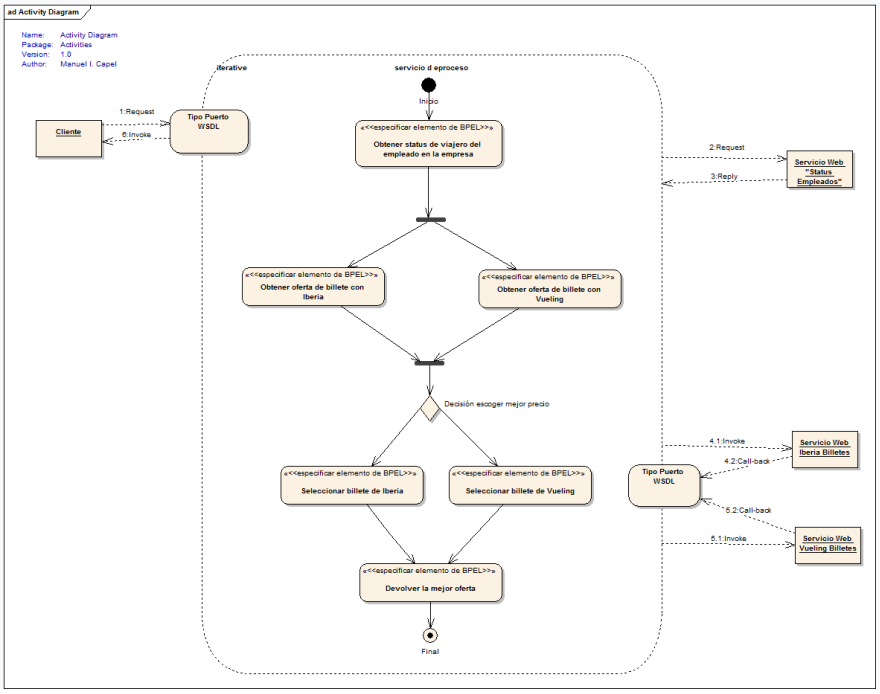
Práctica 2: "Orquestación de servicios Web utilizando WS-BPEL"

1. Reserva de vuelos

Enunciado: La aplicación cliente llama al proceso de negocio BPEL e indica el nombre del empleado, el destino del viaje y las fechas (fecha de salida y regreso). El proceso BPEL comprueba en primer lugar el estatus del empleado para comprar los billetes de avión en la clase (turista, business, jet privado) según el nivel del empleado en la empresa); suponemos la existencia de un WS que proporciona esta última información. Posteriormente, el proceso BPEL comprobará los precios de los billetes con 2 aerolíneas: Iberia y Vueling. Volvemos a suponer que ambas compañías aéreas proporcionan un WS para realizar estas consultas. El proceso BPEL debe ser capaz de seleccionar el precio más barato y devolver un plan de viaje viable a la aplicación cliente.



Tiempo total: 4 horas

Requisitos previos:

1. JDeveloper 12c instalado y configurado.
2. Configuración de Oracle SOA Suite (SOA Composite, BPEL y WebLogic Server).
3. Conocimientos básicos de procesos BPEL y servicios WSDL.

Parte 1. Crear una nueva aplicación SOA Crear una nueva aplicación SOA (15 minutos)

1. Abrir JDeveloper:
    - Iniciar JDeveloper y crear una nueva aplicación SOA:
        - File → New → Applications → Aplicación SOA.
        - Nombrar la aplicación (por ejemplo, FlightServicesProject).
2. Crear un nuevo compuesto SOA:
    - Después de crear la aplicación, crear un nuevo Proyecto SOA dentro de la aplicación:

- Seleccionar Composite con BPEL.
- Nombre del composite (por ejemplo, FlightComposite)

Parte 2: Añadir archivos WSDL para servicios (15 minutos)

3. Crear archivos WSDL:
    - Para cada servicio (Gestor, Empleado, Iberia, Vueling), crear un archivo WSDL que defina las operaciones y los mensajes de entrada/salida.
    - En JDeveloper, hacer clic con el botón derecho en el proyecto → New → WSDL Document.
    - Definir las operaciones necesarias:
        - Gestor puede tener operaciones como getEmployeeDetails.
        - Empleado podría procesar la lógica de tipo empleado.
        - Iberia y Vueling podrían manejar la lógica de precios de los vuelos.

      Ejemplo de operaciones WSDL:
        - Empleado.wsdl → Operación: proceso, Entrada: EmpleadoRequestMessage, Output: EmpleadoResponseMessage.
        - Iberia.wsdl y Vueling.wsdl → Operación: calcularPrecio, Entrada: FlightRequestMessage, Output: PriceResponseMessage.

