

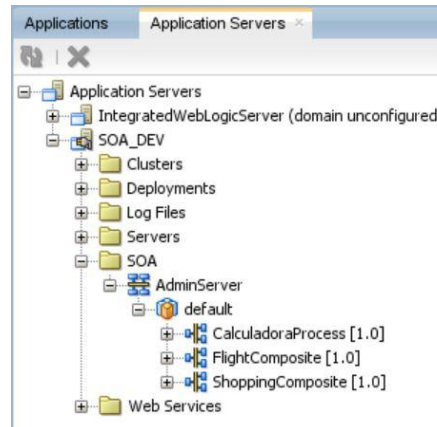
## Práctica 2: Orquestación de servicios web utilizando WS-BPEL

<b>0. Instrucciones de despliegue y pruebas.....</b>	<b>2</b>
<b>1. Reserva de vuelos (parte 1).....</b>	<b>3</b>
1.1. Estructura del compuesto SOA.....	3
1.2. Los procesos BPEL y cómo se relacionan los servicios.....	3
1.3. Casos de prueba: peticiones SOAP.....	4
1.3.1. EmpleadoProcess.....	4
1.3.2. IberiaProcess.....	5
1.3.3. VuelingProcess.....	7
1.3.4. GestorProcess.....	7
<b>2. Regateo de precios entre comprador y vendedor (parte 2).....</b>	<b>9</b>
2.1. Estructura del compuesto SOA.....	9
2.2. Los procesos BPEL y cómo se relacionan los servicios.....	9
2.3. Casos de prueba: peticiones SOAP.....	10
2.3.1. VerCantidad.....	10
2.3.2. Comprador.....	11
2.3.3. Vendedor.....	11
2.3.4. Gestor.....	12

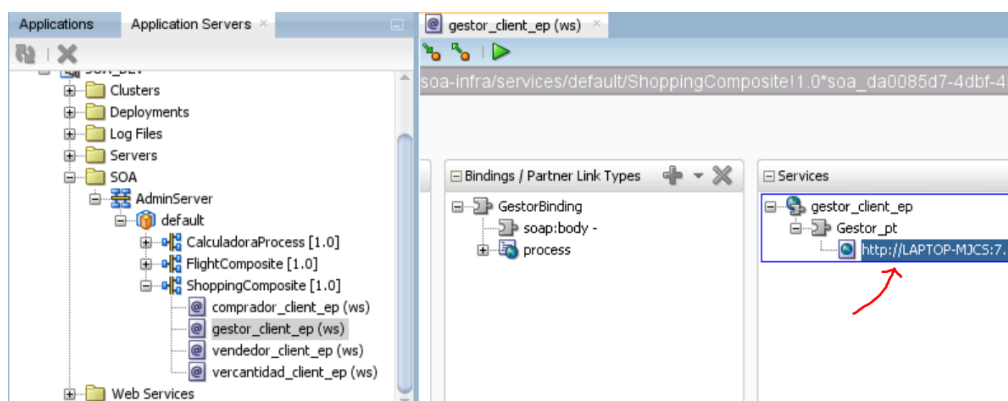
---

## 0. Instrucciones de despliegue y pruebas

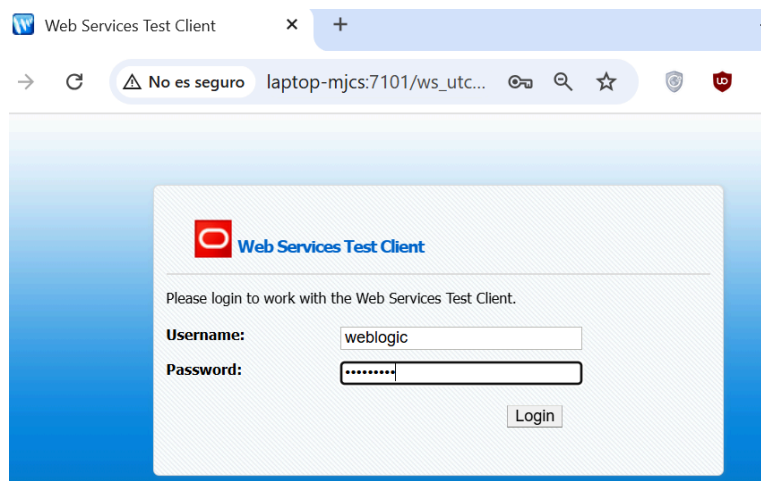
Como viene indicado en la *RELEASE\_NOTE*, este proyecto se despliega en el servidor SOA independiente que se ha estado usando hasta ahora.



