

Práctica 3

Descripción:

La práctica consiste en el desarrollo de un conjunto de servicios web con contratos *WSDL* como hicimos en la práctica 1, un cliente con .NET, 3 bases de datos y un servicio *WSDL* basado en la tecnología de orquestación de servicios *BPEL* el cual también hace uso de un contrato *WSDL*.

Servicios web:

- Un proyecto que representa el servicio del almacén llamado *EmpresaMTIS*.
- Un proyecto que representa el servicio a proveedores a consultar llamado *Proveedor*, en este caso hemos creados dos con los nombres de *ProveedorA* y *ProveedorB* para diferenciarlos.
- Un servicio de orquestación

Bases de datos:

- *empresamtis*: esta base de datos representa al almacén y contiene en una tabla llamada *productos*, con los datos de los productos como *id* (referencia), *nombre* (al final en desuso por falta de tiempo), *precio* (precio final para el comprador) y *cantidad* (número de unidades disponibles en el almacén).
- *proveedor1* y *proveedor2* (para *ProveedorA* y *ProveedorB* respectivamente): estas bases de datos dan soporte a los datos de proveedores con dos tablas, en una primera tenemos los productos, con su *id* (referencia), *nombre* (al final en desuso por falta de tiempo) y *precio* (cada proveedor tiene su precio para recibir solicitudes de compra). Por otro lado disponen de otra tabla llamada *compras*, que en un comienzo iba a ser usada junto a otra tabla llamada *empresas* y se iba a realizar un control de las empresas habían realizado solicitudes de stock identificando las por su cif, pero nuevamente por falta de tiempo son características añadidas no solicitadas de las que hemos tenido que prescindir por falta de tiempo.

Para el desarrollo haremos uso las siguientes herramientas:

- **XAMPP** como servidor de base de datos de *MySQL* consumible por el servidor.
- **Eclipse** como IDE de desarrollo de los servicios web y contratos *WSDL* además del servicio de orquestación con *BPEL*.
- **Visual Studio 2019 Enterprise** como *IDE* de desarrollo tanto del servidor como del cliente que consume dichos servicios haciendo uso de los contratos publicados por los mismos.

Servicios:

Hemos desarrollado los diferentes servicios con sus accesibles métodos (aunque el funcionamiento es el mismo, los contratos y su forma de trabajar con ellos difiere):

Logística: servicio que ofrece gestionar la logística de un almacén (saber si hay una cierta cantidad de un cierto producto y aumentar o reducir el stock de dicho producto).

- **comprobarStock:** proceso que informa de si hay suficientes unidades en el almacén para realizar la compra solicitada.
- **actualizarStock:** proceso encargado de modificar la cantidad de un producto en concreto que devuelve el almacén, aceptando valores negativos (una compra) o positivos (un encargo a un proveedor, incluyendo 0 que supondría una solicitud de 100 unidades como veremos más adelante).
- **actualizarPrecio:** proceso el cual modifica el precio para un producto en concreto, una característica que iba a ser añadida de forma automática a través del servicio BPEL tras comprar stock a los proveedores para ajustar automáticamente el precio en el caso de que uno de ellos lo modificara también.

ProveedorX: servicio que ofrece información sobre los productos de los que provee, así como la cantidad

- **presupuestar:** devuelve el coste que llevaría solicitar cierta cantidad de cierto número de productos (aunque se ha simplificado, se podría jugar con que a partir de ciertas unidades un proveedor hace rebaja y el otro no y ahí podría estar la diferencia a la hora de seleccionar uno u otro a través del servicio de orquestación).
- **solicitar:** este método devuelve si es posible para la empresa (antes se controlaba con el cif, averiguando si la empresa existía entre los datos de los proveedores) solicitar una cierta cantidad de un cierto producto, si es posible, si añade a la lista automáticamente la cantidad del producto que ha comprado y se informa a través de la misma respuesta.

