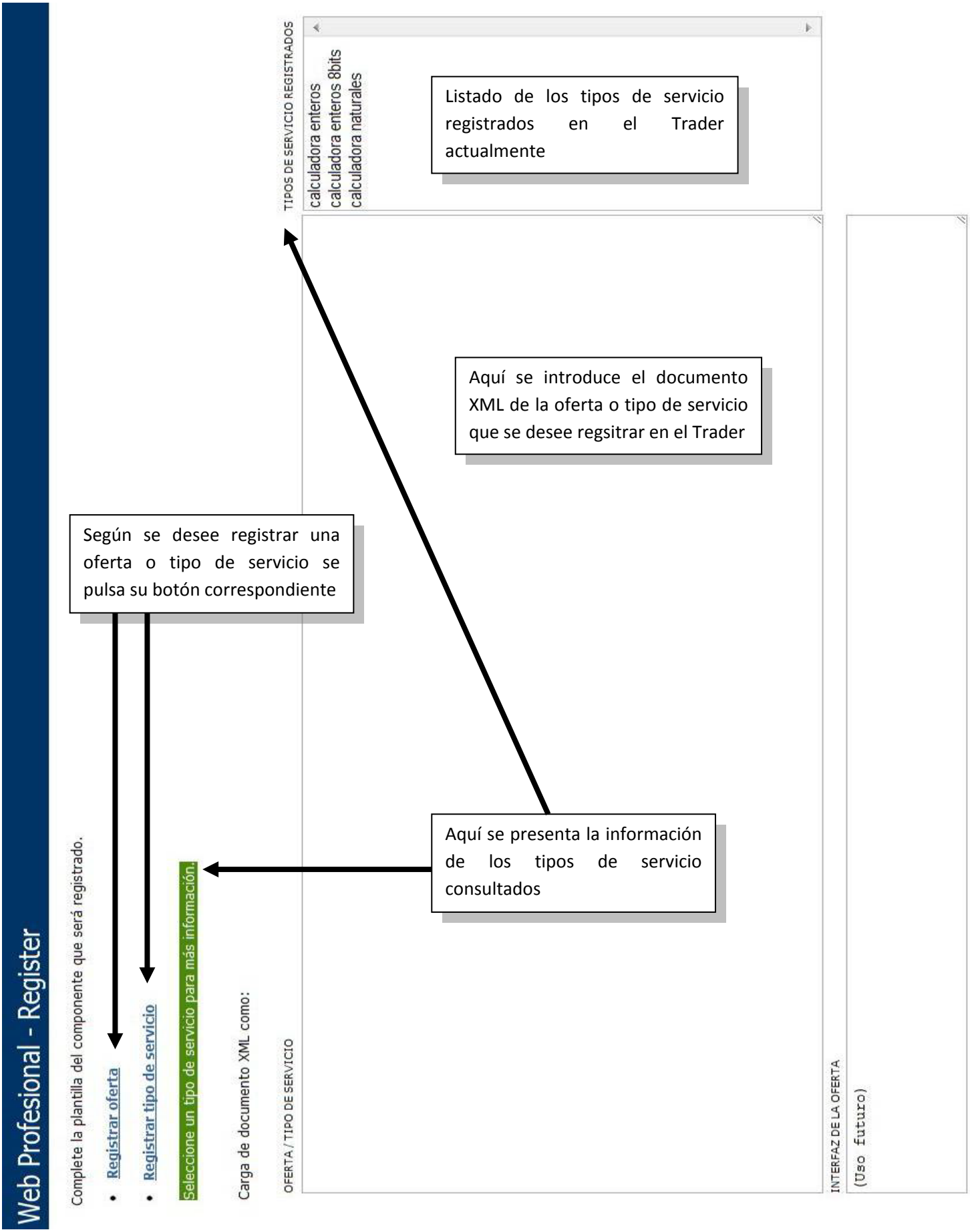
- "Parámetros **POST-CONSULTA**" adapta la visualización de los resultados obtenidos de la consulta ya residentes en el cliente. Este caso de uso no interactua con el Trader.
  Comprende la paginación y ordenación de los resultados según la característica que seleccione el cliente.

19. Web de navegación entre componentes

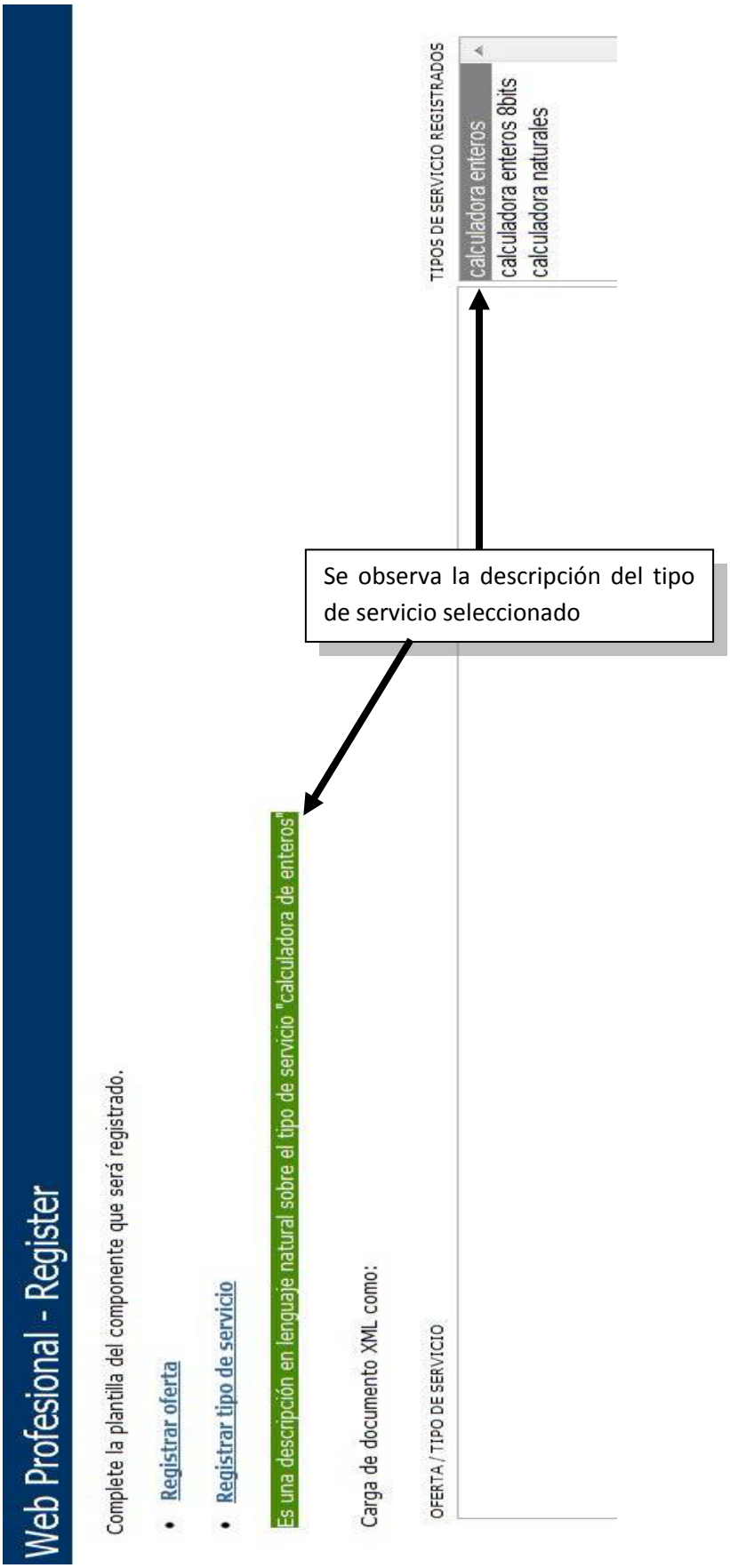
Web destinada al uso por parte de profesionales. Que demandan un componente para TD con determinados criterios de funcionalidad, o desea dar de alta su componente, publicitándolo.
Dispone de una conexión con un repositorio que mediante los servicios de mediación (trader):

Las siguientes capturas muestran la funcionalidad de las interfaces web con descripciones de sus partes:

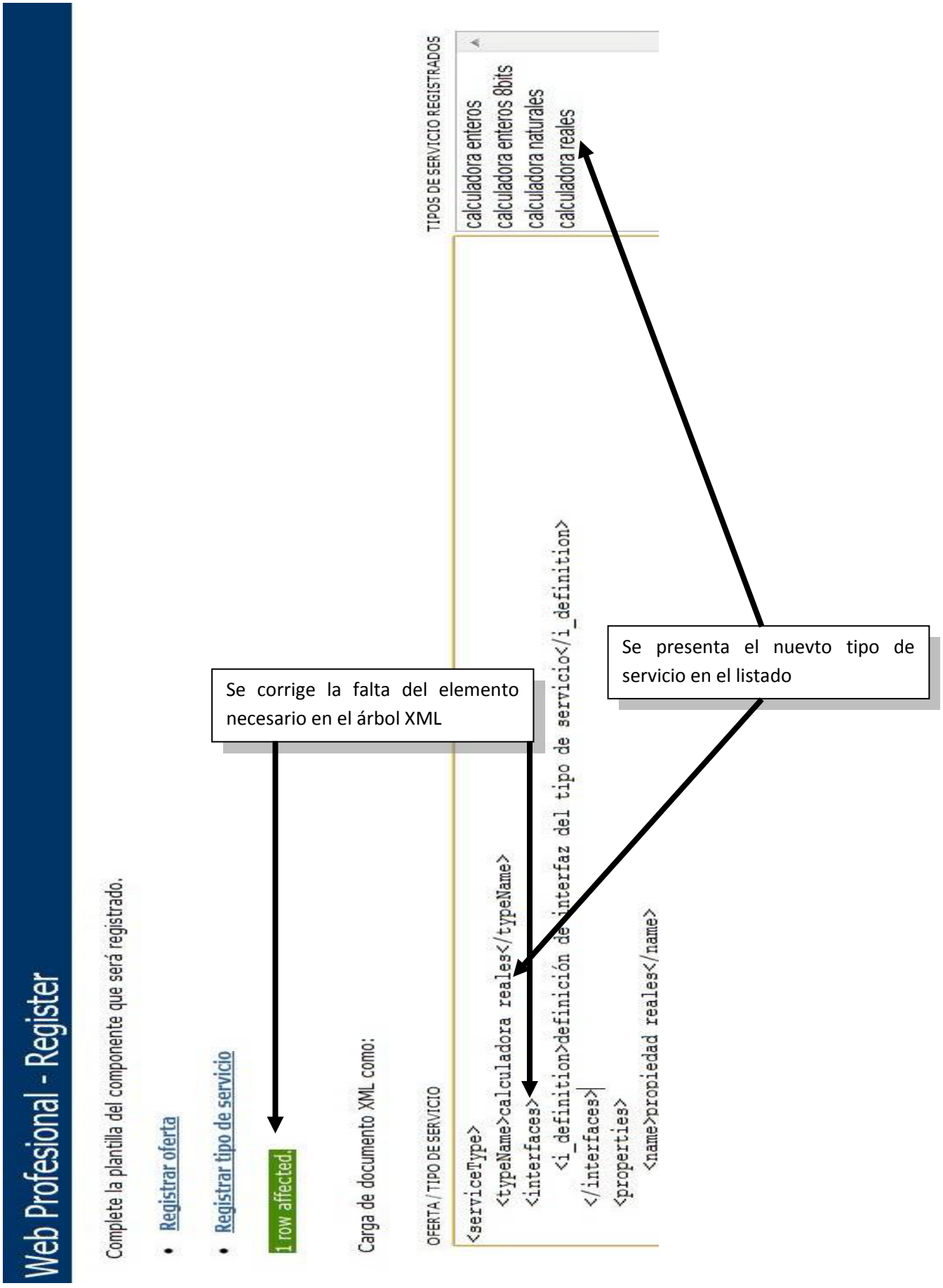
Interfaz gráfica del register



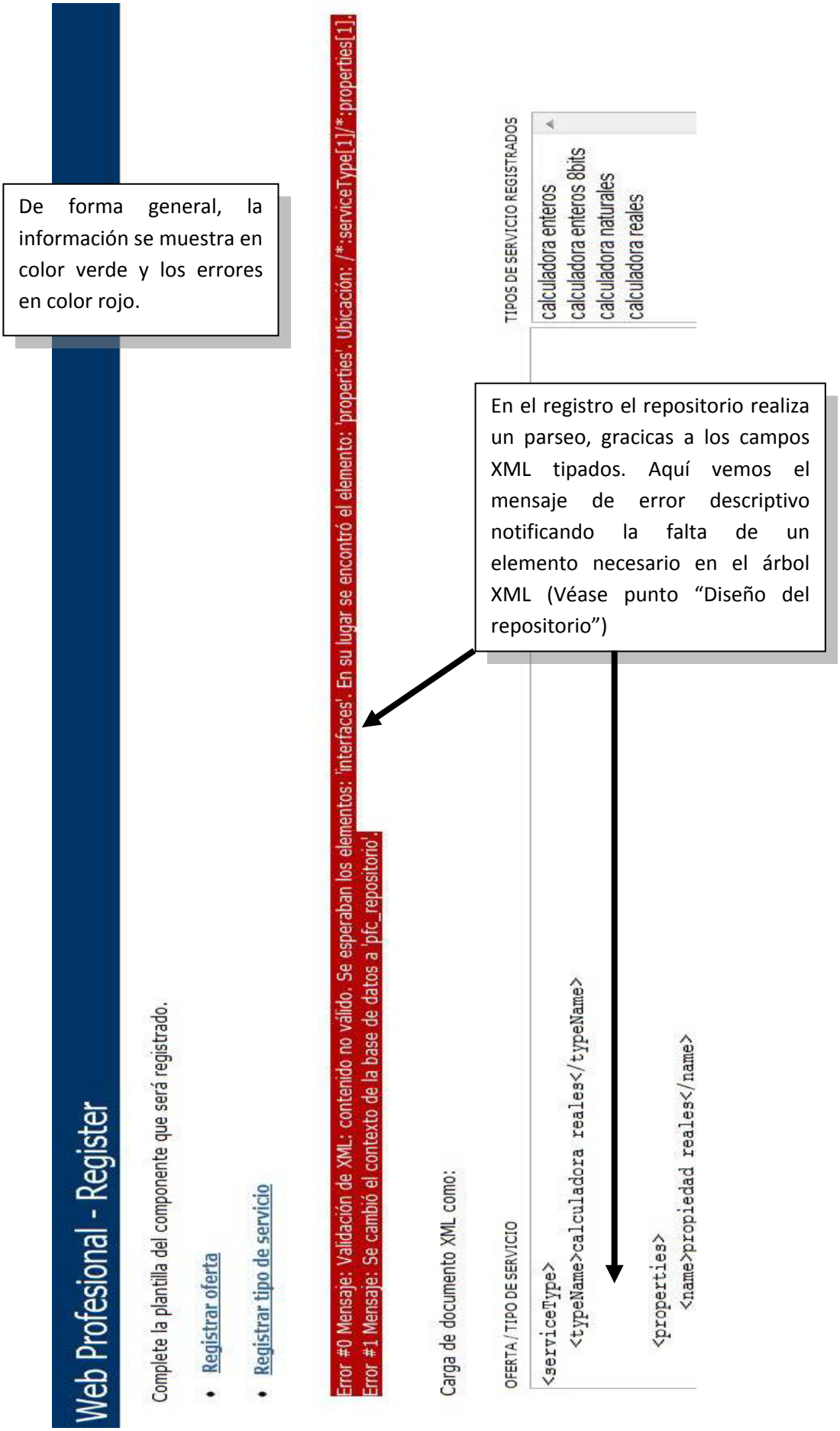
Consultando información de un tipo de servicio en la **interfaz register**



Registro satisfactorio en el Trader de una oferta o tipo de servicio.



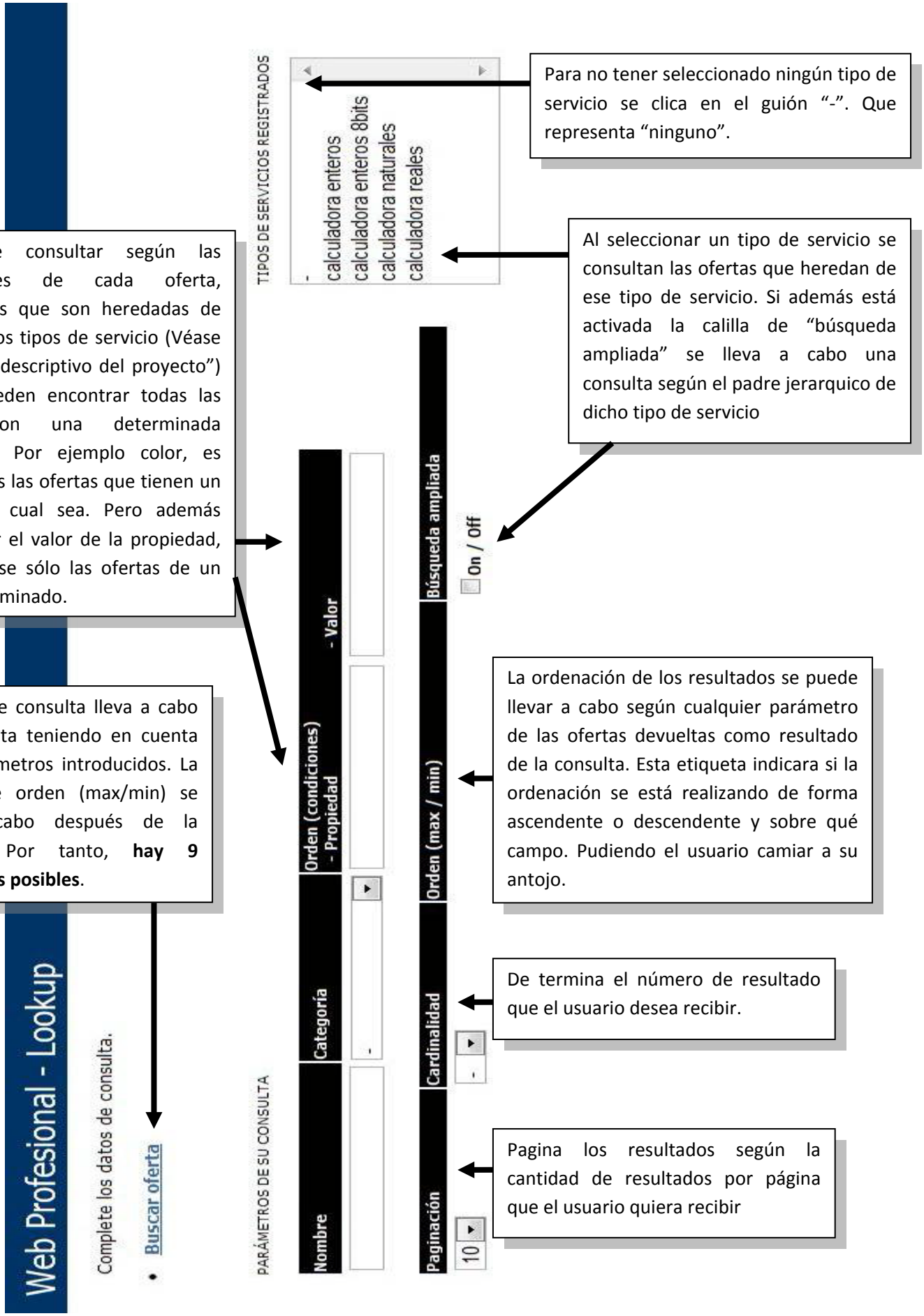
Información devulta al usuario del intento de **registro** de una ofeta o tipo de servicio cuyo **XML** es **erróneo**.