Para realizar los tests (peticiones y respuestas SOAP), se puede acceder a dichos servicios desde una URL en el cliente WS de cada proceso.



Y después se introducen las credenciales correctas (contraseña: password1).

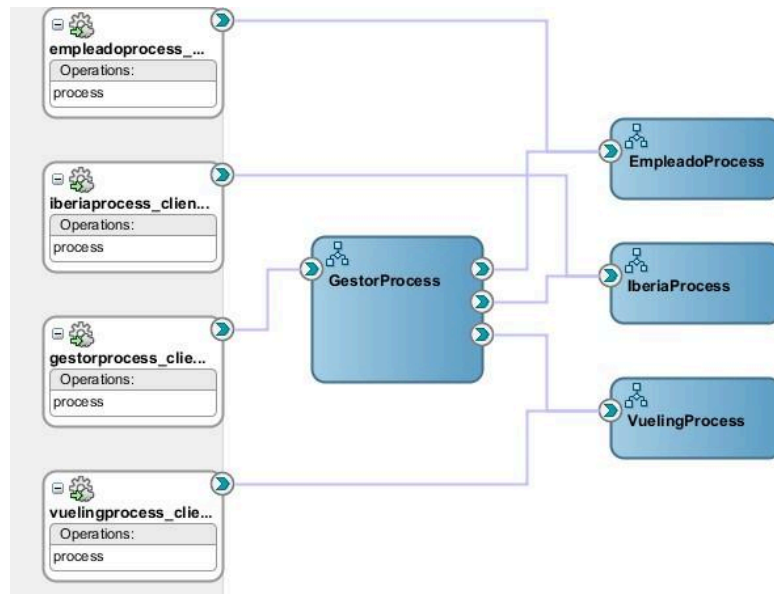


## 1. Reserva de vuelos (parte 1)

Se trata de orquestar la consulta de los precios de vuelo de dos aerolíneas para seleccionar el billete más económico y ofrecer un plan de viaje al cliente.

### 1.1. Estructura del compuesto SOA

El diagrama de *FlightComposite.xml* es el siguiente:



La estructura del compuesto SOA muestra una orquestación de procesos BPEL interconectados que colaboran para manejar la solicitud de reserva de vuelo de un cliente. Un proceso central actúa como gestor (*GestorProcess*), coordinando la interacción entre los servicios que manejan diferentes aspectos del negocio, como la consulta de precios a dos aerolíneas distintas (Iberia y Vueling).

### 1.2. Los procesos BPEL y cómo se relacionan los servicios

- **Empleado.bpel:** recibe el nombre del empleado y devuelve el estatus (o tipo de empleado), que determinará a su vez el tipo de billete (business o tourist).
- **Iberia.bpel:** simula la interacción con el sistema de reservas de Iberia.
  - Recibe los detalles del vuelo (tipo de empleado, fecha de inicio/salida, fecha de fin/regreso y tipo de aeropuerto/destino), comprobando que las fechas sean válidas; es decir, la fecha de regreso no puede ser anterior a la de ida.
  - Devuelve el precio calculado según el tipo de empleado/billete y de aeropuerto (AMS - Ámsterdam, FCO - Roma u otros países).
- **Vueling.bpel:** misma estructura y parámetros que los del proceso de Iberia, solo que cambian las tarifas (distintos multiplicadores), por lo que tendrán precios de vuelo diferentes.