AlmacenBPEL: servicio de orquestación con *BPEL* que se encarga de montar todo el proceso de compra, aceptando como entrada el número de unidades y la referencia al producto que se desea comprar

- **comprar:** devuelve una cadena de texto que informa del proceso, tanto de si la compra se pudo realizar como si no, e incluyendo mensajes de errores personalizados en casos concretos devueltos por los proveedores.

Puesta en marcha:

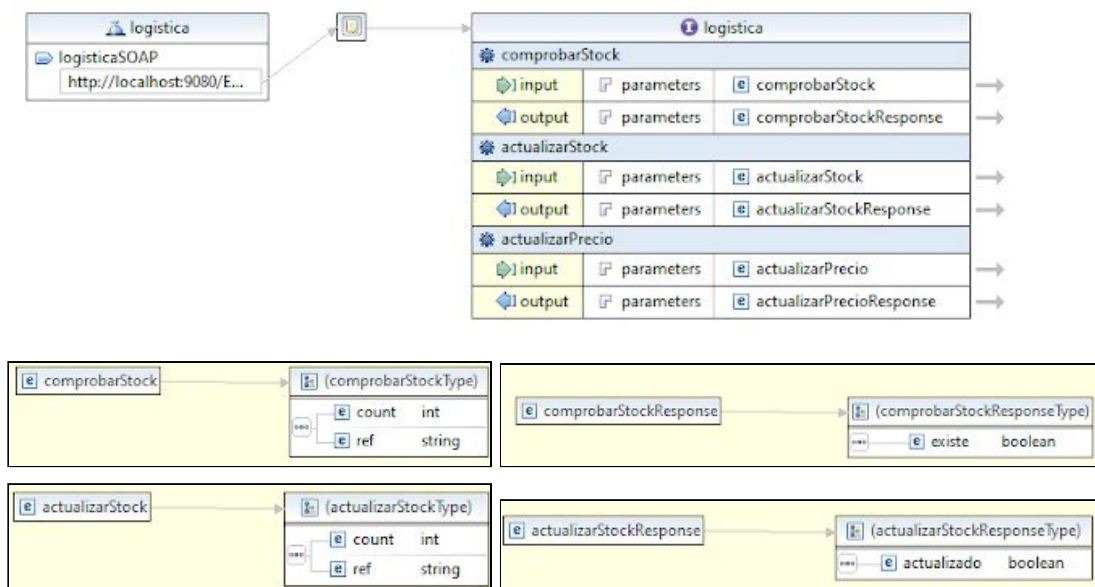
1. Iniciar *XAMPP*, iniciando servidor de *MySQL* con el puerto por defecto (3306).
2. Iniciar proyectos de servidor con *Eclipse*, tanto del tomcat (*Logistica*, *ProveedorA* y *ProveedorB*) como de orquestación (*AlmacenBPEL*).
3. Iniciar el cliente con Visual Studio 2019 Enterprise, y consumir el servicio web.

Demostración:

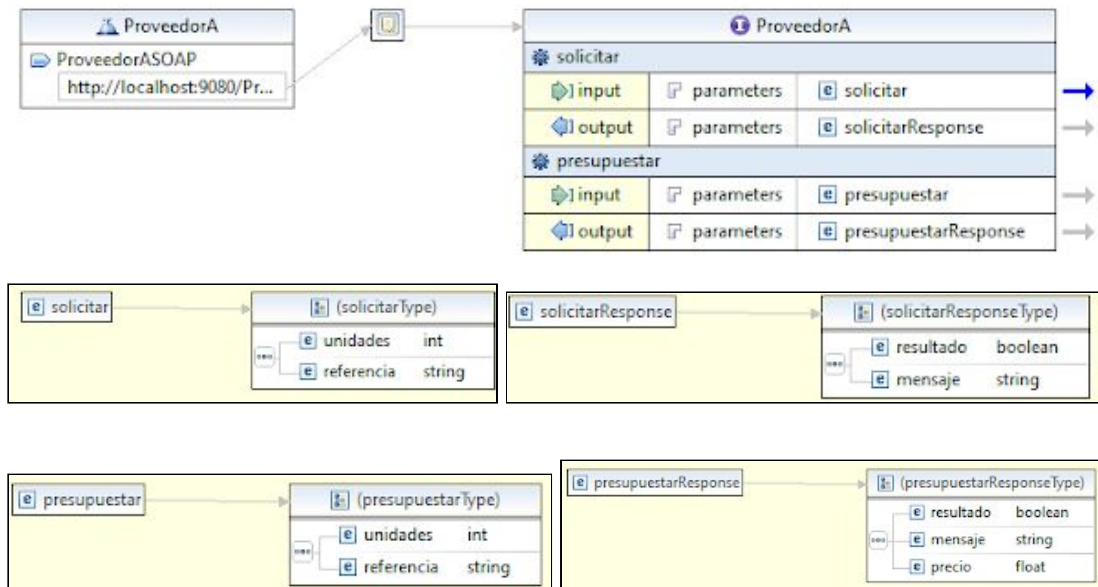
Para comenzar no vamos a analizar las bases de datos ya que queda sobreentendido que se explicaron previamente. A su vez, no entraremos en detalles insignificantes como la programación de los *Skeleton* o la capa de acceso a datos *DAO*.