De forma general, la información se muestra en color verde y los errores en color rojo.

En el registro el repositorio realiza un parseo, gracicas a los campos XML tipados. Aquí vemos el mensaje de error descriptivo notificando la falta de un elemento necesario en el árbol XML (Véase punto “Diseño del repositorio”)

Interfaz gráfica del Lookup



Para no tener seleccionado ningún tipo de servicio se clica en el guión "-". Que representa "ninguno".

Al seleccionar un tipo de servicio se consultan las ofertas que heredan de ese tipo de servicio. Si además está activada la calilla de "búsqueda ampliada" se lleva a cabo una consulta según el padre jerarquico de dicho tipo de servicio

Se puede consultar según las propiedades de cada oferta, recordemos que son heredadas de uno o varios tipos de servicio (Véase "Resumen descriptivo del proyecto") Así se pueden encontrar todas las ofetas con una determinada propiedad. Por ejemplo color, es decir, todas las ofertas que tienen un color, sea cual sea. Pero además determinar el valor de la propiedad, mostrándose sólo las ofertas de un color determinado.

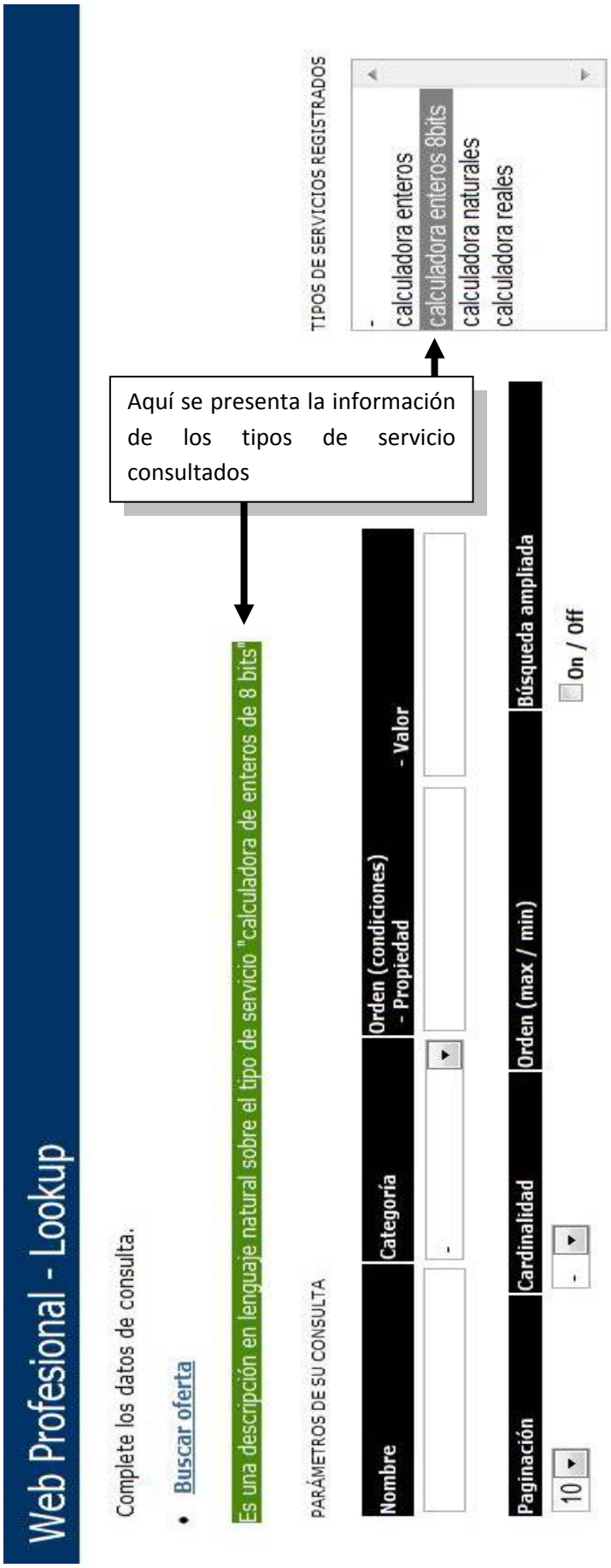
El botón de consulta lleva a cabo una consulta teniendo en cuenta los 8 parámetros introducidos. La política de orden (max/min) se lleva a cabo después de la consulta. Por tanto, **hay 9 parámetros posibles**.

La ordenación de los resultados se puede llevar a cabo según cualquier parámetro de las ofertas devueltas como resultado de la consulta. Esta etiqueta indicara si la ordenación se está realizando de forma ascendente o descendente y sobre qué campo. Pudiendo el usuario camiar a su antojo.

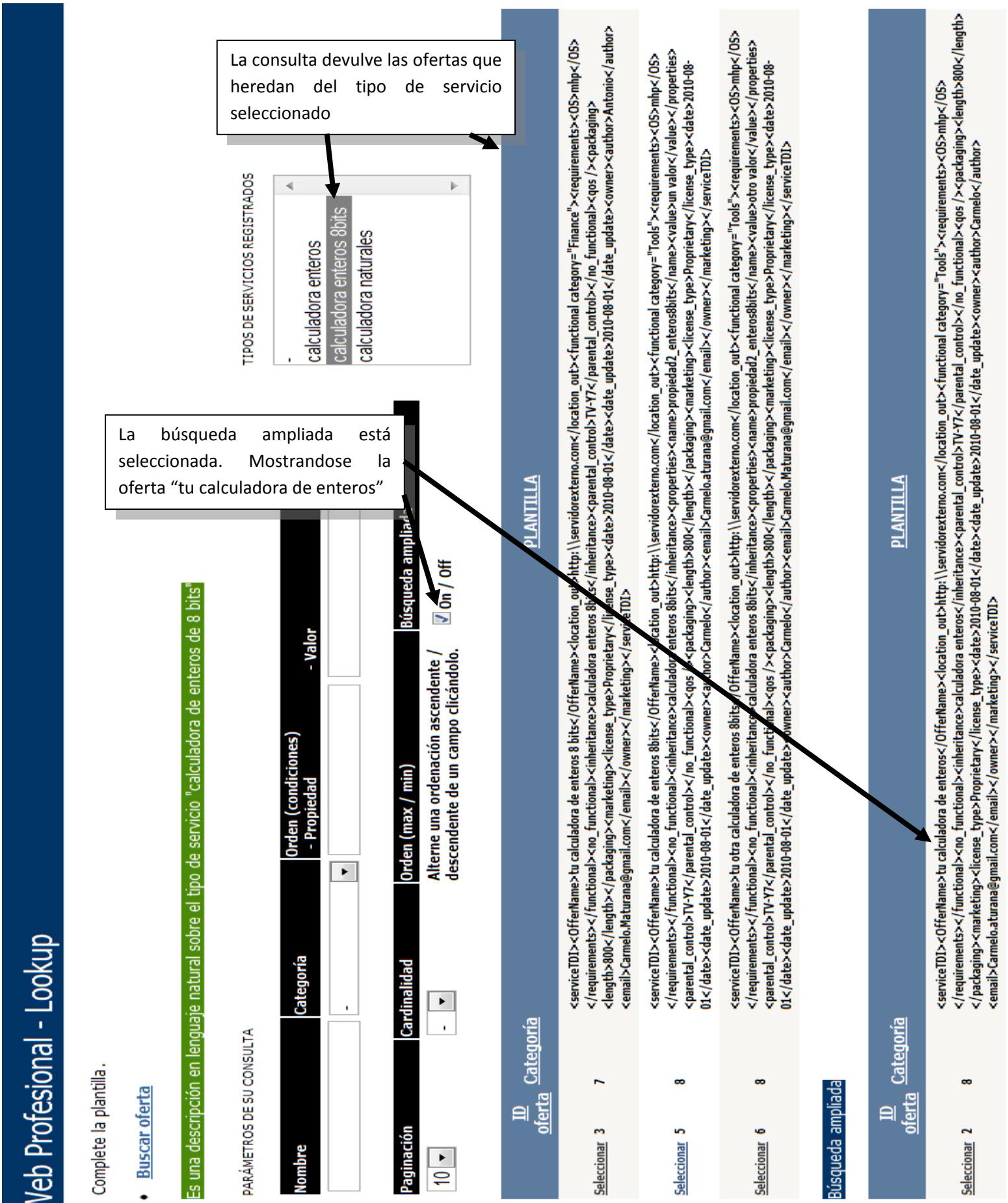
De termina el número de resultado que el usuario desea recibir.

Pagina los resultados según la cantidad de resultados por página que el usuario quiera recibir

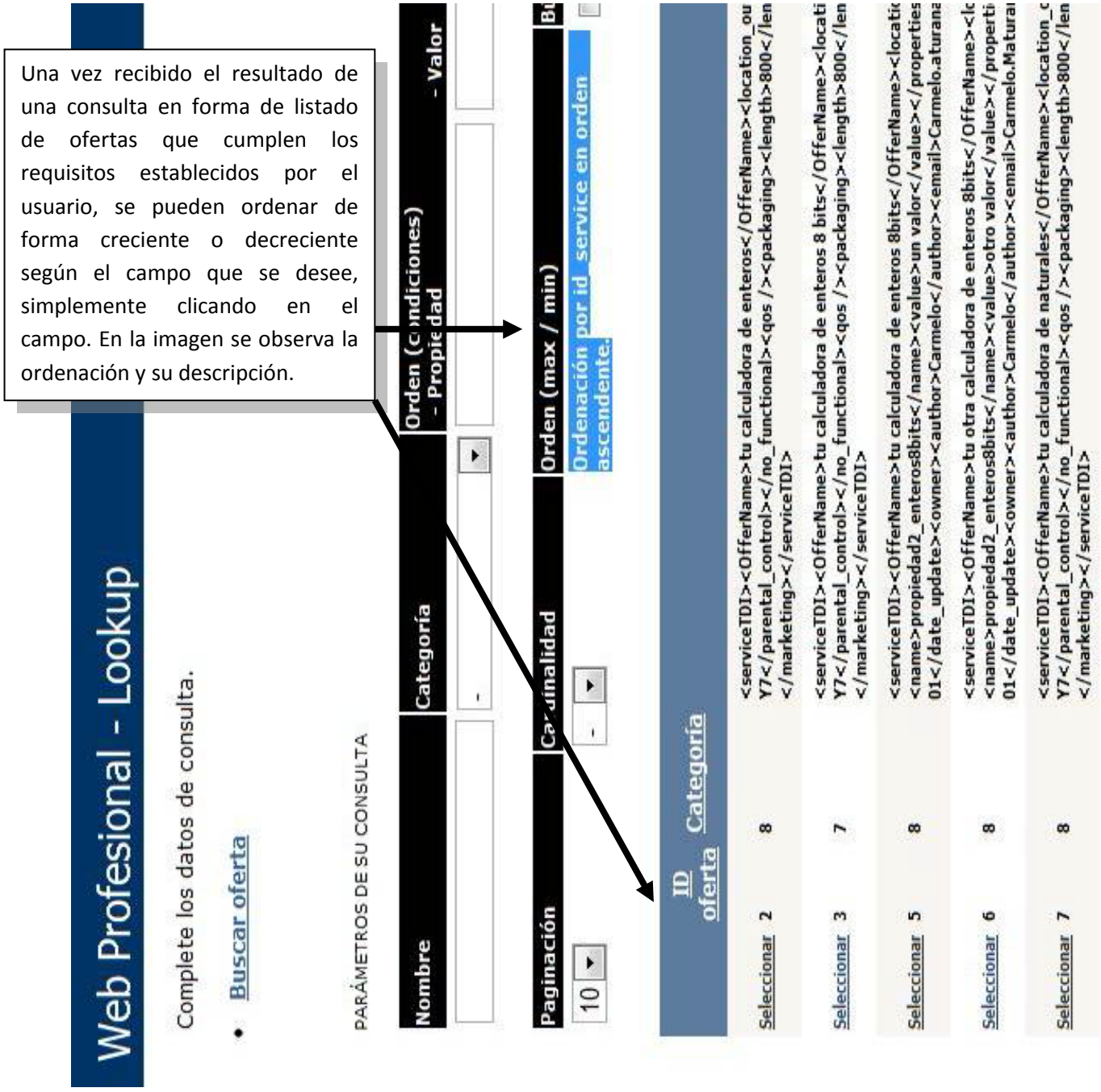
Información básica para conocer **el tipo de servicio a consultar**



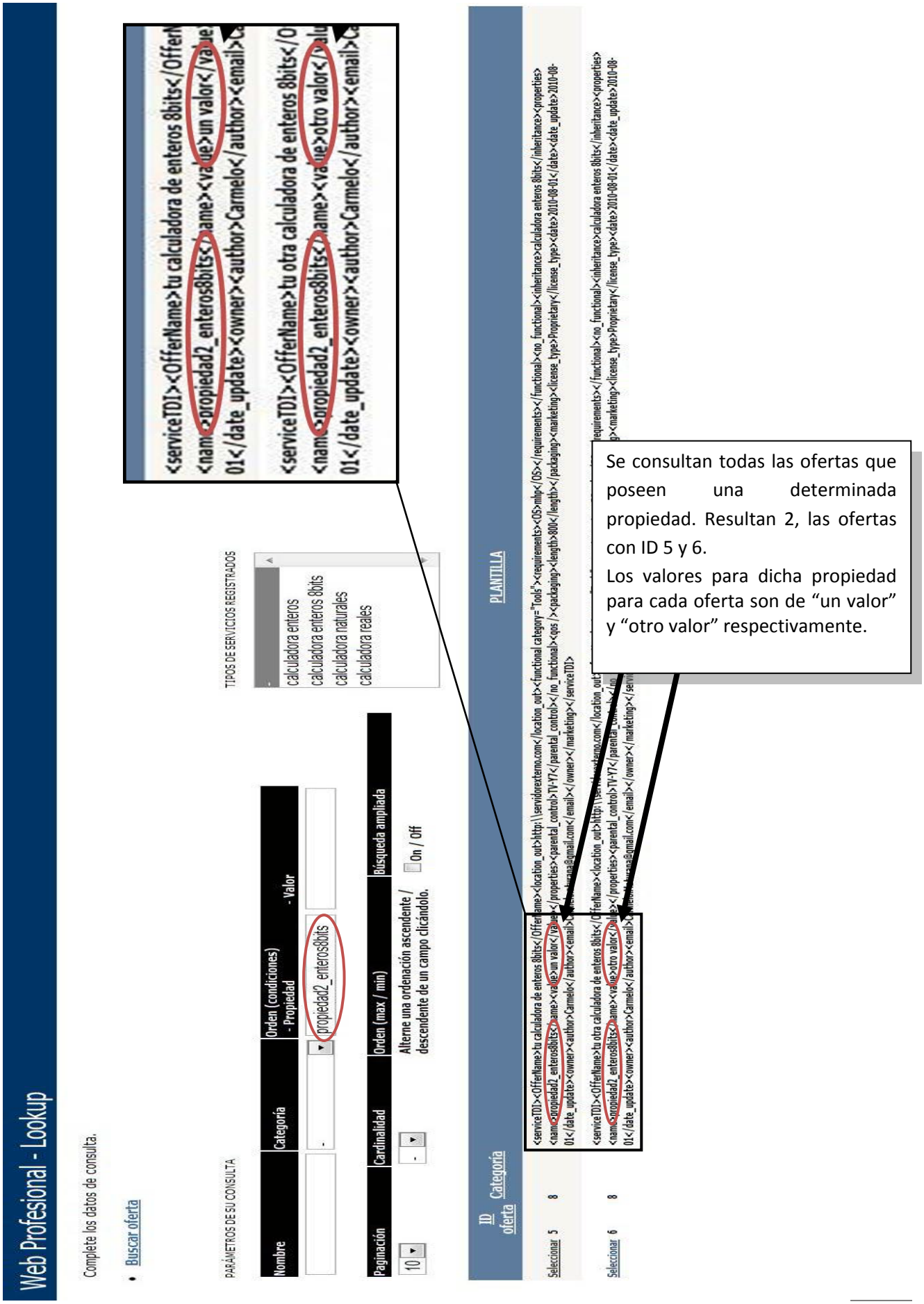
Interfaz gráfica del lookup haciendo uso de la búsqueda ampliada.