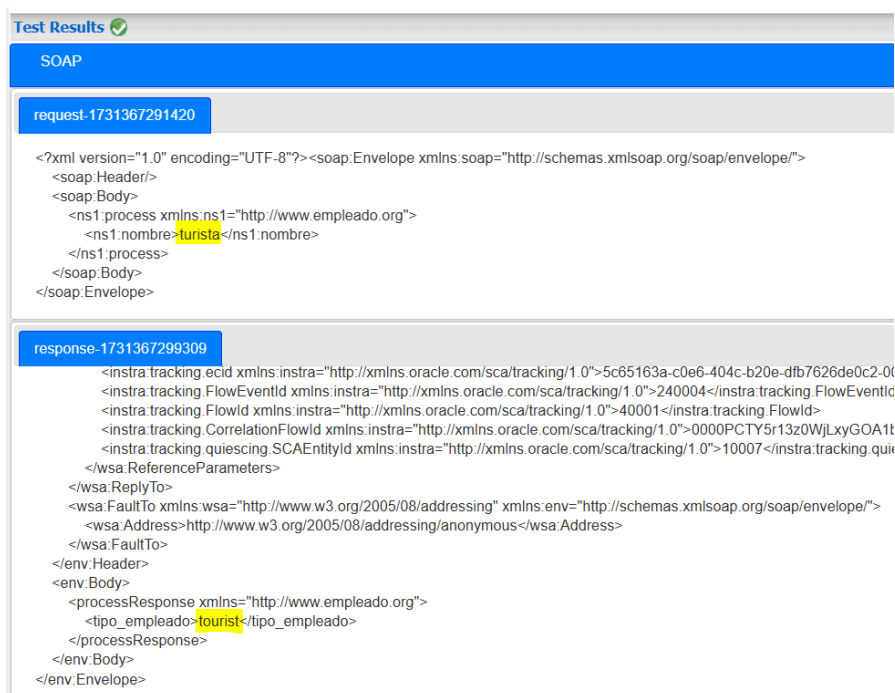
- **Gestor.bpel:** es el núcleo del compuesto, responsable de orquestar la interacción entre los distintos procesos.
  - Recibe los datos iniciales (nombre del empleado, fecha de inicio, fecha de fin y tipo de aeropuerto).
  - Invoca al proceso Empleado para obtener el tipo de empleado/billete, y a los procesos de las dos aerolíneas (Iberia y Vueling), para después comparar las ofertas de vuelo recibidas.
  - Finalmente, selecciona la opción más barata y devuelve al cliente los detalles de la reserva (nombre y tipo del empleado, compañía elegida y precio final).

### 1.3. Casos de prueba: peticiones SOAP

A continuación se detallan algunas solicitudes de prueba y las respuestas obtenidas para cada proceso testado.

#### 1.3.1. EmpleadoProcess

- Test 1: turista
  - Parámetro de entrada:
    - turista (string: nombre)
  - Respuesta obtenida:
    - tourist (string: tipo\_empleado)



The screenshot shows the 'Test Results' window for a SOAP test. It displays the request and response XML messages. The request is for the 'EmpleadoProcess' with the input 'turista'. The response returns 'tourist' as the employee type.

**Test Results** ✓

**SOAP**

request-1731367291420

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <ns1:process xmlns:ns1="http://www.empleado.org">
      <ns1:nombre>turista</ns1:nombre>
    </ns1:process>
  </soap:Body>
</soap:Envelope>
```

response-1731367299309