Ahora vamos a proceder a analizar mediante capturas los archivos principales:

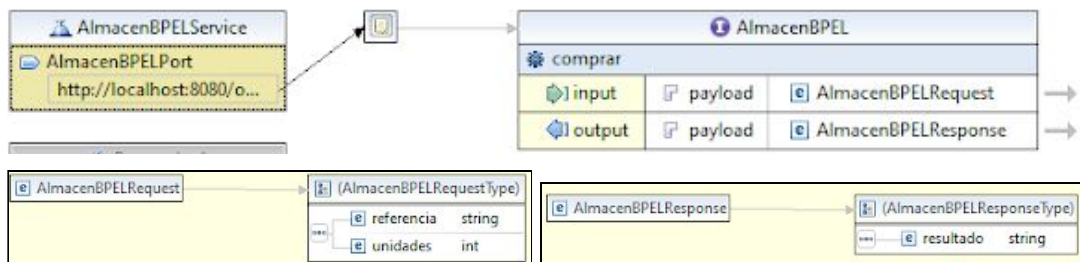
-EmpresaMTIS/logistica.wsdl



-ProveedorX/ProveedorX.wsdl (Por ejemplo ProveedorA)

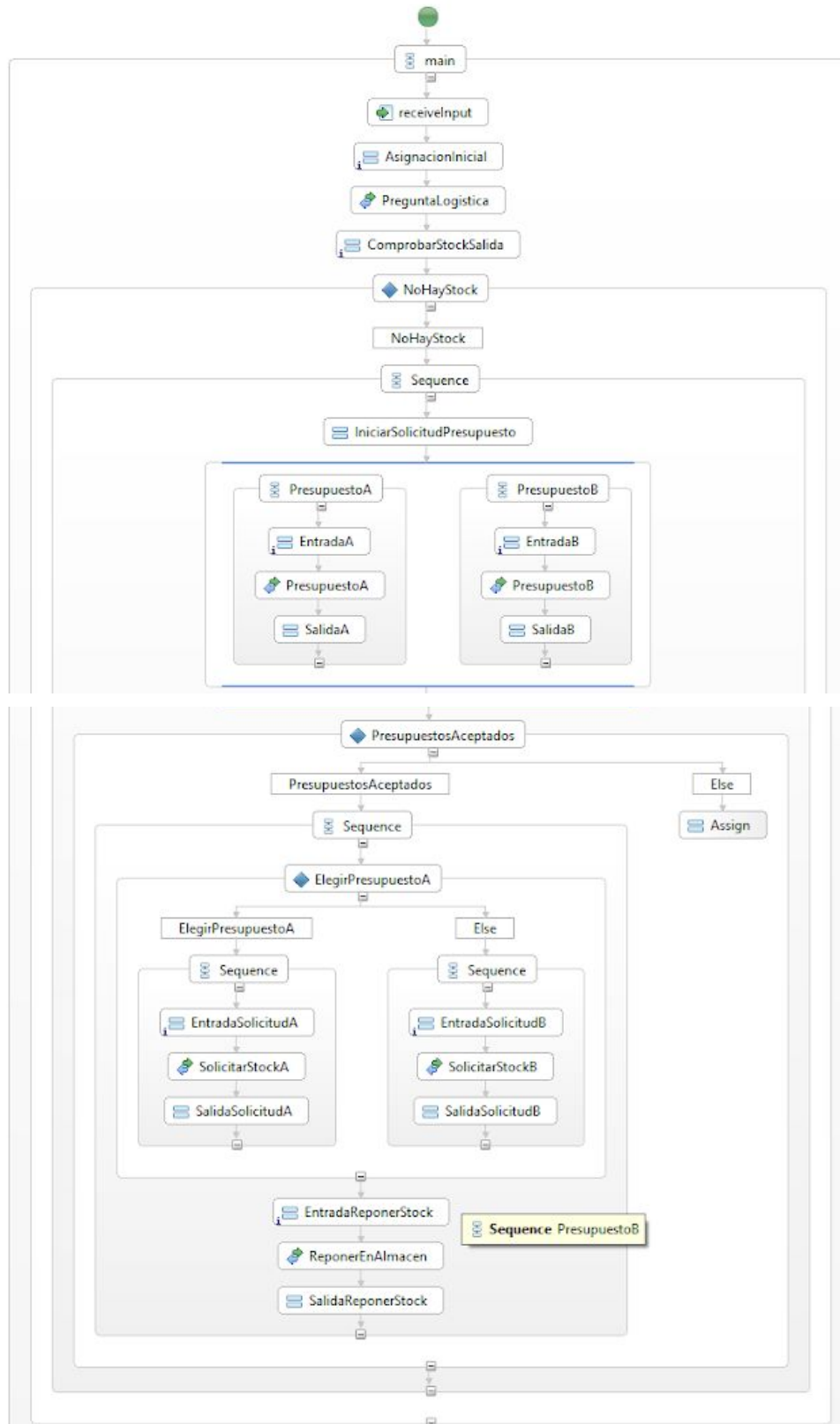


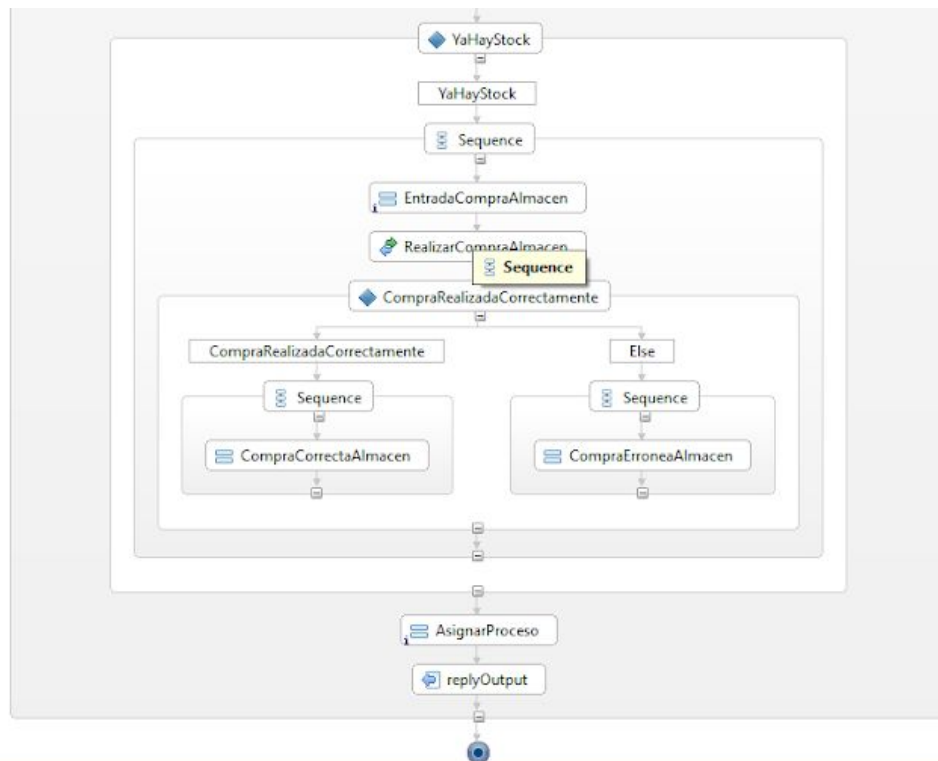
-PracticaAlmacen/bpelContent/AlmacenBPELArtifacts.wsdl