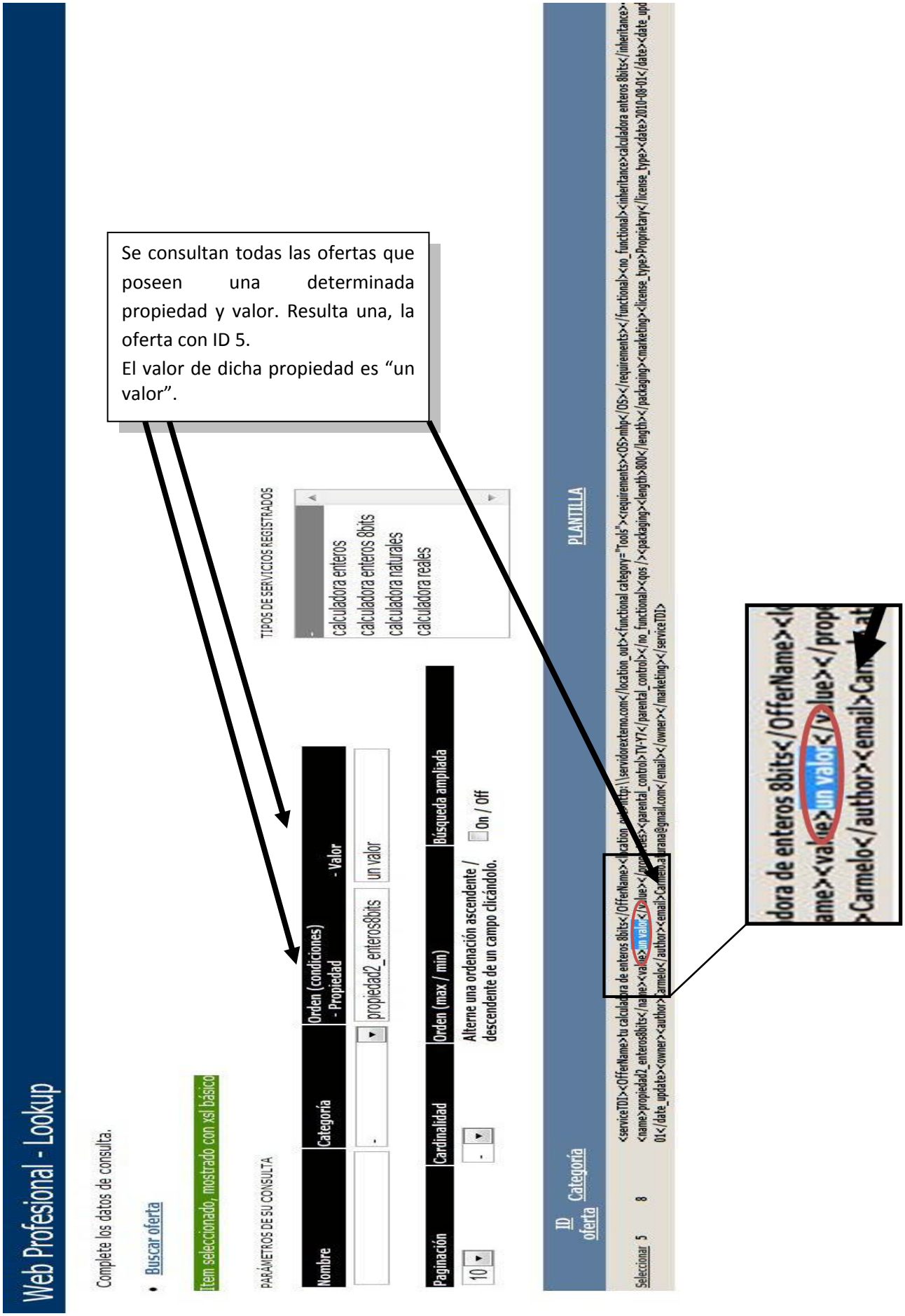
Consulta post ordenada en el lookup



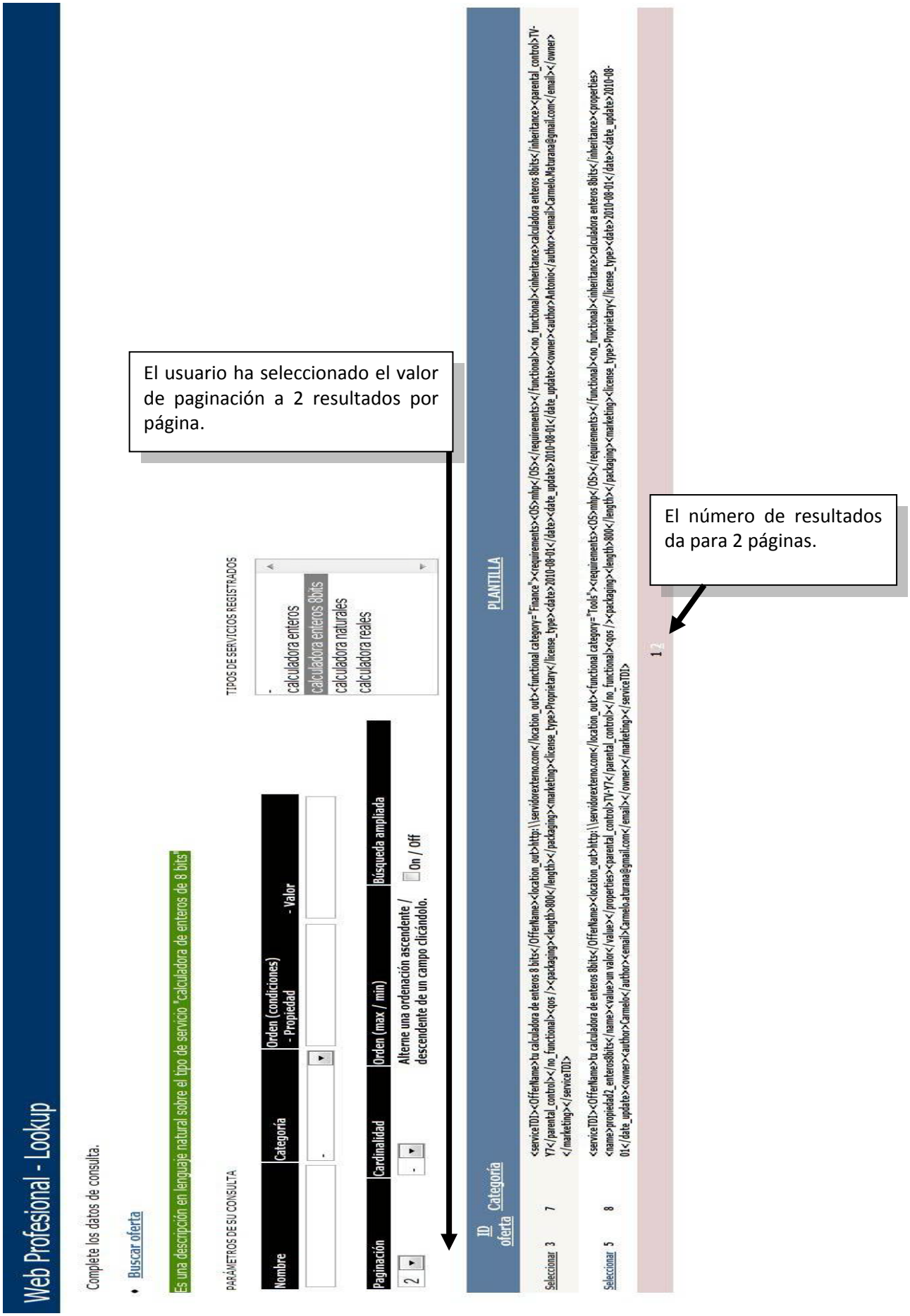
Consulta de ofertas por nombre de parámetros



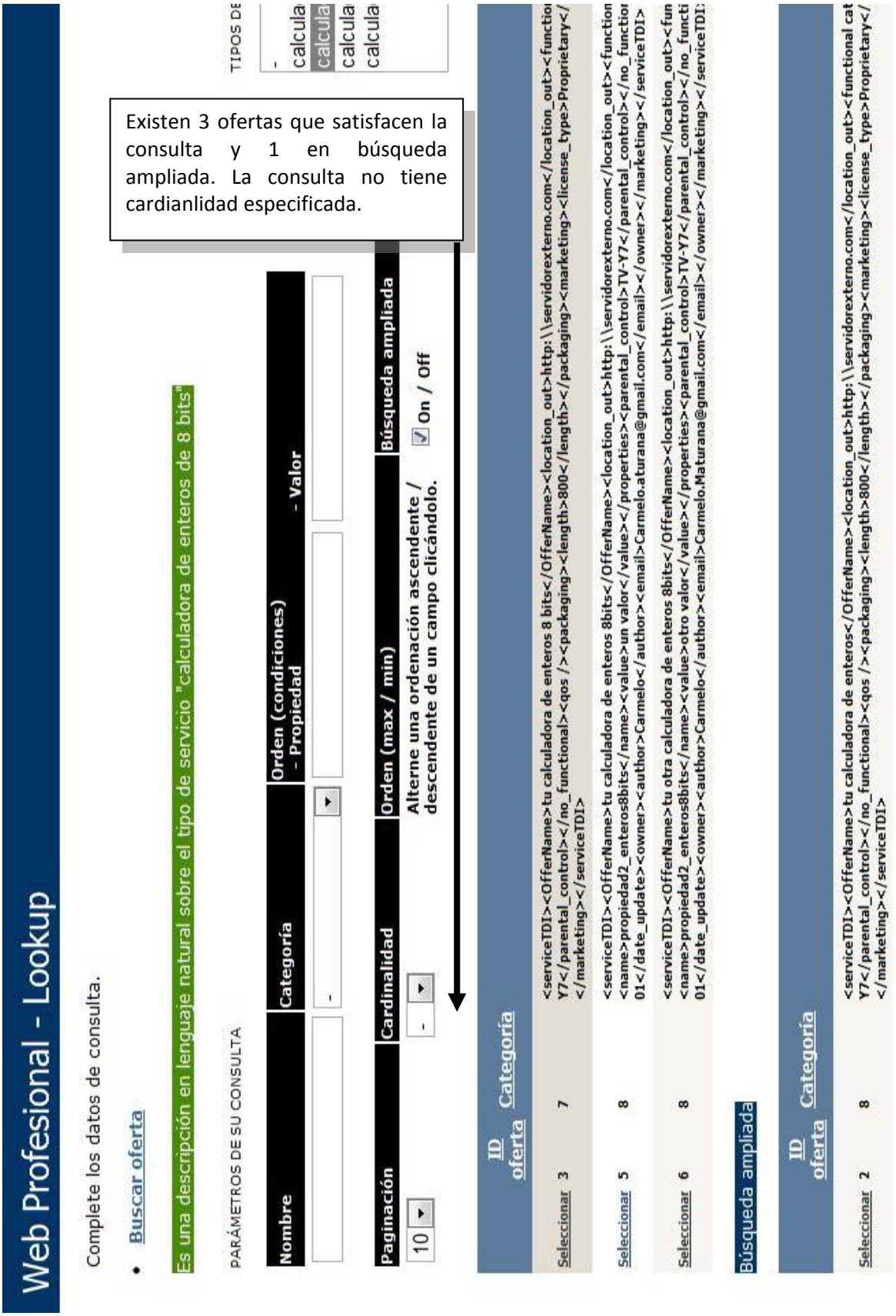
Consulta de ofertas por nombre de parámetros **y valor**



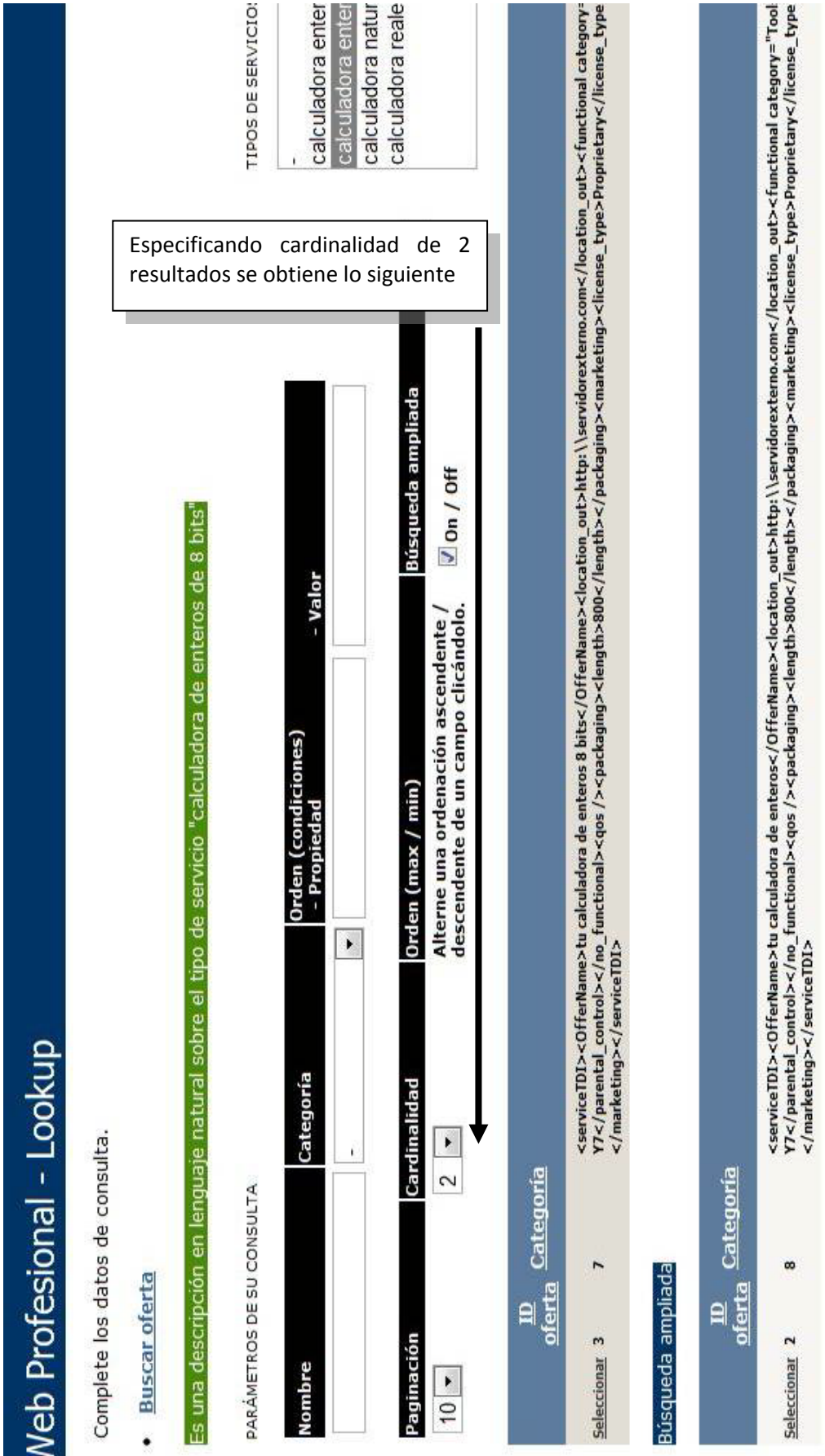
Paginación de resultados en una consulta



Sin cardinalidad en la interfaz lookup

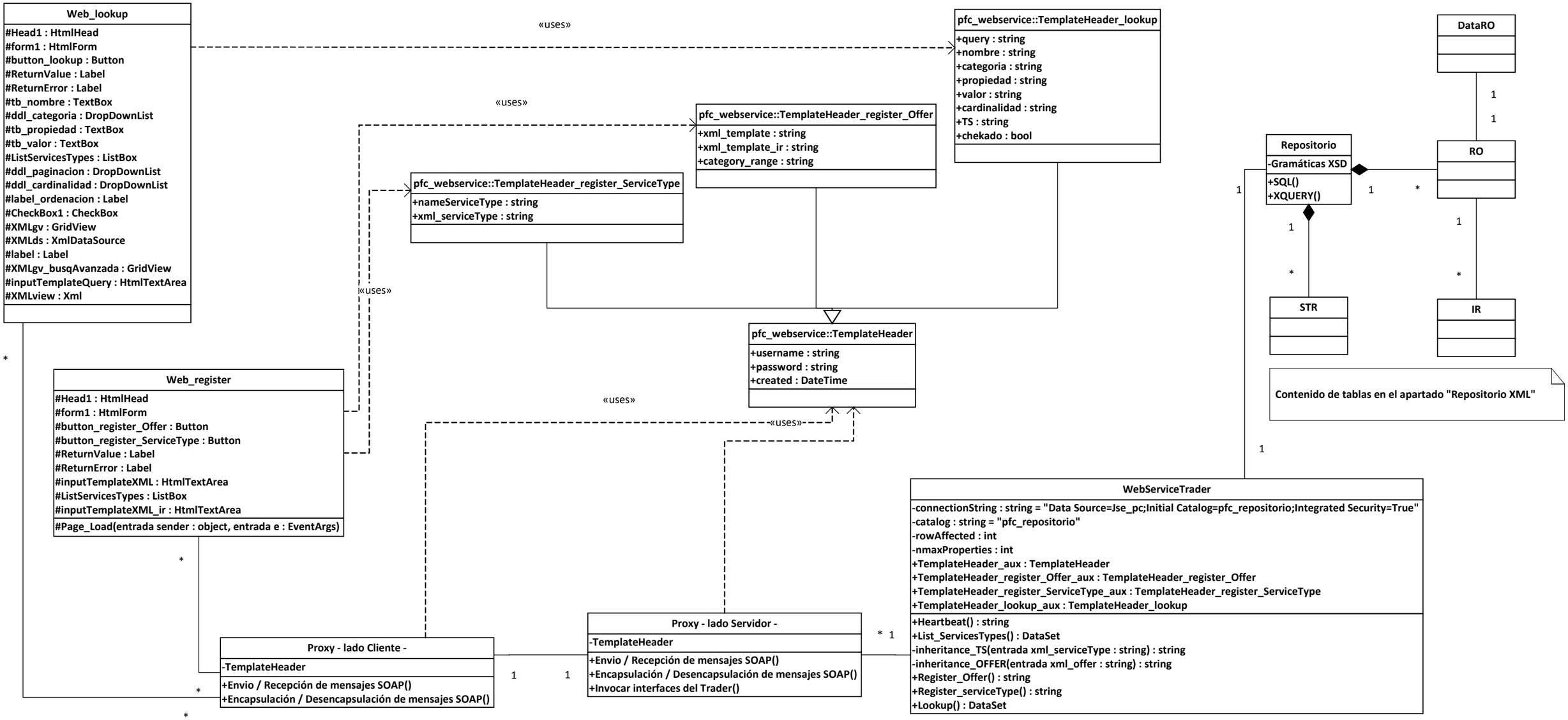


Con cardinalidad en la interfaz lookup

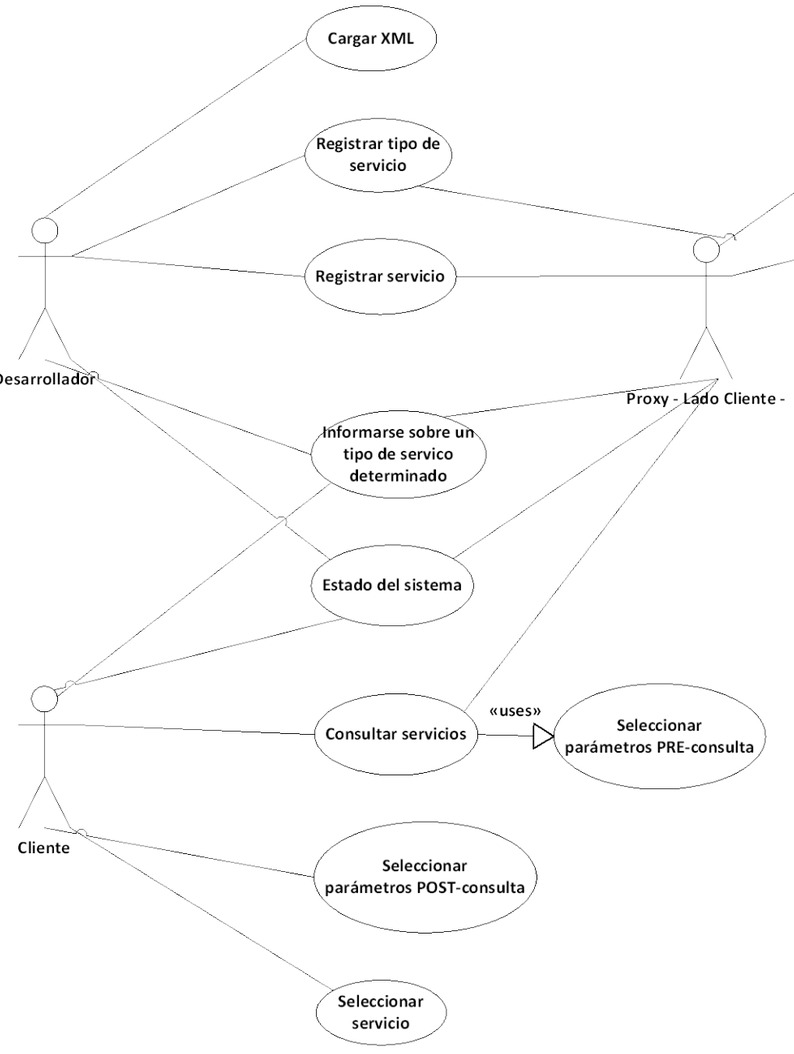


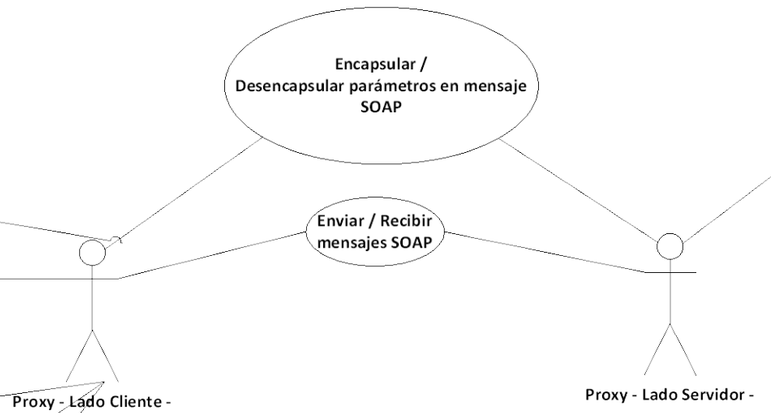
Se presentan los siguientes diagramas de clases, casos de uso y secuencia a fin de esclarecer los detalles del sistema. En el siguiente capítulo se presenta la implementación.