```
<instra:tracking.ecid xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">5c65163a-c0e6-404c-b20e-dfb7626de0c2-0</instra:tracking.FlowEventId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">240004</instra:tracking.FlowEventId>
<instra:tracking.FlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">40001</instra:tracking.FlowId>
<instra:tracking.CorrelationFlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">0000PCTY5r13z0WJLxyGOA1t</instra:tracking.CorrelationFlowId>
<instra:tracking.quiescing.SCAEntityId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">10007</instra:tracking.quiescing.SCAEntityId>
</wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
</wsa:FaultTo>
</env:Header>
<env:Body>
  <processResponse xmlns="http://www.empleado.org">
    <tipo_empleado>tourist</tipo_empleado>
  </processResponse>
</env:Body>
</env:Envelope>
```

- Test 2: business
  - Parámetro de entrada:
    - business (string: nombre)
  - Respuesta obtenida:
    - business (string: tipo\_empleado)

request-1731368019936

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <ns1:process xmlns:ns1="http://www.empleado.org">
      <ns1:nombre>business</ns1:nombre>
    </ns1:process>
  </soap:Body>
</soap:Envelope>
```

response-1731368020036

```
<instra:tracking.FlowEventId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">240009</instra:tracking.Flow
<instra:tracking.FlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">40002</instra:tracking.FlowId>
<instra:tracking.CorrelationFlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">0000PCT^qVg3z0WjLx
relationFlowId>
<instra:tracking.quiescing.SCAEntityId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">10007</instra:track
</wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:env="http://schemas.xmlsoap.org/soap/envelo
<wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
</wsa:FaultTo>
</env:Header>
<env:Body>
  <processResponse xmlns="http://www.empleado.org">
    <tipo_empleado>business</tipo_empleado>
  </processResponse>
</env:Body>
</env:Envelope>
```

### 1.3.2. IberiaProcess

#### ➤ Test 0: fecha inválida

- Parámetros de entrada:
  - empleado (string: tipo\_empleado)
  - 2024-04-05 (date: fecha\_inicio)
  - 2024-04-02 (date: fecha\_fin)
  - AMS (string: tipo\_aeropuerto)
- Respuesta obtenida:
  - Se lanza un error, cuyo faultcode es "invalidVariables", ya que la fecha de fin no puede ser anterior a la de inicio.

#### Test Results

SOAP

request-1731368402887

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <ns1:process xmlns:ns1="http://www.iberia.org">
      <ns1:tipo_empleado>empleado</ns1:tipo_empleado>
      <ns1:fecha_inicio>2024-04-05</ns1:fecha_inicio>
      <ns1:fecha_fin>2024-04-02</ns1:fecha_fin>
      <ns1:tipo_aeropuerto>AMS</ns1:tipo_aeropuerto>
    </ns1:process>
  </soap:Body>
</soap:Envelope>
```

response-1731368403835

```
<?xml version="1.0" encoding="UTF-8"?><env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header>
    <tracking:faultId xmlns:tracking="http://oracle.soa.tracking.core.TrackingProperty" xmlns:env="http://schemas.xmlsoap.org
  </env:Header>
  <env:Body>
    <env:Fault xmlns:ns0="http://docs.oasis-open.org/wsbpel/2.0/process/executable">
      <faultcode>ns0:invalidVariables</faultcode>
      <faultstring>faultName: {(http://docs.oasis-open.org/wsbpel/2.0/process/executable)invalidVariables}
    </env:Fault>
  </env:Body>
</env:Envelope>
```

➤ Test 1: business - AMS

- Parámetros de entrada:
  - business (string: tipo\_empleado)
  - 2024-04-02 (date: fecha\_inicio)
  - 2024-04-05 (date: fecha\_fin)
  - AMS (string: tipo\_aeropuerto)
- Respuesta obtenida:
  - 1250 (float: precio)

```

response-1731368852164
<instra:tracking.FlowEventId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">240021</instra:tracking.FlowEventId>
<instra:tracking.FlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">40004</instra:tracking.FlowId>
<instra:tracking.CorrelationFlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">0000PCTc1co3z0WjLxyGOA1bC
relationFlowId>
<instra:tracking.quiescing.SCAEntityId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">10008</instra:tracking.quies
</wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
</wsa:FaultTo>
</env:Header>
<env:Body>
  <processResponse xmlns="http://www.iberia.org">
    <precio>1250</precio>
  </processResponse>
</env:Body>
</env:Envelope>

```

➤ Test 2: turista - FCO

- Parámetros de entrada:
  - tourist (string: tipo\_empleado)
  - 2024-04-02 (date: fecha\_inicio)
  - 2024-04-05 (date: fecha\_fin)
  - FCO (string: tipo\_aeropuerto)
- Respuesta obtenida:
  - 100 (float: precio)