-PracticaAlmacen/bpelContent/AlmacenBPEL.bpel

Como se puede observar, hemos hecho uso de *Invoke*, *Assign*, *If* y *Flow*.





- **ClienteBPEL**, en este caso haremos capturas de los resultados logrados según varios contextos:

Compra normal directa al almacén:

The screenshot shows a window titled "Cliente Práctica 3 BPEL". The main text area displays the following messages:

[Sistema]: Solicitando 100 unidades del producto con referencia [prod1].

La compra al almacen se realizó correctamente.

At the bottom, there is a form with two input fields: "Producto" with the value "prod1" and "Cantidad" with the value "100". To the right of these fields is a "Comprar" button.

Compra con solicitud de stock a proveedores seleccionando el más económico:

Cliente Práctica 3 BPEL

[Sistema]: Solicitando 100 unidades del producto con referencia [prod2].

[Almacen]: Stock insuficiente, solicitando presupuesto de la cantidad solicitada(100) más 100, total: 200 unidades.

[ProveedorA]: Presupuesto obtenido con éxito: 900.0€

[ProveedorB]: Presupuesto obtenido con éxito: 1190.0€

[Sistema]: Se solicitó Stock a ProveedorA

[ProveedorA]: Stock ENVIADO correctamente

La compra al almacen se realizó correctamente.

Producto: Cantidad:

Compra de un producto que solamente existe en uno de los proveedores:

Cliente Práctica 3 BPEL

[Sistema]: Solicitando 100 unidades del producto con referencia [prod3].

[Almacen]: Stock insuficiente, solicitando presupuesto de la cantidad solicitada(100) más 100, total: 200 unidades.

[ProveedorB]: Presupuesto obtenido con éxito: 19000.0€

[ProveedorA]: El producto no existe

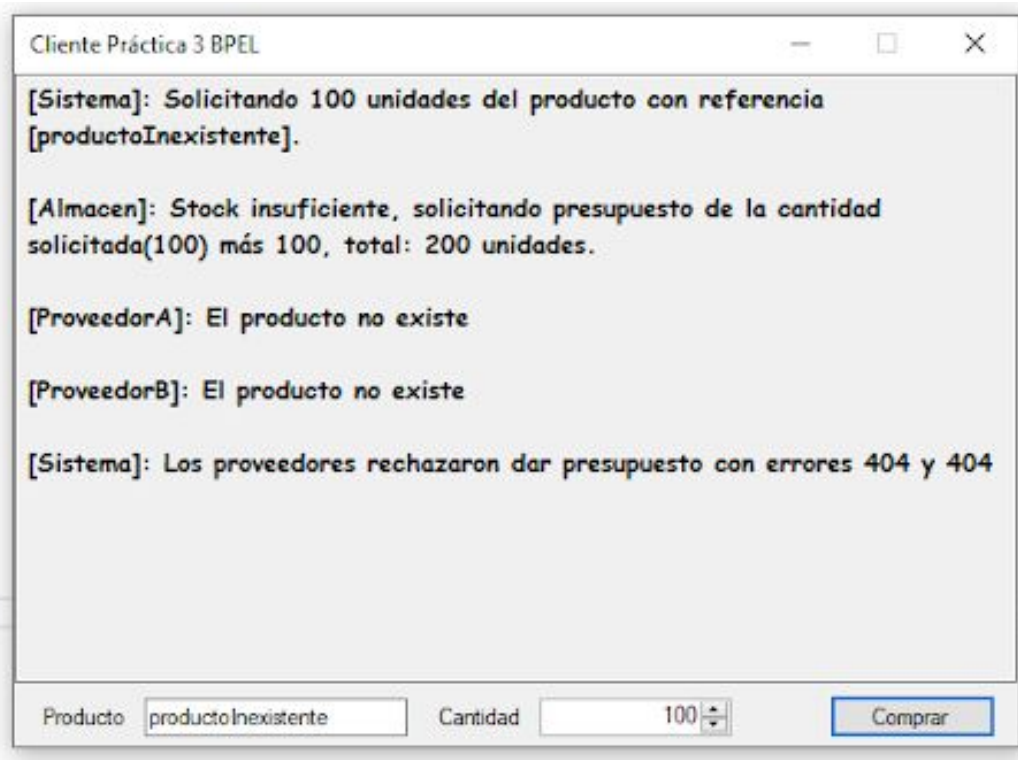
[Sistema]: Se solicitó Stock a ProveedorB