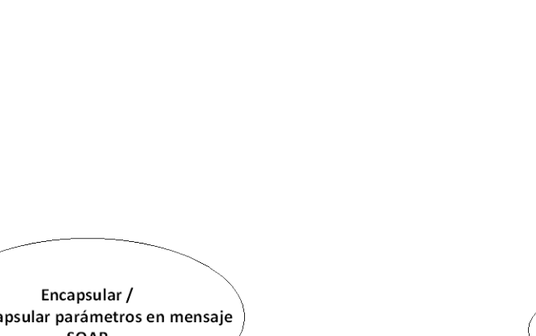
20. Diagrama de clases

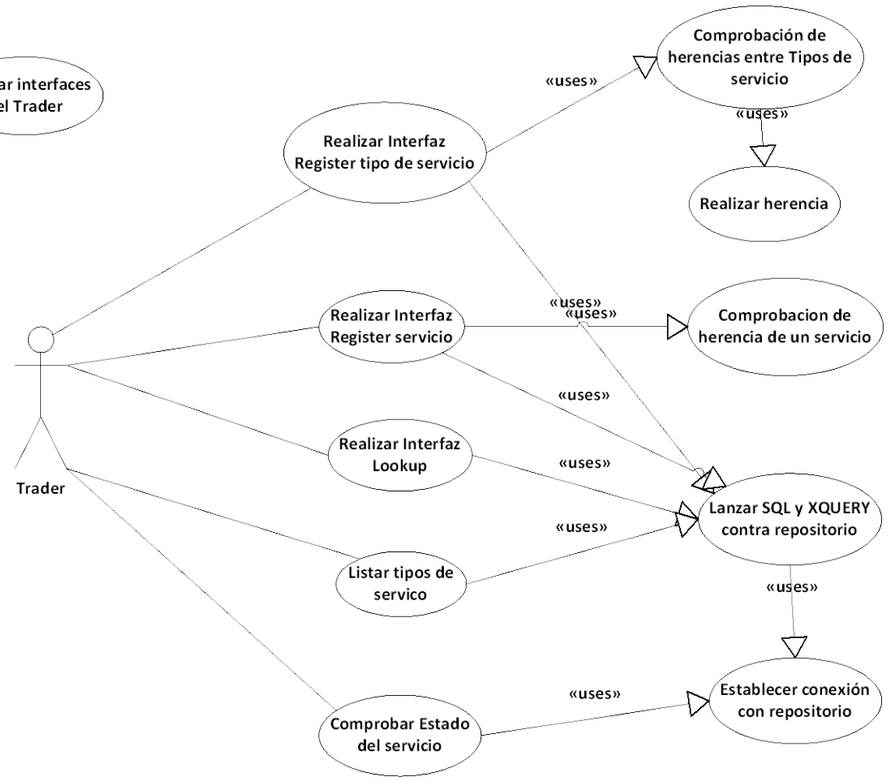


21. Diagrama de casos de uso









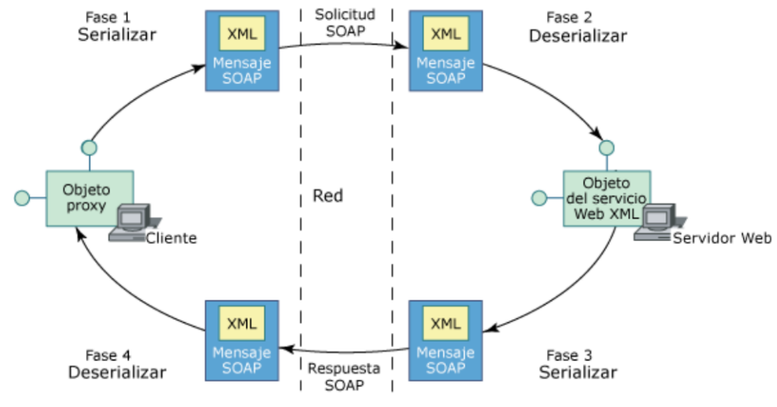
El proyecto se centra en el desarrollo del Trader.
Dejando en segundo plano la típica gestión de usuarios.

- Sólo hay un caso de uso "Interfaz Lookup" PORQUE por definición la interfaz lookup de un Trader realiza búsquedas de servicios, no de tipo de servicios.
- Sólo se realiza la herencia entre tipos de servicio PORQUE la herencia de un servicio depende de la selección del usuario.
Por ello, sólo se comprueba que sea consistente, pero no se puede llevar a cabo de una forma automatizada.

"Parámetros PRE-CONSULTA" añade criterios de búsqueda que serán enviados al TRADER, quien realiza las consultas.
Dichos criterios son: buscar por nombre, categoría, valor de propiedades, cardinalidad y búsqueda ampliada
* La cardinalidad consiste en dejar de buscar servicios cuando durante la búsqueda ya se dispone de un determinado nº de resultados.
* La búsqueda ampliada consiste en realizar la consulta relajando los criterios de búsqueda especificados en caso de que no se obtenga ningún resultado con ellos.

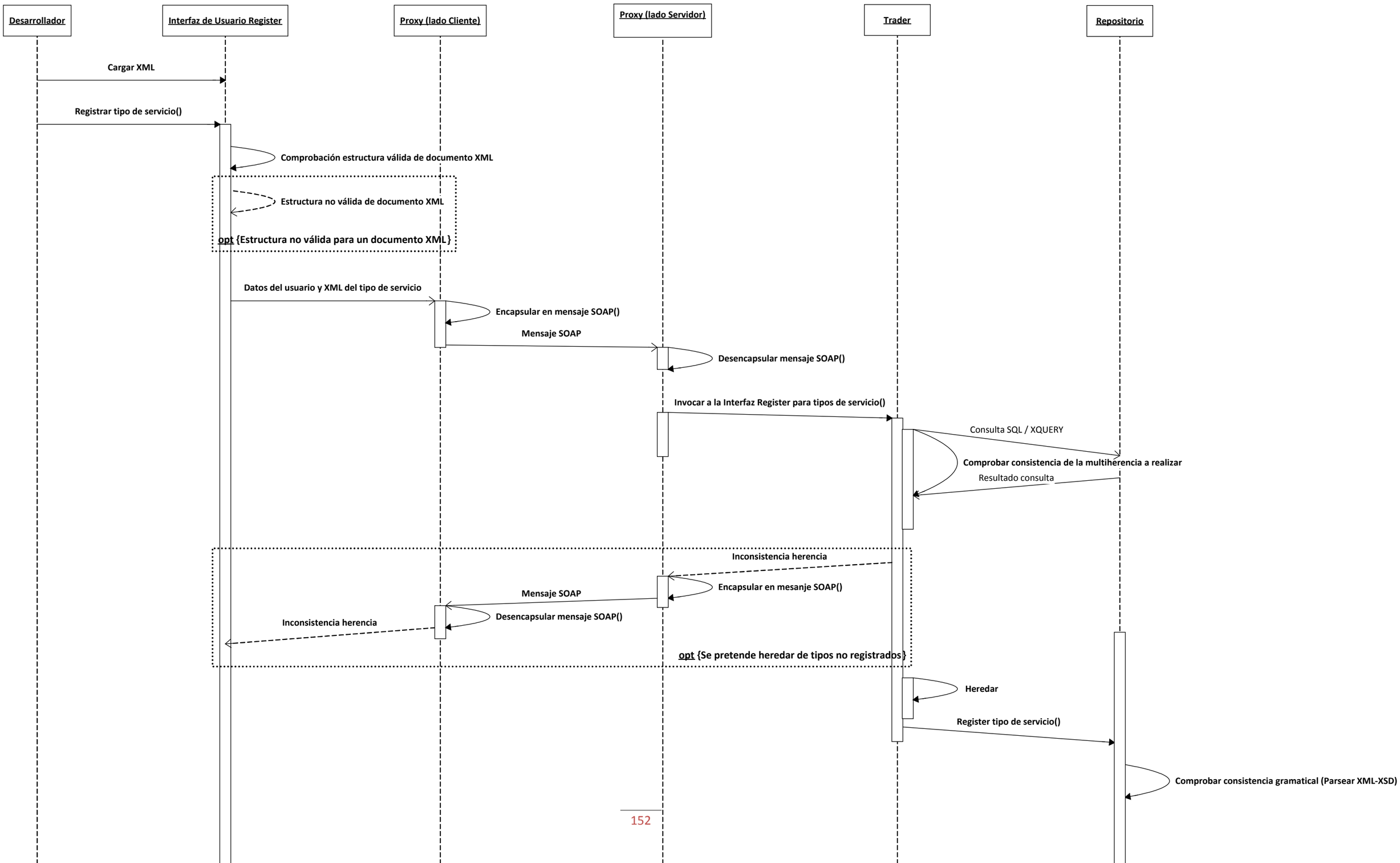
"Parámetros POST-CONSULTA" adapta la visualización de los resultados obtenidos de la consulta ya residentes en el cliente. Este caso de uso no interactua con el Trader.
* Ordenación de los resultados según la característica que seleccione el cliente.

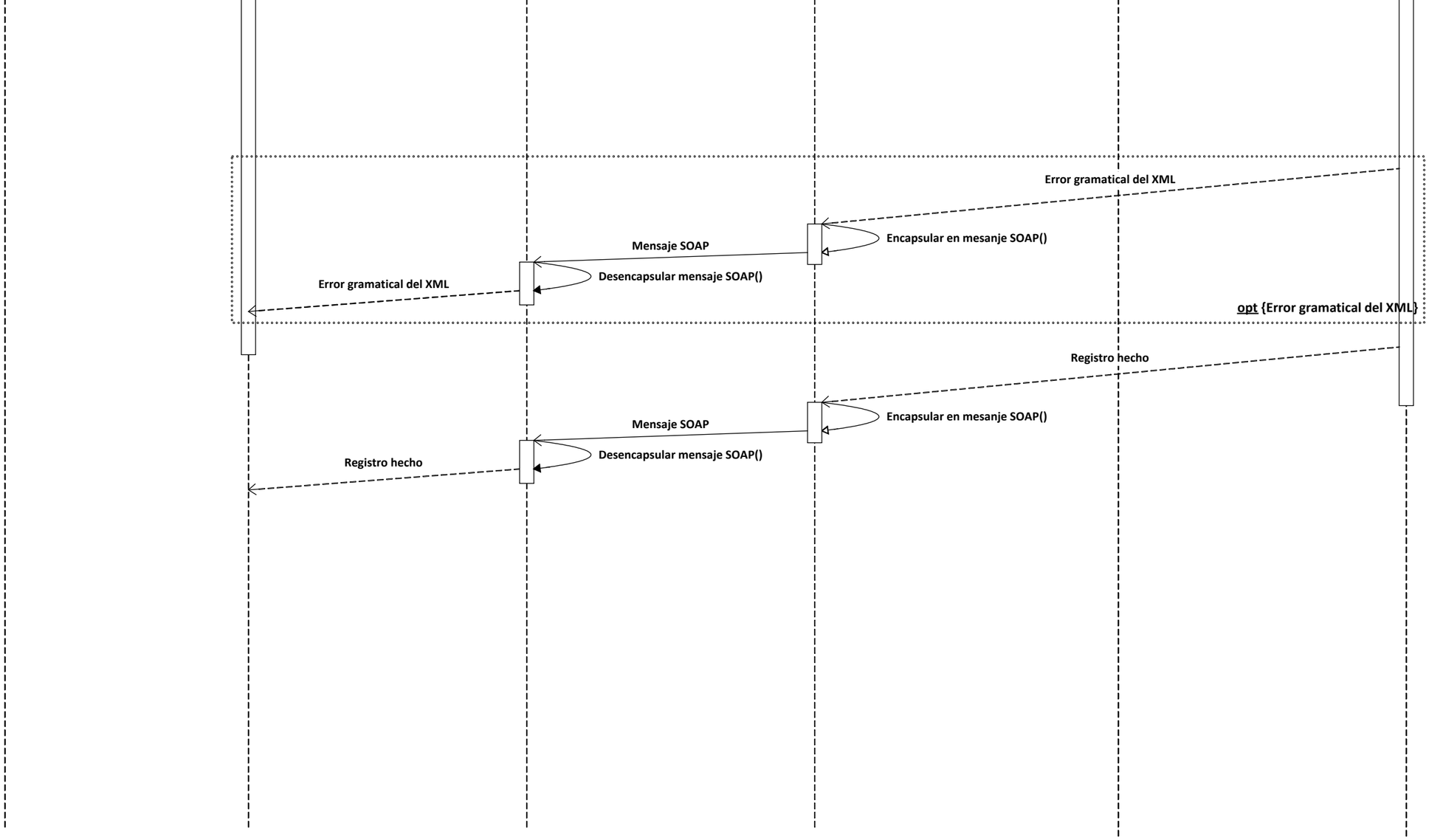
-Notas y aclaraciones



22. Diagrama de secuencia

Se simplifican los pasos de conexión, envio de SQL, etc. El resto de operaciones es de mecánica similar.




Error gramatical del XML
Encapsular en mesanje SOAP()
Mensaje SOAP
Desencapsular mensaje SOAP()
Error gramatical del XML
opt {Error gramatical del XML}
Registro hecho
Encapsular en mesanje SOAP()
Mensaje SOAP
Desencapsular mensaje SOAP()
Registro hecho

23. Código: Objeto cliente – Register

Librerías necesarias para la implementación.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Web_register.aspx.cs"
Inherits="pfc_webservice.Web_register" ValidateRequest="false" %>

<%@ Import Namespace="System.Configuration"%>
<%@ Import Namespace="System.Data"%>

<%@ Import Namespace="System.Xml.Schema"%>
<%@ Import Namespace="System.Xml"%>
<%@ Import Namespace="System.Xml.Linq"%>
<%@ Import Namespace="System.Linq"%>
```

Cominezo del documento HTML. Éste código será embebido con ASP.NET y C#.
Se define las principales variables del objeto cliente, siendo estas: Una colección de esquemas, un objeto DataSet para contenido homogéneo de datos y una instancia ArrayList para los distintos listados que se llevarán a cabo en la capa de presentación al usuario.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

   <script runat="server" language="c#">

       XmlSchemaSet schemas = new XmlSchemaSet();
       DataSet dts = new DataSet();
       ArrayList listadoNombresTS = new ArrayList();
```

Colección de esquemas accesibles desde el lado cliente para futuras comprobaciones de tipado XML.

```
       void coleccionXSD(Object o, EventArgs e)
       {
           schemas.Add("",
"D:/Proyecto/libro/pfc_webservice/pfc_webservice/pfc_webservice/App_Data/RO.xsd")
;
           schemas.Add("IR",
"D:/Proyecto/libro/pfc_webservice/pfc_webservice/pfc_webservice/App_Data/IR.xsd")
;
           schemas.Add("STR",
"D:/Proyecto/libro/pfc_webservice/pfc_webservice/pfc_webservice/App_Data/STR.xsd"
);
```

```
            schemas.Add("comments_ro",
"D:/Proyecto/libro/pfc_webservice/pfc_webservice/pfc_webservice/App_Data/comments
_ro.xsd");
        }
```

Éste método lleva a cabo un listado de los tipos de servicio registrados en el Trader para que el usuario pueda conocerlos, consultarlos y así elegir de entre ellos los tipos de servicio entre los que le interesa heredar para el presente registro. Ya sea de otro tipo de servicio o de una oferta.
La creación del mensaje SOAP en el objeto cliente consiste en la instancia de la clase de una de las cabeceras SOAP que hayamos definido en el servidor, y dar valores a la cabecera y al cuerpo del mensaje.
Su principal característica es el uso exclusivo de documentos XML. Para ello se ha heredado de la clase proxy [32].

```
        void ListST(Object o, EventArgs e)
        {
```

Crear una nueva instancia de la clase proxy desde las librerías cargadas en el servidor. El cliente conoce al servidor y tiene acceso a sus librerías a través del espacio de nombres de éste.

```
            TemplateHeader TemplateHeader_soap = new TemplateHeader();

             // Valores del TemplateHeader_soap.
             TemplateHeader_soap.username =
"usernameCARMELO";//User.Identity.Name;//Jse prueba con una cadena
             TemplateHeader_soap.password =
"passwordCARMELO";//User.Identity.IsAuthenticated;
             TemplateHeader_soap.created = DateTime.UtcNow;
             // Create a new instance oqf the proxy class.
             pfc_webservice.WebServiceTrader proxy = new
pfc_webservice.WebServiceTrader();
```

Se añade al proxy la cabecera

```
             // Add the MyHeader SOAP header to the SOAP request.
             proxy.TemplateHeader_aux = TemplateHeader_soap;
```

Se llama al método del servicio web a través de la clase proxy instanciada. Aquí se está devolviendo el mensaje SOAP des-encapsulado.

```
             // Call the method on the proxy class that communicates with your Web
service method.
              dts = proxy.List_ServicesTypes();
```

Las clausulas a prueba de fallos y control de excepciones son una constante en todo el código.

```
            try
            {
                listadoNombresTS.Clear();
                ListServicesTypes.Items.Clear();
                foreach (DataRow dr in dts.Tables[0].Rows)
                {
                    listadoNombresTS.Add(dr["nameServiceType"].ToString());
                }
                ListServicesTypes.DataSource = listadoNombresTS;

                try
                {
                    ListServicesTypes.DataBind();
                    ListServicesTypes.Enabled = true;
                }
```

Por emplo, aquí se envía un mensaje a la capa de presentación (HTML) sobre un evento del cual será informado el usuario finalmente en pantalla, a través de la web.

```
                catch
                {
                    ReturnError.Text = "No se ha podido recuperar la lista de
tipos de servicio. Comuniquelo al administrador.";
                    ListServicesTypes.Enabled = false;
                }
            }
            catch
            {
                ReturnError.Text = "No se ha podido recuperar la lista de tipos
de servicio. Comuniquelo al administrador.";
                ListServicesTypes.Enabled = false;
            }
        }
```

Debido a que el usuario puede usar recurrentemente el objeto cliente, para que no exista necesidad de ejecutar manualmente el HTML, se ha implementado éste procedimiento que actualiza el listado de tipos de servicio registrados en el Trader. Es importante notar que dicho listado consta de 2 capas siendo informativo en la identificación de los tipos de servicio (en la primera capa) y en caso de consulta (con 1 sólo click) informativo en una descripción del propio tipo de servicio. Consiguiendo una identificación inequívoca y útil del mismo. Este manejo de documentos XML se lleva a cabo mediante elementos xDocument [34].

```
        void List_SelectedIndexChanged (Object o, EventArgs e)
        {
            if (ListServicesTypes.SelectedIndex > -1)
            {
                String expresion = "nameServiceType = '" +
ListServicesTypes.SelectedItem.Text + "'";
                DataRow[] dr = dts.Tables[0].Select(expresion);
                String xml = dr[0].ItemArray.GetValue(2).ToString();
```

```
                XDocument xmldoc = XDocument.Parse(xml);
                var result =
                  from c in xmldoc.Elements("serviceType")
                  select c.Element("annotation").Value;

                ReturnValue.Text = result.Single();
                ReturnError.Text = "";
            }
        }
```

El siguiente procedimiento contacta con las interfaces básicas del Trader implementadas en el Trader consistiendo ésta en el registro de ofertas.

```
        void register_Offer(Object o, EventArgs e)
        {
            XDocument xmldoc = new XDocument();
```

En primer lugar se valida la consistencia del documento XML aportado por el usuario.

```
            try
            {
                xmldoc = XDocument.Parse(inputTemplateXML.Value);
            }
            catch (XmlException excepcionXML)
            {
                ReturnError.Text = excepcionXML.Message;
                return;
            }
```

Comprueba la consistencia en la HERENCIA SIMPLE del TS para una oferta. (Código preparado para multiherencia). Se puede llevar a cabo gracias al uso del lenguaje LINK, embebido en ASP.NET, embebido a su vez en el HTML.