```

response-1731368976759
<instra:tracking.FlowEventId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">240026</instra:tracking.FlowEventId>
<instra:tracking.FlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">40005</instra:tracking.FlowId>
<instra:tracking.CorrelationFlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">0000PCTcW4Q3z0WjLxyGOA1bC
orrelationFlowId>
<instra:tracking.quiescing.SCAEntityId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">10008</instra:tracking.quiesci
</wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
</wsa:FaultTo>
</env:Header>
<env:Body>
  <processResponse xmlns="http://www.iberia.org">
    <precio>100</precio>
  </processResponse>
</env:Body>
</env:Envelope>

```

### 1.3.3. VuelingProcess

Se utilizan los mismos parámetros que en los tests de Iberia para facilitar la comparación de precios más adelante.

- Test 0: fecha inválida
  - Parámetros de entrada:
    - empleado (string: tipo\_empleado)
    - 2024-04-05 (date: fecha\_inicio)
    - 2024-04-02 (date: fecha\_fin)
    - AMS (string: tipo\_aeropuerto)
  - Respuesta obtenida:
    - Se lanza un error, cuyo faultcode es “invalidVariables”, ya que la fecha de fin no puede ser anterior a la de inicio.
- Test 1: business - AMS
  - Parámetros de entrada:
    - business (string: tipo\_empleado)
    - 2024-04-02 (date: fecha\_inicio)
    - 2024-04-05 (date: fecha\_fin)
    - AMS (string: tipo\_aeropuerto)
  - Respuesta obtenida:
    - 400 (float: precio)
- Test 2: turista - FCO
  - Parámetros de entrada:
    - tourist (string: tipo\_empleado)
    - 2024-04-02 (date: fecha\_inicio)
    - 2024-04-05 (date: fecha\_fin)
    - FCO (string: tipo\_aeropuerto)
  - Respuesta obtenida:
    - 200 (float: precio)

### 1.3.4. GestorProcess

- Test 1: Vueling mejor precio
  - Parámetros de entrada:
    - business (string: nombre)
    - 2024-04-02 (date: fecha\_inicio)
    - 2024-04-05 (date: fecha\_fin)
    - AMS (string: tipo\_aeropuerto)
  - Respuesta obtenida:
    - business (string: nombre)
    - business (string: tipo\_empleado)
    - vueling (string: company)
    - 400 (float: precio)

response-1731369853998

```

relationFlowId>
  <instra:tracking.quiescing.SCAEntityId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">10006</instra:tracking.qu
  </wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
</wsa:FaultTo>
</env:Header>
<env:Body>
  <processResponse xmlns="http://www.gestor.org">
    <nombre>business</nombre>
    <tipo_empleado>business</tipo_empleado>
    <company>vueling</company>
    <precio>400</precio>
  </processResponse>
</env:Body>
</env:Envelope>

```

➤ Test 2: Iberia mejor precio

- Parámetros de entrada:
  - turista (string: nombre)
  - 2024-04-02 (date: fecha\_inicio)
  - 2024-04-05 (date: fecha\_fin)
  - FCO (string: tipo\_aeropuerto)
- Respuesta obtenida:
  - turista (string: nombre)
  - tourist (string: tipo\_empleado)
  - iberia (string: company)
  - 100 (float: precio)

response-1731370076873