[ProveedorB]: Stock ENVIADO correctamente

La compra al almacen se realizó correctamente.

Producto: Cantidad:

Producto inexistente en ninguna base de datos:



Cliente Práctica 3 BPEL

[Sistema]: Solicitando 100 unidades del producto con referencia [productoInexistente].

[Almacen]: Stock insuficiente, solicitando presupuesto de la cantidad solicitada(100) más 100, total: 200 unidades.

[ProveedorA]: El producto no existe

[ProveedorB]: El producto no existe

[Sistema]: Los proveedores rechazaron dar presupuesto con errores 404 y 404

Producto Cantidad

Aspectos a destacar:

- Se ha facilitado un *String* de información tanto por parte de los proveedores como por parte del proceso orquestado en el cual se informa de como se realizó el proceso paso a paso.
- En un inicio, pensando que los servicios de proveedores no serían de la misma empresa, se pretendía realizar un control de que empresa realizaba que compra para dar mayor sensación de credibilidad entre otras características añadidas así como actualización automática del precio por parte del almacén o el uso del precio y nombre de los productos por parte del almacén, pero como además de no ser un servicio necesario ni requerido en la práctica he tenido una clara falta de tiempo he tenido que prescindir de estas características innecesarias.

Dificultades encontradas:

1. Complicaciones con los servicios *WSDL* a la hora de manejar el espacio de nombres o la *URL* ofrecida, pues se crea una copia del archivo *WSDL* en la carpeta '*Proyecto/Java Resources/Libraries/resources*' y si modificamos uno no se modifica el otro.
2. A través del proyecto de orquestación hay numerosos errores, desde que una variable no se inicializa bien hasta que la inicialización ofrecida por *Eclipse* es errónea.
3. Eclipse no solo falló con refactorizaciones, sino que el propio servicio de *ODE* y *TOMCAT* cambiaron su configuración sin tocar yo nada, obligandome a descargar nuevamente la carpeta *MTIS* y sobrescribir las carpetas de los *IDE* para que volviera todo a funcionar.
4. Eclipse fallaba en el proceso de crear de forma automática el archivo *XML*, lo que por un buen lado me ha dado la posibilidad de examinar gran cantidad de veces el archivo hasta entender que falla y porque, pero por otro lado ha sido una experiencia desagradable el tener reiteradamente errores por parte de Eclipse.

Conclusión: Ha sido "*una migraña*" y un proceso desagradable de crear numerosas veces proyectos tanto de servicios como el proyecto de orquestación y verificar que cada uno estuviera meticulosamente creado con sumo cuidado, ya que casi cualquier refactorización acababa echando a perder el proyecto creado.

Experiencia personal: aunque me parece muy interesante la orquestación de servicios a través de uno solo, y podría ser una forma muy eficaz de coordinar varios servicios, resulta asfixiante que una herramienta informática provoque semejante cantidad de errores. Cabe decir que he accedido al código en *XML* para editarlo mil veces para aprender que hace *Eclipse* al editar el archivo con extensión *bpel* y me parece increíble las cosas que me ha hecho.