No se lleva a cabo la herencia del TS de una oferta por la sencilla razón de que el usuario tiene que darle un valor.

```
            try
            {
                System.Collections.Generic.IEnumerable<XElement> enumeracion_st =
xmldoc.Element("serviceTDI").Elements("no_functional");
                foreach (XElement st in enumeracion_st)
                {
                    var ServiceType =
                        from c in st.Elements("inheritance")
                        select c.Value;
                    if (!listadoNombresTS.Contains(ServiceType.Single()))
                    {
                        ReturnError.Text = "El tipo de servicio: \"" +
ServiceType.Single() + "\" no está registrado.";
                        return;
                    }
                }
```

```
            }
            catch {
                ReturnError.Text = "Toda oferta debe heredar de un tipo de
servicio mediante el elemento 'inheritance' en las propiedades no funcionales.";
                return;
            }
```

Evita que el cliente tenga que introducir información por duplicado, obteniendo la descripción del tipo de servicio.

```
            var result =
                from c in xmldoc.Elements("serviceTDI").Elements("functional")
                select c.Attribute("category").Value;
```

Aquí se pasa del nombre de categoria a su nº de enumeración (véase el diseño del repositorio). No se ha implementado con un switch porque no adminte trabajar con string, sólo enteros.

```
            string nCategoria = "";
            if      (result.Single() == "Shop")                 nCategoria = "1";
            else if (result.Single() == "Communication")        nCategoria = "2";
            else if (result.Single() == "Sports")               nCategoria = "3";
            else if (result.Single() == "People")               nCategoria = "4";
            else if (result.Single() == "Health")               nCategoria = "5";
            else if (result.Single() == "Lifestyle")            nCategoria = "6";
            else if (result.Single() == "Finance")              nCategoria = "7";
            else if (result.Single() == "Tools")                nCategoria = "8";
            else if (result.Single() == "Multimedia")           nCategoria = "9";
            else if (result.Single() == "News and meteorology") nCategoria =
"10";
            else if (result.Single() == "Learning")             nCategoria =
"11";
            else if (result.Single() == "Productivity")         nCategoria =
"12";
            else if (result.Single() == "Free time")            nCategoria =
"13";
            else if (result.Single() == "Reference")            nCategoria =
"14";
            else if (result.Single() == "Trips")                nCategoria =
"15";
            else if (result.Single() == "Themes")               nCategoria =
"16";
            else if (result.Single() == "Others")               nCategoria =
"17";


            TemplateHeader_register_Offer TemplateHeader_soap = new
TemplateHeader_register_Offer();
```

Se ha incluido código útil para la ampliación del sistema gestor (no del Trader o servicio Web, que es el objetivo del presente proyecto) para que posteriormente se pueda llevar a

cabo una comprobación de la autentificación del usuario que está solicitando hacer uso de las interfaces del Trader. Incluso datos cronológicos para el control por parte de un log.

```
            TemplateHeader_soap.username =
"usernameCARMELO";//User.Identity.Name;//Jse prueba con una cadena
            TemplateHeader_soap.password =
"passwordCARMELO";//User.Identity.IsAuthenticated;
            TemplateHeader_soap.created = DateTime.UtcNow;
```

Se continúa recopilando la información necesaria para conformar el mensaje SOAP que será enviado al Trader para el registro de la oferta o tipo de servicio. La información principal es sencillamente esa la propia oferta o tipo de servicio.

```
            TemplateHeader_soap.xml_template = inputTemplateXML.Value;
            TemplateHeader_soap.xml_template_ir = inputTemplateXML_ir.Value;
            TemplateHeader_soap.category_range = nCategoria;//Resultado linq
            // Create a new instance oqf the proxy class.
            pfc_webservice.WebServiceTrader proxy = new
pfc_webservice.WebServiceTrader();
            // Add the MyHeader SOAP header to the SOAP request.
            proxy.TemplateHeader_register_Offer_aux = TemplateHeader_soap;
            // Call the method on the proxy class that communicates with your Web
service method.
            string results = proxy.Register_Offer();
            // Display the results of the method in a label.
            ReturnValue.Text = results;
            ReturnError.Text = "";
        }
```

Aquí se presenta el código implementado para el registro, en este caso, de un tipo de servicio. Como puede leer en el primer punto de este documento “Resumen descriptivo del proyecto” los tipos de servicio son vitales para el funcionamiento de un Trader.

```
        void register_ServiceType(Object o, EventArgs e)
        {
            XDocument xmldoc = new XDocument();

            try
            {
                xmldoc = XDocument.Parse(inputTemplateXML.Value);
            }
            catch(XmlException excepcionXML)
            {
                ReturnError.Text = excepcionXML.Message;
                return;
            }
```

Comprueba la consistencia en la MULTIHERENCIA de tipos de sercivio para un nuevo tipo de servicio. Es decir, se comprueba que todo tipo de servicio herede de tipos de serivio existentes, en caso de que lo haga. Además se comprueba que al establecerse una jerarquía entre el presente tipo de servicio y otro, este último realmente exista en el catálogo del Trader.

```
            System.Collections.Generic.IEnumerable<XElement> enumeracion_st =
xmldoc.Element("serviceType").Elements("inheritance");
            foreach (XElement st in enumeracion_st)
            {
                var ServiceType =
                    from c in st.Elements("name")
                    select c.Value;
                if (!listadoNombresTS.Contains(ServiceType.Single()))
                {
                    ReturnError.Text = "No se puede heredar del tipo de servicio
\"" + ServiceType.Single() + "\" porque no se encuentra registrado.";
                    return;
                }
            }
            XElement a = xmldoc.Element("serviceType");
            XElement existe = a.Element("top_hierarchy");
            if (existe != null)
            {
                var ServiceType_padre =
                    from c in xmldoc.Elements("serviceType")
                    select c.Element("top_hierarchy").Value;
                if (!listadoNombresTS.Contains(ServiceType_padre.Single()))
                {
                    ReturnError.Text = "El tipo de servicio 'padre' indicado no
está registrado. Indique uno válido.";
                    return;
                }
            }
        var result =
            from c in xmldoc.Elements("serviceType")
            select c.Element("typeName").Value;

        TemplateHeader_register_ServiceType TemplateHeader_soap = new
TemplateHeader_register_ServiceType();

        TemplateHeader_soap.username = "usernameCARMELO";//User.Identity.Name;
        TemplateHeader_soap.password =
"passwordCARMELO";//User.Identity.IsAuthenticated;
        TemplateHeader_soap.created = DateTime.UtcNow;
        TemplateHeader_soap.xml_serviceType = inputTemplateXML.Value;
        TemplateHeader_soap.nameServiceType = result.Single();
        // Create a new instance oqf the proxy class.
        pfc_webservice.WebServiceTrader proxy = new
pfc_webservice.WebServiceTrader();
        proxy.TemplateHeader_register_ServiceType_aux = TemplateHeader_soap;
        string results = proxy.Register_serviceType();
```

Se lleva a cabo la comprobación de la existencia de resultados en una consulta.

```
        if (results.Contains("Error #0"))
        {
            ReturnError.Text = results;
            ReturnValue.Text = "";
        }
        else
        {
            ReturnValue.Text = results;
```

```
            ReturnError.Text = "";
        }
        this.ListST(o, e);
       }

    </script>
```

Habiendo implementado los registros y consultas necesarias para que el usuario disponga de la información necesaria para registrar tanto ofertas como tipos de servicio desde la misma web, ahora se pasa a presentar el código perteneciente a la capa de presentación dada por HTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
```

Se ha creado una hoja de estilos independiente de dicho código para agilizar su lectura y organizalo todo convenientemente.

```
    <link rel="stylesheet" href="StyleSheet1.css"/>

    <title>Web Service Trader</title>

</head>
```

Aquí comienza el cuerpo del código HTML. Recuerde que las líneas que no caben en un solo renglón debido a la limitación de un folio A4 comienzan sin tabulación en el siguiente.

```
<body>

    <div id="div1">
      <p class="heading1">Web Profesional - Register</p><br/>

          <p class="intro">Complete la plantilla del componente que será
registrado.
    </div>

    <div id="content">
                    <form id="form1" runat="server" oninit="ListST"
onload="coleccionXSD">
                        <li>
                        <asp:Button ID="button_register_Offer" Runat="server"
OnClick="register_Offer"
                                    Text="Registrar oferta" CssClass="josebutton"
Width="133px" />
```

Se recomienda implementar una advertencia al usuario antes de proceder al registro, después de toma de datos. Ésta sería de la siguiente forma:

```
OnClientClick="return confirm('¿Confirma el registro de la oferta?');"
```

```
                        </li>

                        <li>
                        <asp:Button ID="button_register_ServiceType"
Runat="server"
                                    OnClick="register_ServiceType"
                                    Text="Registrar tipo de servicio"
CssClass="josebutton" Width="195px" />
```

```
OnClientClick="return confirm('¿Confirma el registro del tipo de servicio?');"
```

```
                            <br />
                        </li>
```

Los mensajes de información hacia el usuario, en la capa de gestión del servicio web se recogen a continuación:

```
                            <br />
                            <asp:Label id="ReturnValue" Runat="server"
CssClass="information" Text="Seleccione un tipo de servicio para más
información."/>
                            <br />
                            <asp:Label id="ReturnError" Runat="server"
CssClass="informationError"/>
                            <br />
```

La inserción de los datos necesarios para el registro a través de la capa HTML (capa de presentación) se implementa aquí como sigue:

```
                        <br />
                        Carga de documento XML como:
                            <br />
                            <table style="width:1200px; height:auto"
class="intro">
                                <tr>
                                    <td class="style1">
                                        <br />
                                        </td>
                                </tr>
                                <tr>
                                    <td class="style1">
                                        OFERTA
                                        / TIPO DE SERVICIO</td>
                                    <td class="style1">
                                        TIPOS DE SERVICIO REGISTRADOS</td>
                                </tr>
                                <tr>
                                    <td>
                                        <textarea id="inputTemplateXML"
runat="server"
                                        style="height:450px; width:1000px"
cols="20" name="S1" rows="1"></textarea></td>
                                    <td>
```

```
                                        <asp:ListBox ID="ListServicesTypes"
runat="server" Height="100%" Width="200px"

OnSelectedIndexChanged="List_SelectedIndexChanged" AutoPostBack="true"
                                    SelectionMode="Single"></asp:ListBox>
                                    </td>
                                </tr>
                                <tr>
                                    <td>
                                        INTERFAZ DE LA OFERTA</td>
                                    <td>
                                    </td>
                                </tr>
                                <tr>
                                    <td colspan="2">
                                        <textarea id="inputTemplateXML_ir"
runat="server" rows="1"
                                        style="height:100px; width:1000px"
cols="20" name="S2"></textarea></td>
                                </tr>
                            </table>
                            <br />
                            <br />
                            <br />
                    </form>
    </div>
</body>
</html>
```

24. Código: Objeto cliente – Lookup

Librerías necesarias para la implementación del objeto cliente de lookup.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Web_lookup.aspx.cs"
Inherits="pfc_webservice.Web_lookup"%>
<%@ Import Namespace="System.Configuration"%>
<%@ Import Namespace="System.Data"%>

<%@ Import Namespace="System.Xml.Linq"%>
<%@ Import Namespace="System.Linq"%>
```

Comienzo del código ASP.NET

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

   <script runat="server" language="c#">

        DataSet dts = new DataSet();
        DataSet dts2 = new DataSet();
        DataSet dts3 = new DataSet();
        ArrayList values = new ArrayList();
        String STseleccionado = "";
```

Procedimiento por el que se lista los tipos de servicio. Gracias ha esto el usuario dispone de información suficiente para consultarlos.

```
        void ListST(Object o, EventArgs e)
        {
            TemplateHeader TemplateHeader_soap = new TemplateHeader();

            TemplateHeader_soap.username =
"usernameCARMELO";//User.Identity.Name;
            TemplateHeader_soap.password =
"passwordCARMELO";//User.Identity.IsAuthenticated;
            TemplateHeader_soap.created = DateTime.UtcNow;
            pfc_webservice.WebServiceTrader proxy = new
pfc_webservice.WebServiceTrader();
            proxy.TemplateHeader_aux = TemplateHeader_soap;
            // Call the method on the proxy class that communicates with your Web
service method.
            dts = proxy.List_ServicesTypes();
            try
            {
                values.Clear();
                ListServicesTypes.Items.Clear();
                values.Add("-");
                foreach (DataRow dr in dts.Tables[0].Rows)
                {
                    values.Add(dr["nameServiceType"].ToString());
                }
                ListServicesTypes.DataSource = values;
```

```
                try
                {
                    ListServicesTypes.DataBind();
                    ListServicesTypes.Enabled = true;
                }
                catch
                {
                    ReturnValue.Text = "No se ha podido recuperar la lista de
tipos de servicio. Comuniquelo al administrador.";
                    ListServicesTypes.Enabled = false;
                }
            }
            catch
            {
                ReturnValue.Text = "No se ha podido recuperar la lista de tipos
de servicio. Comuniquelo al administrador.";
                ListServicesTypes.Enabled = false;
            }
        }
```

Aquí se procede a la consulta de la información suplementaria de los tipos de servicio. De esta manera el usuario los conoce suficientemente como para hacer uso de ellos.

```
        void List_SelectedIndexChanged(Object o, EventArgs e)
        {

            if (ListServicesTypes.SelectedIndex > -1)
            {
                Cache["Cache_STseleccionado"] =
ListServicesTypes.SelectedItem.Text;
                if ("-" == ListServicesTypes.SelectedItem.Text) {
                    ReturnValue.Text = "";
                    return;
                }
                String expresion = "nameServiceType = '" +
ListServicesTypes.SelectedItem.Text + "'";
                DataRow[] dr = dts.Tables[0].Select(expresion);
                String xml = dr[0].ItemArray.GetValue(2).ToString();

                XDocument xmldoc = XDocument.Parse(xml);
                var result =
                  from c in xmldoc.Elements("serviceType")
                  select c.Element("annotation").Value;

                ReturnValue.Text = result.Single();
            }
        }
```

Las consultas al Trader se hacen a través de su interfaz lookup, la cual conecta en el objeto cliente mediante este procedimiento.

```
        void lookup(Object o, EventArgs e)
        {
```

```
            label_ordenacion.Text = "Alterne una ordenación ascendente /
descendente de un campo clicándolo.";//La ordenación se lleva a cabo después de
la consulta.

            TemplateHeader_lookup TemplateHeader_soap = new
TemplateHeader_lookup();
            // Valores del TemplateHeader_soap.
            TemplateHeader_soap.username =
"usernameCARMELO";//User.Identity.Name;
            TemplateHeader_soap.password =
"passwordCARMELO";//User.Identity.IsAuthenticated;
            TemplateHeader_soap.created = DateTime.UtcNow;
            TemplateHeader_soap.nombre = tb_nombre.Text;
            TemplateHeader_soap.categoria = ddl_categoria.SelectedValue;
            TemplateHeader_soap.propiedad = tb_propiedad.Text;
            TemplateHeader_soap.valor = tb_valor.Text;
            TemplateHeader_soap.cardinalidad = ddl_cardinalidad.SelectedValue;
            TemplateHeader_soap.TS = (String)Cache["Cache_STseleccionado"];
            if (TemplateHeader_soap.TS == null) TemplateHeader_soap.TS = "-";
            pfc_webservice.WebServiceTrader proxy = new
pfc_webservice.WebServiceTrader();
            proxy.TemplateHeader_lookup_aux = TemplateHeader_soap;
            dts = proxy.Lookup();
            Cache["Cache_dts_lookup"] = dts;
```

El siguiente código corresponde con la búsqueda ampliada. Dicha concepto consiste en en la jerarquía entre TS. Esto es, al registrar un TS se puede especificar otro que semánticamente hablando sea de funcionalidad similar y más amplia. Por ejemplo, un desarrollador crea un componente con la funcionalidad (especializada por cualquier motivo) de sumar y restar números enteros. Lo registra en el trader bajo el TS de "calculadora simple de enteros", sería conveniente que indicara al TS "calculadora simple de reales" como superior en la jerarquía. De este modo, cuando un cliente pregunte al trader si tiene algún servicio de calculadora de enteros, si la oferta de componentes del TS "…enteros" no se puede usar porque ya no esté disponible o sea muy cara, el sistema puede ofrecer la alternativa de las ofertas disponibles del TS "…reales" con un porcentaje de certeza elevado, de que este otro tipo de servicio le puede ser útil al cliente.