```

relationFlowId>
  <instra:tracking.quiescing.SCAEntityId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">10006</instra:tracking.q
  </wsa:ReferenceParameters>
</wsa:ReplyTo>
<wsa:FaultTo xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
</wsa:FaultTo>
</env:Header>
<env:Body>
  <processResponse xmlns="http://www.gestor.org">
    <nombre>turista</nombre>
    <tipo_empleado>tourist</tipo_empleado>
    <company>iberia</company>
    <precio>100</precio>
  </processResponse>
</env:Body>
</env:Envelope>

```

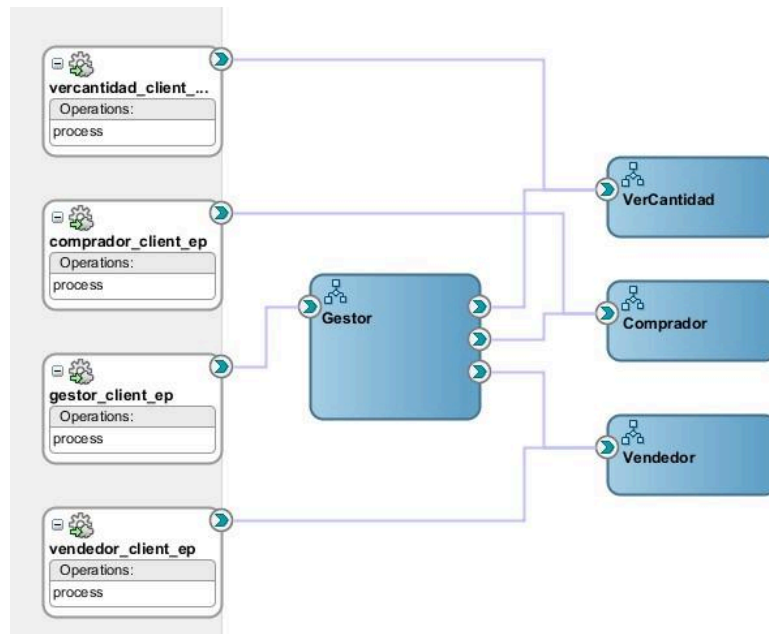


## 2. Regateo de precios entre comprador y vendedor (parte 2)

Consiste en orquestar la actividad de comercialización (regateo de precios) entre un comprador y un vendedor de un producto solicitado.

### 2.1. Estructura del compuesto SOA

El diagrama de *ShoppingComposite.xml* es el siguiente:



Sigue una estructura similar al de la primera parte de la práctica, donde se tiene un proceso central denominado *Gestor* que coordina la interacción entre varios procesos especializados: *VerCantidad*, *Comprador* y *Vendedor*. Cada uno de estos procesos maneja aspectos específicos del flujo de negocio, mientras que el gestor actúa como intermediario, orquestando las solicitudes y respuestas entre los clientes y los procesos internos.

### 2.2. Los procesos BPEL y cómo se relacionan los servicios

- **VerCantidad.bpel:** verifica la disponibilidad de productos en stock. Recibe el nombre del producto, consulta la cantidad disponible y devuelve un mensaje con el estado de disponibilidad (boolean) y la cantidad en caso de que el producto exista.
- **Comprador.bpel:** gestiona las ofertas de precio realizadas por el comprador. Recibe un precio original y una oferta, y evalúa si la oferta es aceptable, comparándola con un porcentaje del 70% del precio original. Devuelve un indicador de aceptación o rechazo (boolean) de la oferta.
- **Vendedor.bpel:** proporciona contraofertas en base a las ofertas del comprador. Calcula un nuevo precio con un descuento del 10% sobre la oferta recibida y lo devuelve.

- **Gestor.bpel:** actúa como el coordinador principal, integrando la lógica de negocio y orquestando la interacción entre los procesos anteriores.
  - Recibe la solicitud inicial del cliente (el nombre del producto).
  - Verifica la disponibilidad del producto, invocando al proceso *VerCantidad*.
  - Partiendo de una oferta inicial, que es igual al precio original, inicia la negociación, representada por un bucle *while* que se ejecuta mientras el indicador de aceptación del comprador sea *false*.
  - Se procede con el regateo de precios entre el comprador y vendedor (el vendedor va generando contraofertas hasta que el comprador la acepta), y así hasta llegar a un acuerdo o finalizar la negociación.
  - Por último:
    - En caso de que no esté disponible dicho producto, devuelve el nombre del producto y un mensaje informando de que no se ha podido encontrar.
    - En cualquier otro caso, devuelve el nombre del producto, el precio original y final (tras el regateo), y un mensaje de confirmación de la compra.

## 2.3. Casos de prueba: peticiones SOAP

### 2.3.1. VerCantidad

- Test 1: producto no válido
  - Parámetros de entrada:
    - INVALID\_PROD (string: producto)
  - Respuesta obtenida:
    - false (boolean: disponibilidad)