```
            if (CheckBox1.Checked == true && TemplateHeader_soap.TS != "-"){
                try
                {
                    dts2 = proxy.List_ServicesTypes();
                    XDocument xml = new XDocument();
                    foreach (DataRow a in dts2.Tables[0].Rows)
                    {
                        xml = XDocument.Parse(a[2].ToString());
                        var result =
                            from c in xml.Elements("serviceType")
                            select c.Element("typeName").Value;
                        if(result.Single() ==
(String)Cache["Cache_STseleccionado"]){
```

```
                            TemplateHeader_soap.TS =
xml.Element("serviceType").Element("top_hierarchy").Value;
                            proxy.TemplateHeader_lookup_aux =
TemplateHeader_soap;
                            dts3.Clear();
                            dts3 = proxy.Lookup();

                            try
                            {
                                XMLgv_busqAvanzada.Columns[0].Visible = false;
                                XMLgv_busqAvanzada.Columns[1].Visible = false;
                                XMLgv_busqAvanzada.Columns[2].Visible = true;
                                XMLgv_busqAvanzada.Columns[3].Visible = true;
                                XMLgv_busqAvanzada.Columns[4].Visible = true;
                                XMLgv_busqAvanzada.DataSource =
dts3.Tables[0].DefaultView;
                                try
                                {
                                    XMLgv_busqAvanzada.DataBind();
                                    XMLgv_busqAvanzada.AllowPaging = false;
                                    XMLgv_busqAvanzada.Enabled = true;
                                    label.Visible = true;
                                }
                                catch
                                {
                                    ReturnError.Text = "No se han encontrado
coincidencias.";
                                    XMLgv_busqAvanzada.Enabled = false;
                                }
                            }catch{
                                ReturnError.Text = "Posiblemente la sintaxis de
la consulta es incorrecta.";
                                XMLgv_busqAvanzada.Enabled = false;
                            }
                        }
                    }
                }
                catch
                {
                    ReturnValue.Text = "No se ha podido recuperar la lista de
tipos de servicio. Comuniquelo al administrador.";
                }
            }
```

Este código corresponde a la capa de presentación. Las tablas que se muestran al usuario se adaptan a los resultados que mostrarán de la siguente forma:

```
            if (dts.Tables[0].TableName == "STR") {
                XMLgv.Columns[0].Visible = true;
                XMLgv.Columns[1].Visible = true;
                XMLgv.Columns[2].Visible = false;
                XMLgv.Columns[3].Visible = false;
                XMLgv.Columns[4].Visible = false;
            }
            if (dts.Tables[0].TableName == "RO")
            {
                XMLgv.Columns[0].Visible = false;
```

```
                XMLgv.Columns[1].Visible = false;
                XMLgv.Columns[2].Visible = true;
                XMLgv.Columns[3].Visible = true;
                XMLgv.Columns[4].Visible = true;
            }
```

Uno de los requisitos entre los que se estaba interesado era proveer a la web de un sistema de paginación, de manera, que el usuario puediera elegir la cantidad de información que quería recibir y de esta forma adaptarla a sus necesidades.
Además se implementa la casuística necesaria para que la presentación de la web sea coherente en todo momento, existan suficientes resultados que mostrar o no.

```
            if (ddl_paginacion.SelectedValue == "-")
            {
                XMLgv.AllowPaging = false;
            } else {
                XMLgv.AllowPaging = true;
                XMLgv.PageSize =
System.Convert.ToInt32(ddl_paginacion.SelectedValue);
            }

            try
            {
                XMLgv.DataSource = dts.Tables[0].DefaultView;
                try
                {
                    XMLgv.DataBind();
                    XMLgv.Enabled = true;
                }
                catch
                {
                    ReturnError.Text = "No se han encontrado coincidencias.";
                    XMLgv.Enabled = false;
                }
            }
            catch
            {
                ReturnError.Text = "Posiblemente la sintaxis de la consulta es
incorrecta.";
                XMLgv.Enabled = false;
            }
        }
```

Este código posibilita que el usuario pagine unos resultados de distintas formas sin tener que realizar sendas consutlas.

```
        void ddl_paginacion_SelectedIndexChanged(Object sender, EventArgs
e)//Paginación es una preferencia de presentación
        {
            if (ddl_paginacion.SelectedValue == "-")
            {
                XMLgv.AllowPaging = false;
            }
            else
            {
```

```
                XMLgv.AllowPaging = true;
                XMLgv.PageSize =
System.Convert.ToInt32(ddl_paginacion.SelectedValue);

                dts = (DataSet)Cache["Cache_dts_lookup"];
                if (dts != null)
                {
                    XMLgv.DataSource = dts.Tables[0].DefaultView;
                    XMLgv.DataBind();
                }
            }
        }

    </script>
```

Información de uno de los resultados.

```
        void XMLgv_SelectedIndexChanged(Object sender, EventArgs e)
        {

            GridViewRow row = XMLgv.SelectedRow;

            int nrow = row.Cells.Count;
            string a = "";
            try
            {
                a = row.Cells[nrow-1].Text.Replace("&lt;", "<");
                a = a.Replace("&quot;", "\"");
                a = a.Replace("&gt;", ">");
            }
            catch {
                ReturnValue.Text = "Item seleccionado.";
            }
            try
            {
                XMLview.DocumentContent = a;
                XMLview.Visible = true;
            }
            catch
            {
                ReturnValue.Text = "La información se ha encontrado corrupta.";
            }
            ReturnValue.Text = "Item seleccionado, mostrado con xsl básico";
        }

        void XMLgv_PageIndexChanging(Object sender, EventArgs e)
        {
            dts = (DataSet)Cache["Cache_dts_lookup"];
            if (dts != null)
            {
                XMLgv.DataSource = dts.Tables[0].DefaultView;
                XMLgv.PageIndex =
((System.Web.UI.WebControls.GridViewPageEventArgs)(e)).NewPageIndex;
                XMLgv.DataBind();
            }
        }
```

Uno de los requisitos del Trader básico estudiado desde la OMG consistía en devolver los resultados al usuario de forma ordenada. Aquí el usuario puede reordenar los resultados desde la misma página según cualquier criterio que desee.

```
        void XMLgv_Sorting(Object sender, GridViewSortEventArgs e)
        {
            dts = (DataSet)Cache["Cache_dts_lookup"];
            if (dts != null)
            {
                if (dts.Tables[0].DefaultView.Sort == "" ||
dts.Tables[0].DefaultView.Sort.Contains(" ASC"))
                {
                    dts.Tables[0].DefaultView.Sort = e.SortExpression + " DESC";
                    label_ordenacion.Text = "Ordenación por " +
e.SortExpression.ToString() + " en orden descendente.";
                }
                else
                {
                    dts.Tables[0].DefaultView.Sort = e.SortExpression + " ASC";
                    label_ordenacion.Text = "Ordenación por " +
e.SortExpression.ToString() + " en orden ascendente.";
                }

                XMLgv.DataSource = dts.Tables[0].DefaultView;
                XMLgv.DataBind();
            }
        }
```

La interfaz gráfica para con el usuario viene dada por el siguiente HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">

    <link rel="stylesheet" href="StyleSheet1.css"/>

    <title>Web Service Trader</title>


    <style type="text/css">
        .style1
        {
            width: 160px;
        }
        .style3
        {
            width: 261px;
            color: #000000;
            font-weight: bold;
            font-size: xx-small;
        }
        .style4
        {
            width: 238px;
```

```
            color: #000000;
            font-weight: bold;
            font-size: xx-small;
        }
        .style6
        {
            width: 868px;
        }
        #t_nombre
        {
            width: 164px;
            margin-left: 0px;
        }
        #t_nombre0
        {
            width: 164px;
            margin-left: 0px;
        }
        #t_nombre0
        {
            width: 164px;
            margin-left: 0px;
        }
        .style9
        {
            height: 16px;
        }
        .style10
        {
            width: 169px;
            color: #000000;
            font-weight: bold;
            font-size: xx-small;
        }
        .style11
        {
            height: 16px;
            width: 160px;
        }
        .style12
        {
            height: 16px;
            width: 161px;
        }
    </style>

</head>

<body>

    <div id="content">

      <p class="heading1">Web Profesional - Lookup</p><br/>

          <p class="intro">Complete la plantilla.<form id="form1" runat="server"
oninit="ListST">
                        <li>
                            <asp:Button ID="button_lookup" Runat="server"
OnClick="lookup" Text="Buscar oferta" CssClass="josebutton" />
```

```
                        </li>
                        <br />
                            <asp:Label id="ReturnValue" Runat="server"
CssClass="information" />
                            <br />
                            <asp:Label id="ReturnError" Runat="server"
CssClass="informationError"/>
                            <br />
                            <table style="width: 100%; height: 175px;">
                                <tr>
                                    <td class="style6">
                                        PARÁMETROS DE SU CONSULTA</td>
                                    <td>
                                        TIPOS DE SERVICIOS REGISTRADOS</td>
                                </tr>
                                <tr>
                                    <td class="style6">
                                        <table>
                                            <tr>
                                                <td class="style12"
```

En las consultas se puede especificar el **nombre**, **categoría**, y **propiedades** de las ofertas, que serán mostradas al usuario bajo su demanda de **paginación**, **orden**, **cardinalidad** y resultados ampliados (**búsqueda ampliada**) cuando lo requiera.

```
                                                    style="background-color:
#000000; color: #FFFFFF; font-weight: bold; font-size: xx-small; ">
                                                    Nombre</td>
                                                <td class="style11"

                                                    style="background-color:
#000000; color: #FFFFFF; font-weight: bold; font-size: xx-small; ">
                                                Categoría</td>
                                                <td class="style9"

                                                    style="background-color:
#000000; color: #FFFFFF; font-weight: bold; font-size: xx-small; "
                                                    colspan="2">
                                                    Orden (condiciones)<br
/> 
                                                    - Propiedad
- Valor</td>
                                            </tr>
                                            <tr>
                                                <td class="style12">
                                                    <asp:TextBox ID="tb_nombre"
runat="server"></asp:TextBox>
                                                    </td>
```

Se listan los distintos tipos de ofertas para que el usuario elija de entre ellas para realizar sus consultas.

```
                                                <td class="style11">
                                                    <asp:DropDownList
ID="ddl_categoria" runat="server" AppendDataBoundItems="True"
```

```
                                                    style="margin-left: 0px"
Width="162px">
                                                    <asp:ListItem
Selected="True" Value="*">-</asp:ListItem>
                                                    <asp:ListItem
Value="1">Shop</asp:ListItem>
                                                    <asp:ListItem
Value="2">Communication</asp:ListItem>
                                                    <asp:ListItem
Value="3">Sports</asp:ListItem>
                                                    <asp:ListItem
Value="4">People</asp:ListItem>
                                                    <asp:ListItem
Value="5">Health</asp:ListItem>
                                                    <asp:ListItem
Value="6">Lifestyle</asp:ListItem>
                                                    <asp:ListItem
Value="7">Finance</asp:ListItem>
                                                    <asp:ListItem
Value="8">Tools</asp:ListItem>
                                                    <asp:ListItem
Value="9">Multimedia</asp:ListItem>
                                                    <asp:ListItem
Value="10">News and meteorology</asp:ListItem>
                                                    <asp:ListItem
Value="11">Learning</asp:ListItem>
                                                    <asp:ListItem
Value="12">Productivity</asp:ListItem>
                                                    <asp:ListItem
Value="13">Free time</asp:ListItem>
                                                    <asp:ListItem
Value="14">Reference</asp:ListItem>
                                                    <asp:ListItem
Value="15">Trips</asp:ListItem>
                                                    <asp:ListItem
Value="16">Themes</asp:ListItem>
                                                    <asp:ListItem
Value="17">Others</asp:ListItem>
                                                </asp:DropDownList>
                                            </td>
                                            <td class="style9">
                                                <asp:TextBox
ID="tb_propiedad" runat="server"></asp:TextBox>
                                            </td>
                                            <td class="style10">
                                                <asp:TextBox ID="tb_valor"
runat="server"></asp:TextBox>
                                            </td>
                                        </tr>
                                    </table>
                                </td>
```

Aquí se presenta el listado de tipos de servicio registrados.

```
                                <td rowspan="2">
                                    <asp:ListBox ID="ListServicesTypes"
runat="server" Height="100%" Width="200px"
```

```
OnSelectedIndexChanged="List_SelectedIndexChanged" AutoPostBack="true"
                                    SelectionMode="Single" style="margin-top:
0px"></asp:ListBox>
                                </td>
                            </tr>
                            <tr>
                                <td class="style6">
                                    <table>
                                        <tr>
```

La entrada de datos para las consultas desde la capa de presentación.

```
                                            <td class="style11"
style="background-color: #000000; color: #FFFFFF; font-weight: bold; font-size:
xx-small; ">
                                Paginación</td>
                                            <td class="style1"
                                style="background-color: #000000; color:
#FFFFFF; font-weight: bold; font-size: xx-small; width: 160px;">
                                Cardinalidad</td>
                                            <td class="style3"
                                style="background-color: #000000; color:
#FFFFFF; font-weight: bold; font-size: xx-small; ">
                                Orden (max / min)</td>
                                            <td class="style4"

                                style="background-color: #000000; color:
#FFFFFF; font-weight: bold; font-size: xx-small; ">
                                Búsqueda ampliada</td>
                                        </tr>
                                        <tr>
                                            <td class="style11">
                                                <asp:DropDownList
ID="ddl_paginacion" runat="server"
                                    AppendDataBoundItems="True"
AutoPostBack="True" OnSelectedIndexChanged="ddl_paginacion_SelectedIndexChanged">
                                                    <asp:ListItem Value="-">-
</asp:ListItem>

<asp:ListItem>2</asp:ListItem>

<asp:ListItem>5</asp:ListItem>
                                                    <asp:ListItem
Selected="True">10</asp:ListItem>

<asp:ListItem>20</asp:ListItem>
                                                </asp:DropDownList>
                                            </td>
                                            <td class="style1">
                                                <asp:DropDownList
ID="ddl_cardinalidad" runat="server" AppendDataBoundItems="True">
                                                    <asp:ListItem
Selected="True" Value="-">-</asp:ListItem>

<asp:ListItem>2</asp:ListItem>

<asp:ListItem>5</asp:ListItem>
```

```
<asp:ListItem>10</asp:ListItem>

<asp:ListItem>20</asp:ListItem>

<asp:ListItem>50</asp:ListItem>
                                                    </asp:DropDownList>
                                                </td>
                                                <td class="style3">
                                                    <asp:Label
ID="label_ordenacion" runat="server"></asp:Label>
                                                </td>
                                                <td class="style4">
                                                    <asp:CheckBox ID="CheckBox1"
runat="server" Text="On / Off" />
                                                </td>
                                            </tr>
                                        </table>
                                    </td>
                                </tr>
                            </table>
                            <br />
```

Los resultados se mostrarán en objetos GridView.

```
                            <asp:GridView ID="XMLgv" runat="server"
AutoGenerateColumns="false" AutoGenerateSelectButton="true"
                            CellPadding="4" ForeColor="#333333" GridLines="None"
emptydatatext="Consulta sin resultado." EnableModelValidation="True"
                            OnSelectedIndexChanged="XMLgv_SelectedIndexChanged"
OnPageIndexChanging="XMLgv_PageIndexChanging"
                            OnSorting="XMLgv_Sorting" AllowPaging="True"
PageSize="2" Font-Bold="True" AllowSorting="True">
                                <RowStyle BackColor="#F7F6F3" ForeColor="#333333"
/>
                                <AlternatingRowStyle BackColor="White"
ForeColor="#284775" />
                                <EditRowStyle BackColor="#999999" />
                                <FooterStyle BackColor="#E4CFD0" Font-Bold="True"
ForeColor="White" />
                                <HeaderStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
                                <PagerSettings FirstPageText="Primera"
LastPageText="Última" NextPageText="Siguiente" PreviousPageText="Anterior" />
                                <PagerStyle BackColor="#E4CFD0" ForeColor="White"
HorizontalAlign="Center" Font-Bold="True" Font-Italic="False" Font-
Names="Andalus" Font-Overline="False" Font-Size="Large" />
                                <SelectedRowStyle BackColor="#E2DED6" Font-
Bold="True" ForeColor="#333333" />
                                <Columns>
                                    <asp:BoundField HeaderText="Tipo de servicio"
DataField="nameServiceType" SortExpression="nameServiceType" Visible="false" />
                                    <asp:BoundField HeaderText="PLANTILLA"
DataField="ServiceType" SortExpression="ServiceType" Visible="false" />
                                    <asp:BoundField HeaderText="ID oferta"
DataField="id_service" SortExpression="id_service" Visible="false" />
                                    <asp:BoundField HeaderText="Categoría"
DataField="category_range" SortExpression="category_range" Visible="false" />
```

```
                                    <asp:BoundField HeaderText="PLANTILLA"
DataField="template" SortExpression="template" Visible="false" />
                                </Columns>
                            </asp:GridView>
                        <br />
                        <asp:Label ID="label" runat="server" Visible="False"
BackColor="#003366"
                                ForeColor="White">Búsqueda ampliada</asp:Label>
                        <br />
                        <asp:Label ID="l_busqAvanzada" runat="server"
Text=""></asp:Label>
                        <br />
                        <asp:GridView ID="XMLgv_busqAvanzada" runat="server"
AutoGenerateColumns="false" AutoGenerateSelectButton="true"
                        CellPadding="4" ForeColor="#333333" GridLines="None"
                            emptydatatext="Consulta sin resultado."
EnableModelValidation="True"
                        OnSelectedIndexChanged="XMLgv_SelectedIndexChanged"
OnPageIndexChanging="XMLgv_PageIndexChanging"
                        OnSorting="XMLgv_Sorting" AllowPaging="False"
PageSize="2" Font-Bold="True"
                            AllowSorting="True">
                            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
                            <AlternatingRowStyle BackColor="White"
ForeColor="#284775" />
                            <EditRowStyle BackColor="#999999" />
                            <FooterStyle BackColor="#E4CFD0" Font-Bold="True"
ForeColor="White" />
                            <HeaderStyle BackColor="#5D7B9D" Font-Bold="True"
ForeColor="White" />
                            <PagerSettings FirstPageText="Primera"
LastPageText="Última" NextPageText="Siguiente" PreviousPageText="Anterior" />
                            <PagerStyle BackColor="#E4CFD0" ForeColor="White"
HorizontalAlign="Center" Font-Bold="True" Font-Italic="False" Font-
Names="Andalus" Font-Overline="False" Font-Size="Large" />
                            <SelectedRowStyle BackColor="#E2DED6" Font-
Bold="True" ForeColor="#333333" />
                            <Columns>
                                <asp:BoundField HeaderText="Tipo de servicio"
DataField="nameServiceType" SortExpression="nameServiceType" Visible="false" />
                                <asp:BoundField HeaderText="PLANTILLA"
DataField="ServiceType" SortExpression="ServiceType" Visible="false" />
                                <asp:BoundField HeaderText="ID oferta"
DataField="id_service" SortExpression="id_service" Visible="false" />
                                <asp:BoundField HeaderText="Categoría"
DataField="category_range" SortExpression="category_range" Visible="false" />
                                <asp:BoundField HeaderText="PLANTILLA"
DataField="template" SortExpression="template" Visible="false" />
                            </Columns>
                        </asp:GridView>
                        <br />
```

El siguiente código corresponde a un área prueba, que dejo disponible para posteriores desarrolladores.

```
                        <textarea id="inputTemplateQuery" runat="server"
style="height:12px; width:117px"
```

```
                                    cols="20" name="S1" rows="1"
visible="False">--Admite SQL y XQUERY
USE pfc_repositorio
SELECT *
FROM RO;

/*
SELECT template.query('/serviceTDI/functional/requirements/OS') as Result
FROM RO
WHERE category_range=3
*/
                        </textarea><br />
                            <br />
                            <asp:Xml ID="XMLview" runat="server">
                        </asp:Xml>


                    </form>

     </div>

</body>
</html>
```

25. Código: Objeto cliente principal

Da acceso a los objetos clientes descritos en los dos puntos anteriores. Además contiene un procedimiento que he llamado “Estado del servicio” el cual comprueba las conexiones entre todas las partes del proyecto: objetos clientes, servidor, trader, repositorios, etc. Usando los mecanismos reales de comunicación mediante mensajes SOAP para corroborar su buen funcionamiento.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Web_Profesional.aspx.cs" Inherits="pfc_webservice.Web_Profesional" %>


    <script runat="server" language="c#">
        void Heartbeat(Object o, EventArgs e)
        {
            TemplateHeader TemplateHeader_soap = new TemplateHeader();

            TemplateHeader_soap.username =
"usernameCARMELO";//User.Identity.Name;
            TemplateHeader_soap.password =
"passwordCARMELO";//User.Identity.IsAuthenticated;
            TemplateHeader_soap.created = DateTime.UtcNow;
            pfc_webservice.WebServiceTrader proxy = new
pfc_webservice.WebServiceTrader();
            proxy.TemplateHeader_aux = TemplateHeader_soap;
            string results = proxy.Heartbeat();
            ReturnValue.Text = results;
        }
    </script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">

    <link rel="stylesheet" href="StyleSheet1.css"/>

    <title>Web Service Trader</title>
</head>
<body>
    <div id="content">
      <p class="heading1">Web Profesional</p><br/>

          <p class="intro">Las operaciones siguientes son ofrecidas por el
servicio web. Para una definición formal, revise la <a
href="WebServiceTrader.asmx?WSDL">descripción de servicios</a>. </p>
```

El protocolo de comunicación usado sobre HTTP para el envio de mensajes SOAP es HTTP POST [2].

```
                <form id="form1" runat="server">
                    <asp:Button ID="button_estadoServicio" Runat="server"
OnClick="Heartbeat" Text="Estado del servicio" CssClass="josebutton" />
                    <asp:Label id="ReturnValue" Runat="server"
CssClass="frmtext"/>
                </form>
                <li>
                    <a href="http://localhost:50162/Web_register.aspx"
class="josebutton">Register</a>
                </li>

                <li>
                    <a href="http://localhost:50162/Web_lookup.aspx"
class="josebutton">Lookup</a>
                </li>

        <hr />

        <h3>El espacio de nombres provisional utilizado por el servicio Web es
http://tempuri.org/. </h3>
        <p class="intro">Cada servicio Web XML necesita un espacio de nombres
único para que las aplicaciones de cliente puedan distinguir este servicio de
otros servicios del Web. http://tempuri.org/ está disponible para servicios Web
XML que están en desarrollo, pero los servicios Web XML publicados deberían
utilizar un espacio de nombres más permanente.</p>
        <p class="intro">Debemos identificar el servicio Web XML con un espacio
de nombres propio. Por ejemplo, puede utilizar el nombre de dominio de Internet
de su compañía como parte del espacio de nombres. aunque muchos espacios de
nombres de servicios Web XML parecen direcciones URL, éstos no pueden señalar a
recursos reales en el Web. (Los espacios de nombres de los servicios Web XML son
los URI.)</p>
```

Aquí se lleva a cabo un ejemplo de presentación para la Web final de carácter comercial. (No compete en este proyecto llevarla a cabo)

```
        <p class="intro">Row</p>
        <pre>Data row // Ejemplo de presentación.
        </pre>
        <iframe id="iframe_out" width="100%" frameborder="0" >
        </iframe>

          <p class="intro">Para obtener más detalles acerca de espacios de
nombres XML, vea la sugerencia W3C en <a href="http://www.w3.org/TR/REC-xml-
names/"">Espacio de nombres en XML</A>.</p>
          <p class="intro">Para obtener más detalles acerca de WSDL, vea <a
href="http://www.w3.org/TR/wsdl">Especificación WSDL</a>.</p>
          <p class="intro">Para obtener más detalles sobre los URI, vea <a
href="http://www.ietf.org/rfc/rfc2396.txt">RFC 2396</a>.</p>

     </div>
</body>
</html>
```

26. Código: Servidor Web

El servidor web aloja principalmente el código básico de interacción con los objetos clientes, la definición de los mensajes SOA y el Trader con sus interfaces.

Las librerías necesarias para el servidor web contenedor del Trader se listan a continuación. En primer lugar las predeterminadas de toda aplicación:

```
using System;
using System.Collections.Generic;
```

Librerías para manejo de documentos XML en C#, las librerías LINQ.

```
using System.Linq;
using System.Xml.Linq;
```

Librerías para soporte de servicios web y protocolos como SOAP

```
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.UI.WebControls;
```

Por último las librerías para la conexión al repositorio. En este caso se trata de una conexión al motor de BBDD de SQL Server

```
using System.Data;
using System.Data.SqlClient;
```

26.1 Código: SOA

Definición de la cabecera SOAP heredando desde la clase SoapHeader.

```
    public class TemplateHeader : SoapHeader
    {
        public string username;
        public string password;
        public DateTime created;
    }
```

A continuación se especifican las definiciones de la cabecera SOAP para los tipos de paquetes de dicho protocolo que se usarán para comunicar los registros de ofertas y tipos de servicios. Además de las consultas estableciendo la propia consulta, el nombre, categoría, propiedad, valor, cardinalidad, etc. (Véase el siguiente código)

La política de orden no se especifica aquí, ya que, por motivos de eficiencia y manejo útil por parte del usuario, el orden se puede establecer directamente desde el objeto cliente, sin necesidad de interactuar reiteradamente con el repositorio.

```
    public class TemplateHeader_register_Offer : TemplateHeader
    {
        public string xml_template;
        public string xml_template_ir;
        public string category_range;
    }

    public class TemplateHeader_register_ServiceType : TemplateHeader
    {
        public string nameServiceType;
        public string xml_serviceType;
    }

    public class TemplateHeader_lookup : TemplateHeader
    {
        public string query;
        public string nombre;
        public string categoria;
        public string propiedad;
        public string valor;
        public string cardinalidad;
        public string TS;
        public bool   chekado;
    }
```

Es muy común usar la etiqueta **<sumary>** como anotación sobre el tipo de proyecto software que se desarrolla. En este caso sería Descripción WebServiceTrader. Servicio de Trader sobre componentes dirigidos a la televisión digital, alojado en un servicio Web.

En el lado servidor, esta es la asignación de un tipo de cabecera de un mensaje como entrada de argumentos de un método del servidor Web.
En el trader se recibe el mensaje SOAP del cliente, accede a sus datos, interactúa con el repositorio y devuelve, de nuevo a través de su proxy, el resultado de la intervención. Como puede ser un conjunto de ofertas, un mensaje de error de parseo de los XML tipados, etc.

```
namespace pfc_webservice
{

    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]

    public class WebServiceTrader : System.Web.Services.WebService
    {
        private string  connectionString = "Data Source=Jse_pc;Initial
Catalog=pfc_repositorio;Integrated Security=True";
```

```
        private string  catalog = "pfc_repositorio";
        private int     rowAffected;
        private int     nmaxProperties = 99;
```

Añade una variable del tipo derivado de SoapHeader.

```
        public TemplateHeader TemplateHeader_aux;
        public TemplateHeader_register_Offer TemplateHeader_register_Offer_aux;
        public TemplateHeader_register_ServiceType
TemplateHeader_register_ServiceType_aux;
        public TemplateHeader_lookup TemplateHeader_lookup_aux;
```

26.2 Código: Interfaces del servidor web

```
        [WebMethod]
        [SoapHeader("TemplateHeader_aux")]
        public string Heartbeat()
        {
            //if (TemplateHeader_aux.username != "")//Sólo los usuarios
registrados podrían acceder a este servicio.
            //{
                string servidor = "Conexión al servidor: OK! ";
                SqlConnection connection = new SqlConnection(connectionString);

                if (catalog == connection.Database)
                {
                    return servidor + " Conexión al repositorio: OK! ";
                }
                else
                {
                    connection.Close();
                    return servidor + " Conexión al repositorio: Sin conexión.";
                }
        }
```

```
        [WebMethod]
        [SoapHeader("TemplateHeader_aux")]
        public DataSet List_ServicesTypes()
        {
```

El siguiente código representa una puerta de identificación para que sólo los posibles usuarios registrados puedan acceder a este servicio.

```
            //if (TemplateHeader_aux.username != "")
            string sql = "USE pfc_repositorio SELECT * FROM STR WHERE available =
0;";//Listado de los tipos de servicio disponibles (no borrados)
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);
```

```
            DataSet dataset = new DataSet("result_ListST");
            try
            {
                adapter.FillSchema(dataset, SchemaType.Source, "STR");
                adapter.Fill(dataset, "STR");
            }
            catch { }
            if (connection.State != ConnectionState.Closed)
            {
                connection.Close();
            }

            return dataset;

        }
```

26.3 Código: Componentes del Trader

Este procedimiento lleva a cabo la multiherencia entre tipos de sericio, según haya sido indicado en el documento XML proporcionado por el usuario.

```
        private string inheritance_TS(string xml_serviceType)
        {
            XDocument xml_hijo = new XDocument();
            xml_hijo = XDocument.Parse(xml_serviceType);
            System.Collections.Generic.IEnumerable<XElement> enumeracion_st =
xml_hijo.Element("serviceType").Elements("inheritance");
            SqlConnection connection = new SqlConnection(connectionString);
            foreach (XElement st in enumeracion_st)
            {
                var TSpadre =
                    from c in st.Elements("name")
                    select c.Value;
                string sql = "USE pfc_repositorio SELECT * FROM STR WHERE
available = 0 and nameServiceType = '" + TSpadre.Single() + "';";
                connection.Open();
                SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);
                DataSet dataset = new DataSet("result_ListST");
                try
                {
                    adapter.FillSchema(dataset, SchemaType.Source, "STR");
                    adapter.Fill(dataset, "STR");
                }
                catch { }

                string string_xml_padre =
dataset.Tables[0].Rows[0][2].ToString();

                XDocument xml_padre = new XDocument();
                xml_padre = XDocument.Parse(string_xml_padre);
                System.Collections.Generic.IEnumerable<XElement>
enumeracion_interfaces = xml_padre.Element("serviceType").Elements("interfaces");
                System.Collections.Generic.IEnumerable<XElement>
enumeracion_propiedades =
xml_padre.Element("serviceType").Elements("properties");
```

```
xml_hijo.Element("serviceType").Element("interfaces").AddBeforeSelf(enumeracion_i
nterfaces);

xml_hijo.Element("serviceType").Element("properties").AddBeforeSelf(enumeracion_p
ropiedades);
            }

            if (connection.State != ConnectionState.Closed)
            {
                connection.Close();
            }

            return xml_hijo.ToString();
        }
```

Comprueba la consistencia de la herencia en ofertas.

```
        private string inheritance_OFFER(string xml_offer)
        {
            XDocument xml_doc_offer = new XDocument();
            xml_doc_offer = XDocument.Parse(xml_offer);
            System.Collections.Generic.IEnumerable<XElement> TS_padre =
xml_doc_offer.Element("serviceTDI").Elements("no_functional");
            SqlConnection connection = new SqlConnection(connectionString);

            XElement st = TS_padre.ElementAt(0);
            var TSpadre =
                from c in st.Elements("inheritance")
                select c.Value;
            string sql = "USE pfc_repositorio SELECT * FROM STR WHERE available =
0 and nameServiceType = '" + TSpadre.Single() + "';";
            connection.Open();
            SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);
            DataSet dataset = new DataSet("result_ListST");
            try
            {
                adapter.FillSchema(dataset, SchemaType.Source, "STR");
                adapter.Fill(dataset, "STR");
            }
            catch { }

            string string_xml_padre = dataset.Tables[0].Rows[0][2].ToString();
            XDocument xml_padre = new XDocument();
            xml_padre = XDocument.Parse(string_xml_padre);
            System.Collections.Generic.IEnumerable<XElement> enu_TS_propiedades =
xml_padre.Element("serviceType").Elements("properties");

            XDocument xml_propi_of = new XDocument();
            xml_propi_of = XDocument.Parse(xml_offer);
            System.Collections.Generic.IEnumerable<XElement> enu_OF_propiedades =
xml_propi_of.Element("serviceTDI").Element("no_functional").Elements("properties"
);

            string[] TS_propiedad_names;
            TS_propiedad_names = new string[nmaxProperties];
            int i = 0;
            foreach (XElement prop in enu_TS_propiedades)
```

```
            {
                var TS_propiedad =
                    from c in prop.Elements("name")
                    select c.Value;

                TS_propiedad_names[i] = TS_propiedad.Single();
                i++;
            }

            string[] OF_propiedad_names;
            OF_propiedad_names = new string[nmaxProperties];
            i = 0;
            foreach (XElement prop in enu_OF_propiedades)
            {
                var OF_propiedad =
                    from c in prop.Elements("name")
                    select c.Value;

                OF_propiedad_names[i] = OF_propiedad.Single();
                i++;
            }

            string nombres_extraños ="";
            foreach (string OF_propiedad_name in OF_propiedad_names)
            {
                if (!TS_propiedad_names.Contains(OF_propiedad_name))
                {
                    nombres_extraños = nombres_extraños + OF_propiedad_name + ",
";
                }
            }
            if (nombres_extraños != "") {
                nombres_extraños = nombres_extraños.Substring(0,
nombres_extraños.Length - 2);
                return "Todas las propiedades deben ser heredadas de un TS. Error
en: \"" + nombres_extraños + "\".";
            }
            return "";
        }
```

26.4 Código: Interfaces del Trader

26.4.1 Código: Interfaz del Trader – Register offer

```
        [WebMethod]
        [SoapHeader("TemplateHeader_register_Offer_aux")]
        public string Register_Offer()
        {
```

La carga de los datos se podría hacer obviando los objetos cliente y realizarla directamente en el servidor usando un fichero como base.

```
            System.IO.StreamReader file;
            file = System.IO.File.OpenText("*");
            string sql = file.ReadToEnd();
            file.Close();
```

Comprueba la consistencia de la herencia en ofertas.

```
            string error_inheritance =
inheritance_OFFER(TemplateHeader_register_Offer_aux.xml_template.ToString());
            if (error_inheritance != "") {
                return error_inheritance;
            }
```

Lleva a cabo el registro de la oferta, informando al usuario de posibles eventualidades.

```
            string sql = "";
            try
            {
                sql =
                    "USE pfc_repositorio\n" +
                    "INSERT INTO RO (category_range, available, template) VALUES
(" +
                    TemplateHeader_register_Offer_aux.category_range.ToString() +
", " +
                    "0, '" +
                    TemplateHeader_register_Offer_aux.xml_template.ToString() +
"')" +
                    "SELECT SCOPE_IDENTITY();";
            }
            catch {
                return "Complete correctamente los campos.";
            }

            SqlConnection connection = new SqlConnection(connectionString);
            try
            {
                connection.Open();
                SqlCommand command = new SqlCommand(sql, connection);
                rowAffected = command.ExecuteNonQuery();
```

```
            }
            catch (SqlException e)
            {
                string msg = "";
                for (int i = 0; i < e.Errors.Count; i++)
                {
                    msg += "Error #" + i + " Mensaje: " + e.Errors[i].Message +
"<br />";
                }
                return msg;
            }
            finally
            {
                if (connection.State != ConnectionState.Closed)
                {
                    connection.Close();
                }
            }
        return rowAffected + " row affected.";
        }
```

26.4.2 Código: Interfaz del Trader – Register service type

```
        [WebMethod]
        [SoapHeader("TemplateHeader_register_ServiceType_aux")]
        public string Register_serviceType()
        {
```

Llama al procedimiento encargado de llevar a cabo la posible multiherencia entre tipos de servicio, anteriormente implementado.

```
            string xml_serviceType =
inheritance_TS(TemplateHeader_register_ServiceType_aux.xml_serviceType.ToString()
);
```

El siguiente código se emplea para el registro del tipo de servicio.

```
            string sql = "";
            try
            {
                sql =
                    "USE pfc_repositorio\n" +
                    "INSERT INTO STR (nameServiceType, available, ServiceType)
VALUES ('" +

TemplateHeader_register_ServiceType_aux.nameServiceType.ToString() + "', " +
                    "0, '" +
                    xml_serviceType + "');";
            }
            catch
            {
                return "Complete correctamente los campos.";
            }
```

```
            SqlConnection connection = new SqlConnection(connectionString);
            try
            {
                connection.Open();
                SqlCommand command = new SqlCommand(sql, connection);
                rowAffected = command.ExecuteNonQuery();

            }
            catch (SqlException e)
            {
                string msg = "";
                for (int i = 0; i < e.Errors.Count; i++)
                {
                    msg += "Error #" + i + " Mensaje: " + e.Errors[i].Message +
"<br />";
                }
                return msg;
            }
            finally
            {
                if (connection.State != ConnectionState.Closed)
                {
                    connection.Close();
                }
            }
            return rowAffected + " row affected.";
        }
```

26.4.3 Código: Interfaz del Trader – Lookup

La implementación de la interfaz finalmente estriba en las siguientes líneas.

```
        [WebMethod]
        [SoapHeader("TemplateHeader_lookup_aux")]
        public DataSet Lookup()
        {
            string sql = "";

            if (TemplateHeader_lookup_aux.TS != "-")
            {
                sql = "USE pfc_repositorio " +
                      "SELECT id_service, category_range, available,
template.query(' " +
                      "if (/serviceTDI/no_functional/inheritance=\"" +
TemplateHeader_lookup_aux.TS.ToString() + "\") " +
                      "then /serviceTDI else ()') as template " +
                      "FROM RO " +
                      "WHERE available=0;";
            }
            else
            {
                sql = "USE pfc_repositorio " +
```

```
                        "SELECT id_service, category_range, available,
template.query(' " +
                        "if (\"nombre\"=\"nombre\" and
\"propiedad\"=\"propiedad\" and \"valor\"=\"valor\") " +
                        "then /serviceTDI else ()') as template " +
                        "FROM RO " +
                        "WHERE available=0 and
\"category_range\"=\"category_range\";";

                if (TemplateHeader_lookup_aux.nombre != "")
                {
                    sql = sql.Replace("\"nombre\"=\"nombre\"",
"/serviceTDI/OfferName=\"" + TemplateHeader_lookup_aux.nombre + "\"");
                }
                if (TemplateHeader_lookup_aux.propiedad != "")
                {
                    sql = sql.Replace("\"propiedad\"=\"propiedad\"",
"/serviceTDI/no_functional/properties/name=\"" +
TemplateHeader_lookup_aux.propiedad + "\"");
                }
                if (TemplateHeader_lookup_aux.valor != "")
                {
                    sql = sql.Replace("\"valor\"=\"valor\"",
"/serviceTDI/no_functional/properties/value=\"" + TemplateHeader_lookup_aux.valor
+ "\"");
                }
                if (TemplateHeader_lookup_aux.categoria != "*")
                {
                    sql = sql.Replace("\"category_range\"=\"category_range\"",
"category_range=" + TemplateHeader_lookup_aux.categoria);
                }
            }
```

El siguiente código se encarga de llevar a cabo la política de cardinalidad.