```

</env:Header>
<env:Body>
  <processResponse xmlns="http://xmlns.oracle.com/ShoppingService/ShoppingServiceProject/VerCantidad">
    <cantidad/>
    <disponibilidad>false</disponibilidad>
  </processResponse>
</env:Body>
</env:Envelope>

```

- Test 2: producto disponible
  - Parámetros de entrada:
    - CAMISA (string: producto)
  - Respuesta obtenida:
    - 30 (int: cantidad)
    - true (boolean: disponibilidad)

```

<env:Body>
  <processResponse xmlns="http://xmlns.oracle.com/ShoppingService/ShoppingServiceProject/VerCantidad">
    <cantidad>30</cantidad>
    <disponibilidad>true</disponibilidad>
  </processResponse>
</env:Body>
</env:Envelope>

```

### 2.3.2. Comprador

➤ Test 1: oferta no aceptada

- Parámetros de entrada:
  - 100 (float: precio\_original)
  - 85 (float: precio\_ofrecido)
- Respuesta obtenida:
  - false (boolean: acceptable)

```
<env:Body>
  <processResponse xmlns="http://xmlns.oracle.com/ShoppingService/ShoppingServiceProject/Comprador">
    <acceptable>false</acceptable>
  </processResponse>
</env:Body>
</env:Envelope>
```

➤ Test 1: oferta aceptada

- Parámetros de entrada:
  - 100 (float: precio\_original)
  - 63 (float: precio\_ofrecido)
- Respuesta obtenida:
  - true (boolean: acceptable)

```
</env:Header>
<env:Body>
  <processResponse xmlns="http://xmlns.oracle.com/ShoppingService/ShoppingServiceProject/Comprador">
    <acceptable>true</acceptable>
  </processResponse>
</env:Body>
</env:Envelope>
```

### 2.3.3. Vendedor

➤ Test 1: contraoferta

- Parámetros de entrada:
  - 100 (float: oferta)
- Respuesta obtenida:
  - 90 (float: contraoferta)

```
<env:Body>
  <processResponse xmlns="http://xmlns.oracle.com/ShoppingService/ShoppingServiceProject/Vendedor">
    <contraoferta>90</contraoferta>
  </processResponse>
</env:Body>
</env:Envelope>
```

### 2.3.4. Gestor

➤ Test 1: producto no válido

- Parámetros de entrada:
  - INVALID\_PROD (string: producto)
- Respuesta obtenida:
  - INVALID\_PROD (string: producto)
  - No se ha encontrado el producto (string: mensaje)

```
<processResponse xmlns="http://xmlns.oracle.com/ShoppingService/ShoppingServiceProject/Gestor">
  <producto>INVALID_PROD</producto>
  <precio_original/>
  <precio_final/>
  <mensaje>No se ha encontrado el producto</mensaje>
</processResponse>
</env:Body>
</env:Envelope>
```

➤ Test 2: regatear precio camisa

- Parámetros de entrada:
  - CAMISA (string: producto)
- Respuesta obtenida:
  - CAMISA (string: producto)
  - 100 (float: precio\_original)
  - 65,61 (float: precio\_final)
  - La compra fue realizada con éxito (string: mensaje)

```
<env:Body>
  <processResponse xmlns="http://xmlns.oracle.com/ShoppingService/ShoppingServiceProject/Gestor">
    <producto>CAMISA</producto>
    <precio_original>100</precio_original>
    <precio_final>65.61000000000001</precio_final>
    <mensaje>La compra fue realizada con éxito</mensaje>
  </processResponse>
</env:Body>
</env:Envelope>
```