```
            if (TemplateHeader_lookup_aux.cardinalidad != "-") {
                sql = sql.Replace("SELECT", "SELECT TOP " +
TemplateHeader_lookup_aux.cardinalidad + " ");
            }

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);
            DataSet dataset = new DataSet("result_lookup");
            try
            {
                if(sql.Contains("FROM STR")){
                    adapter.FillSchema(dataset, SchemaType.Source, "STR");
                    adapter.Fill(dataset, "STR");
                }
                if (sql.Contains("FROM RO"))
                {
                    adapter.FillSchema(dataset, SchemaType.Source, "RO");
                    adapter.Fill(dataset, "RO");
                }
            }
            catch { }
```

```
            if (connection.State != ConnectionState.Closed)
            {
                connection.Close();
            }

            for ( int i=0 ; i < dataset.Tables[0].Rows.Count ; i++ )
            {
                if (dataset.Tables[0].Rows[i].ItemArray[3].ToString() == "") {
                    dataset.Tables[0].Rows[i].Delete();
                }
            }

            return dataset;
        }
    }
}
```

27. Conclusiones y trabajo futuro.

El mercado de componentes tiene un gran futuro. Su problema consiste en hacer fácil el comercio con el cliente y que éste sólo pida un servicio y lo obtenga, sea cual sea la complejidad de crear dicho servicio. A su vez que los desarrolladores ven facilitado su trabajo programando pequeñas aplicaciones. Esto es lo que precisamente intenta resolver el trader. Como hemos visto la solución pasa por asociar a los componentes de información semántica con una organización dispuesta para ser explotada computacionalmente hablando.

Una vez creado el trader con sus dos interfaces básicas, dando servicio desde un servidor ampliamente compatible en la red, el próximo paso consistiría en incrementar las interfaces funcionales del trader o más interesante aún crear mediante MHP interfaces de usuario para acceder tanto al propio trader como a los objetos cliente que acceden al trader actualmente a través del televisor. Aun hoy día, hay más televisores que ordenadores.

28. Recursos

Propios

- **Análisis Web.html**
  Tabla comparativa de Web comerciales en formato .html

- **Gramáticas**
  Escritas en XML de W3C

  - **Multi_idiom.xsd**
    Recurso multilingüe propuesto como 4ª oportunidad de negocio.

  - **RO.xsd**
    Plantilla del COTS para entorno de televisión digital. Juegan el papel de ofertas en el contexto del Trader.

  - **IR.xsd**
    Plantilla de interfaces de las ofertas.

  - **STR.xsd**
    Plantilla del COTS para entorno de televisión digital.

  - **CommentsRO.xsd**
    Plantilla de comentarios sobre ofertas.

- **BD.sql**
  Los scripts para la creación de:

  - **Base de Datos particionada en grupos de archivos**
  - **Colección de gramáticas**
  - **Definición de particiones lógicas**
  - **Repositorios**

Ajenos

- **xletView** (emulador STB)
  http://www.xletview.org/index.php

- **MHP Tester** (emulador STB)
  http://sourceforge.net/projects/mhptester/

- **MHP** (tutoriales MHP)

http://www.code4tv.com/c/tutorialmhp112
http://sourceforge.net/projects/bse/
http://www.mhp-knowledgebase.org/
http://www.mhp-interactive.org/content/code-samples

- **iLab** (emulador hardware externo STB)
  http://ilab.ita.es/index.php?option=com_content&task=view&id=67&Itemid=86

- **CCM – DTD for Software Package Descriptor defined by the CORBA Components Specification**
  http://openccm.ow2.org/dtd/ccm/softpkg.dtd

- **Aplicaciones y ejemplos en MHP**
  Todos los códigos aquí recogidos están destinados a un uso no comercial de los mismos y su objetivo es el de servir como referencia a desarrolladores.

> The examples on this page are designed to give developers an idea of how to use the various MHP APIs.
>
> Copyright © Steven Morris 2002-2010. All Rights Reserved.
> These samples are all copyright Steve Morris, and are **free for non-commercial use**.
> MHP 1.0.3
>
> http://www.mhp-interactive.org/content/code-samples

> MHP Knowledgebase.
> Via this portal, the MHP Knowlege Project offers registered users a variety of resources - free of charge.
>
> http://www.mhp-knowledgebase.org/

> Broadcast Signing Engine es una herramienta basada en Java para firmar digitalmente las aplicaciones DTV como se especifica en la Multimedia Home Platform (MHP). También tiene características de administración de certificados.
> http://sourceforge.net/projects/bse/

> Curso ha sido publicado en la web de mhp.org como una referencia para desarrolladores.
> Tutorial mhp con ejemplos.

Los ejercicios resueltos, con licencia **Creative Commons Attribution-Noncommercial-Share A like 3.0 Unported License**), los cuales son accesibles en cada capítulo (ver más abajo).

Licencia de uso:
http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode
Ejemplos: http://www.code4tv.com/c/tutorialmhp112

- **Emuladores software offline**

  - xletView.
    http://www.xletview.org/index.php

  - mhpTester
    http://sourceforge.net/projects/mhptester/

- **Emuladores online**

En principio se encontraron tres instituciones que proveían de un servicio de emulación de un set top box capacitado para ejecución de código mhp. Éstos pertenecen a un proyecto cofinanciado públicamente: iLAB: Interconexión de laboratiorios de TDT sobre MHP.
http://ilab.ita.es/index.php?option=com_content&task=view&id=67&Itemid=86

Existen estos 3 laboratorios.

- Laboratorio iLAB del Instituto Tecnológico de Aragón.
- Laboratorio i-Lab de la Universidad Politécnica de Madrid.
- Laboratorio iLab de la Salle. (Descartado por no emular un canal de retorno)

29. Fuentes de información

Libros recomendados

- **COTS -- Commercial Off The Shelf**

  A Software Development Process for COTS-Based Information System Infrastructure

  Part I http://www.stsc.hill.af.mil/crosstalk/1998/mar/development.asp
  Greg Fox (TRW Systems Integration Group), Karen Lantner (EDS) and Steven Marcom (TRW Information Services Division)

  Part II http://www.stsc.hill.af.mil/crosstalk/1998/apr/process.asp
  Greg Fox and Steven Marcom (TRW) and Karen W. Lantner (EDS)

- **Service-Oriented Architecture. A Field Guide to Integrating XML and Web Services.**
  Edit. Thomas Erl. ISBN: 0-13-142898-5

- **Professional XML Web Services.**
  Edit. Wrox. ISBN: 1-861005-09-1

Consultas MSDN y Technet. Libros en línea de Microsoft.

http://msdn.microsoft.com/es-ar/default.aspx
http://technet.microsoft.com/es-es/default.aspx

1. **IDL**
   http://www.omg.org/technology/documents/idl2x_spec_catalog.htm
   http://www.omg.org/gettingstarted/omg_idl.htm (OMG)
   http://www.omg.org/cgi-bin/doc?formal/02-06-39 (Especificación)

   MILD (Versión IDL de Microsoft)
   http://msdn.microsoft.com/en-us/library/aa367091(VS.85).aspx

2. **SOAP**
   http://www.w3.org/TR/soap/ (¿Qué es?)
   http://www.w3.org/TR/2007/REC-soap12-part0-20070427/ (Recomendación W3C)
   http://msdn.microsoft.com/es-es/library/77hkfhh8.aspx
   http://msdn.microsoft.com/es-es/library/dkwy2d72.aspx

http://www.w3schools.com/soap/default.asp (¿Cómo funciona?)
http://msdn.microsoft.com/en-us/library/ms995800.aspx
Uso de SOAP en HTTP POST
http://www.w3c.es/Traducciones/es/TR/2003/REC-soap12-part0-20030624/#L26866

Understanding SOAP
Aaron Skonnard (DevelopMentor)
Fecha: Marzo 2003
http://msdn.microsoft.com/en-us/library/ms995800.aspx

3. **COTS**
http://www.cotstrader.com

4. **Trading for COTS Components in Open Environments**
Luis Iribarne, José M. Troya y Antonio Vallecillo.
Fecha: 7 de agosto, 2003.
http://www.computer.org/portal/web/csdl/doi/10.1109/EURMIC.2001.952435
http://www.lcc.uma.es/~av/Publicaciones/04/CJ-trader.pdf

5. **Eclipse PDE Build - Tutorial**
Lars Vogel and Dominik Zapf
Copyright © 2008 - 2009 Lars Vogel
http://www.vogella.de/articles/EclipsePDEBuild/article.html

PDE Does Plug-ins
By Wassim Melhem and Dejan Glozic, IBM Canada Ltd.
Copyright © 2003 International Business Machines Corporation. September 8, 2003
http://www.eclipse.org/articles/Article-PDE-does-plugins/PDE-intro.html

Complementos (Plug-in y extensiones)
http://es.wikipedia.org/wiki/Complemento_(informática)

Eclipse
http://www.chuidiang.com/chuwiki/index.php?title=Instalar_plugins_en_eclipse
http://www.day.com/eclipse/
http://agile.csc.ncsu.edu/SEMaterials/tutorials/plugin_dev/#section2_0
http://conejeala.blogspot.com/2008/05/pde-ejemplo-sencillo-de-plug-inpara.html

Mozilla - Firefox
https://addons.mozilla.org/es-ES/developers
https://developer.mozilla.org/en/Building_an_Extension
https://developer.mozilla.org/en/Install_Manifests
https://developer.mozilla.org/es/XUL

https://addons.mozilla.org/es-ES/developers/addon/validate

XPCOM
https://developer.mozilla.org/en/XPCOM
https://developer.mozilla.org/en/XPCOM/Language_Bindings

6. **Consorcios en Televisión Digital**
   http://www.dvb.org/ (DVB)
   http://www.etsi.org/WebSite/homepage.aspx (ETSI)
   http://www.code4tv.com/res/mhp112course/02-CODE4TV-MHP-QUE-ES.pdf

7. **Sistema de información en MHP**
   http://www.code4tv.com/res/mhp112course/16-CODE4TV-MHP-SI-DVB.pdf

8. **Aplicaciones firmadas y no firmadas en MHP**
   http://www.code4tv.com/res/mhp112course/07-CODE4TV-MHP-APP-SIGNALLING.pdf

9. **Control paterno**
   http://www.fcc.gov/cgb/consumerfacts/vchip.html
   http://www.tvguidelines.org/

10. **Metalenguaje**
    http://es.wikipedia.org/wiki/Metalenguajes

11. **XML**
    http://www.w3.org/standards/xml/
    http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
    http://geneura.ugr.es/~jmerelo/xml/
    http://decsai.ugr.es/~cb/CSharp2007/xml/xml-schema.xml (xml - BBDD)
    http://msdn.microsoft.com/es-es/library/ms171945(VS.80).aspx

12. **XSD**
    http://www.w3.org/2001/03/webdata/xsv (Validador)
    http://www.w3schools.com/Schema/schema_elements_ref.asp (Consulta rápida)
    http://validator.w3.org/#validate_by_uri+with_options
    http://www.xmlforasp.net/SchemaValidator.aspx

    Tipos de datos en la gramática XML 2001
    http://infohost.nmt.edu/tcc/help/pubs/rnc/xsd.html

13. Desarrollo de Software Basado en Componentes

Jonás A. Montilva C., Nelson Arapé y Juan Andrés Colmenares
Facultades de Ingeniería de las Universidades de los Andes y de Zulia
http://webdelprofesor.ula.ve/ingenieria/jonas/Productos/Publicaciones/Congresos/CAC03%20Desarrollo%20de%20componentes.pdf

Desarrollo de Software basado en Componentes (CBSD)
http://msdn.microsoft.com/es-es/library/bb972268.aspx

14. Servicios Web

http://www.w3.org/standards/webofservices/
http://www.ibm.com/developerworks/ssa/webservices/newto/index.html

Tutoriales IBM
http://www.ibm.com/developerworks/views/webservices/library.jsp

Especificaciones de servicios Web
http://www.ibm.com/developerworks/offers/WS-Specworkshops/

15. Comités responsables de la arquitectura y reglamentación de los servicios Web

W3C
http://www.w3c.es/

OASIS
http://www.oasis-open.org/specs/

16. UDDI

http://uddi.xml.org/wiki
http://www.w3schools.com/wsdl/wsdl_uddi.asp
http://www.ibm.com/developerworks/webservices/library/ws-featuddi/

17. ODP

http://www.dmoz.org/World/Español/about.html

18. MHP y GEM

http://www.mhp.org/developers.htm
http://www.etsi.org/deliver/etsi_ts/102700_102799/102728/01.01.01_60/ts_102728v010101p.pdf
http://www.mhp.org/developers.htm#tutorials (Tutorial)

http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode (Códigos básicos de guía MHP)
http://es.wikipedia.org/wiki/Multimedia_Home_Platform#DVB-J (DVB-J)
http://en.wikipedia.org/wiki/DVB-HTML (DVB-HTML)
http://www.mhp.org/docs/tm3749.mug180.gem-iptv_white_paper.pdf (GEM-IPTV)

19. URI. Uniform Resource Identifier

http://es.wikipedia.org/wiki/Uniform_Resource_Identifier

20. Diferencia entre Marketing y Publicidad

http://la2daelegida.com.ar/marketing-vs-publicidad-para-principiantes/

21. Multi idioma

Google
http://groups.google.com/group/android-developers/web/localizing-android-apps-draft

Microsoft
http://apirola.wordpress.com/2008/08/19/multilingual-controls-in-c-easy/

TDT
http://www.loc.gov/standards/iso639-2/
http://www.loc.gov/standards/iso639-5/id.php
http://www.loc.gov/standards/iso639-5/langhome5.html

22. WSDL

http://www.w3.org/TR/wsdl (Estándar recomendado W3C)
http://www.w3schools.com/wsdl/default.asp (Introducción a la tecnología)
http://msdn.microsoft.com/es-es/library/ms175476.aspx (.NET)
http://es.wikipedia.org/wiki/WSDL#Estructura_del_WSDL (Estructura WSDL)
http://java.sun.com/webservices/technologies/management/ (Sun)

Crear WSDL (Tutoriales)
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=WSDL2Java#wsdlEclipse

http://www.hachisvertas.net/blog/01/2008/07/01/web-services-primeros-pasos-con-eclipse-generar-servicio-web

http://jcesarperez.blogsome.com/2007/07/24/desarrollar-un-webservice-desde-wsdl-con-axis2/

http://help.eclipse.org/help32/index.jsp?topic=/org.eclipse.wst.wsdl.ui.doc.user/tasks/tcrtwsdl.html

23. Android Market

http://www.androlib.com/appstats.aspx
http://es.wikipedia.org/wiki/Android_Market
http://es.wikipedia.org/wiki/Google_Checkout

24. APP STORE

http://appleweblog.com/2010/04/apple-cambia-las-condiciones-de-los-desarrolladores-un-duro-golpe-para-adobe

25. QoS

http://es.wikipedia.org/wiki/Calidad_de_Servicio

26. Open CCM

http://openccm.ow2.org/
http://openccm.ow2.org/doc/0.8/deploy_meta_inf.html

Versión 3.0 (Usada en trabajos anteriores)
http://openccm.ow2.org/doc/0.8/deploy_meta_inf.html#p_a_softpkg

Esquema SOFTPKG en notación DTD
http://openccm.ow2.org/dtd/ccm/softpkg.dtd

OMG CCM (Versión actual: 4.0)
http://www.omg.org/technology/documents/formal/components.htm
http://www.omg.org/technology/documents/corba_spec_catalog.htm#CCM

27. Licencias de software (tipos)

http://www.monografias.com/trabajos55/licencias-de-software/licencias-de-software.shtml http://es.wikipedia.org/wiki/Licencia_de_software

28. XML in MySQL

Using XML in MySQL 5.1 and 6.0
Jon Stephens works on the MySQL Documentation Team
http://dev.mysql.com/tech-resources/articles/xml-in-mysql5.1-6.0.html

MySQL
http://dev.mysql.com/doc/refman/5.5/en/index.html

Manejo XML
http://dev.mysql.com/doc/refman/5.5/en/xml-functions.html
http://dev.mysql.com/doc/refman/5.5/en/load-xml.html
http://dev.mysql.com/doc/refman/5.5/en/xml-functions.html
http://dev.mysql.com/doc/refman/5.5/en/mem-custom-data-item-xml-file.html

29. **Creación de archivos de partición en una BD**
    http://msdn.microsoft.com/es-es/library/ms190787(SQL.90).aspx

30. **Creación de colecciones de esquemas**
    http://technet.microsoft.com/es-es/library/ms187856(SQL.90).aspx] [33]

31. **Creación de los repositorios y asociaciones para particiones, colecciones de esquemas e índices.**
    http://msdn.microsoft.com/es-es/library/ms174979.aspx
    http://technet.microsoft.com/es-es/library/ms191497(SQL.90).aspx

32. **Linq**
    http://msdn.microsoft.com/en-us/library/system.xml.linq.aspx

ASP
http://msdn.microsoft.com/es-es/library/3hc29e2a.aspx
http://msdn.microsoft.com/es-es/library/ms178207.aspx
http://msdn.microsoft.com/es-es/library/7ytf5t7k.aspx

Otras formas de trabajar con XML: LINQ
http://msdn.microsoft.com/en-us/netframework/aa904594.aspx
http://msdn.microsoft.com/es-es/library/bb397676.aspx
http://msdn.microsoft.com/es-es/library/bb397678.aspx
http://msdn.microsoft.com/es-es/library/bb384065.aspx
http://msdn.microsoft.com/en-us/library/bb387098(v=vs.90).aspx

33. **Especificación OMG TRADER**
    http://www.omg.org/spec/TRADE/1.0/

34. **XML DOM**
    http://www.w3.org/DOM/

Correspondencia XML y estructuras de datos DataSet (DOM)
http://msdn.microsoft.com/es-es/library/84sxtbxh(VS.80).aspx

XML en ASP se trabaja en xDocument
http://msdn.microsoft.com/en-us/library/system.xml.linq.xdocument_members(v=VS.90).aspx

http://msdn.microsoft.com/es-es/library/bb340331.aspx

http://msdn.microsoft.com/es-es/library/system.xml.schema.xmlschemaset(v=VS.90).aspx

http://msdn.microsoft.com/es-es/library/system.xml.schema.xmlschemaset.validationeventhandler(v=VS.90).aspx

35. XQuery

http://msdn.microsoft.com/es-es/library/ms190798.aspx
http://msdn.microsoft.com/es-es/library/ms191474.aspx

36. La clase PROXY en .NET de Microsoft

http://msdn.microsoft.com/es-es/library/xy59yt45(v=VS.80).aspx
http://msdn.microsoft.com/es-es/library/d2s8y7bs.aspx