Parte 3: Crear el proceso Empleado.bpel (30 minutos)

4. Añadir un Proceso BPEL para Empleado:
    - Hacer clic con el botón derecho del ratón en el composite → New → BPEL Process.
    - Nombrar el proceso Empleado y seleccionar un Synchronous BPEL Process (ya que esperamos una respuesta).
5. Definir variables de entrada/salida:
    - Asignar los mensajes de entrada y salida del archivo WSDL a variables BPEL.
    - Por ejemplo, crear inputVariable para EmpleadoRequestMessage y outputVariable para EmpleadoResponseMessage.
6. Implantar la lógica de negocio:
    - Utilizar una actividad <receive> para recibir la solicitud del servicio Empleado.
    - Utilizar una condición <if> para decidir si el tipo de empleado debe ser de primera clase o turista en función del nombre del empleado.
    - Utilizar una actividad <assign> para establecer el tipo_empleado en el mensaje de salida.
    - Terminar con una actividad <reply> para devolver la respuesta.

Parte 4: Crear el proceso Iberia.bpel (30 minutos)

7. Añadir el Proceso BPEL de Iberia:
    - Haga clic con el botón derecho del ratón en el composite → New → BPEL Process.
    - Nómbrarlo Iberia, seleccionar Synchronous BPEL Process.
8. Definir variables:
    - Crear inputVariable para solicitudes de vuelo (por ejemplo, FlightRequestMessage).
    - Crear outputVariable para respuestas de precios (por ejemplo, PriceResponseMessage).
9. Aplicar la lógica de precios:
    - Añadir una actividad <receive> para recibir los detalles del vuelo.
    - Utilizar una condición <if> para comprobar si las fechas de vuelo son válidas (fin >= inicio). Si no son válidas, lanza un fallo utilizando una actividad <throw>.

- Calcular el precio en función del tipo_empleado (por ejemplo, si es de primera clase, multiplicar por 25; en caso contrario, por otra cantidad).
- Asignar el precio calculado a la outputVariable.
- Añadir una actividad <reply> para devolver el precio.

Parte 5: Aplicación de precios específicos para aeropuertos (30 minutos)

10. Añadir más lógica a Iberia.bpel:
    - Utilizar condiciones <if> adicionales para ajustar el precio en función de tipo_aeropuerto:
        - Por ejemplo, si es AMS (Amsterdam), multiplicar por 25.
        - Si es FCO (Roma), multiplicar por 15.
        - Para otros aeropuertos, multiplicar por 5.
11. Finalizar y desplegar:
    - Después de implementar la lógica, desplegar el proceso BPEL y probarlo utilizando peticiones SOAP para asegurarse de que el cálculo del precio funciona.

Parte 6: Desarrollar el proceso Vueling.bpel (30 minutos)

12. Añada el proceso Vueling BPEL:
    - Haga clic con el botón derecho del ratón en el composite → New → BPEL Process.
    - Nómbrarlo Vueling, seleccionar Synchronous BPEL Process.
13. Implantar la lógica empresarial:
    - Utilizar una lógica similar a Iberia.bpel, pero ajustar los cálculos si es necesario (por ejemplo, diferentes multiplicadores de precio o reglas de negocio).
    - Asignar el precio calculado a la outputVariable y devolver el resultado utilizando una actividad <reply>.

Parte 7: Conectar los componentes (30 minutos)

14. Servicios de orquestación:
    - En el compuesto, combinar el componente Gestor para llamar a los servicios Empleado, Iberia y Vueling.
    - Hacer clic con el botón derecho del ratón en el componente → Service Binding y asegurarse que todos los componentes están correctamente conectados.
    - Crear conexiones entre los servicios para que puedan interactuar.
15. Implementar y probar:
    - Desplegar el compuesto actualizado en WebLogic (se recomienda desplegar en SOA_DEV).
    - Utilizar SOAP-UI o el analizador HTTP integrado de JDeveloper para enviar peticiones de prueba y verificar que los servicios funcionan de extremo a extremo (por ejemplo, el servicio Gestor invoca a Empleado y a Iberia/Vueling para obtener los datos de los empleados y calcular los precios de los vuelos).

Parte 8: Pruebas exhaustivas (30 minutos)

16. Pruebas de extremo a extremo:
    - Probar todo el composite, empezando por Gestor y pasando por Empleado, Iberia y Vueling.
    - Crear diferentes casos de prueba:
        - Tipos de empleados de prueba (primera clase, turista).
        - Probar de fechas de vuelo válidas y no válidas.

        - Probar diferentes tipos de aeropuertos (AMS, FCO, otros).
17. Depurar cualquier problema:
    - Utilice los registros de WebLogic o de SOA_DEV y las herramientas de depuración de JDeveloper para resolver cualquier error o problema en la lógica del proceso, las asignaciones de variables o el cableado.

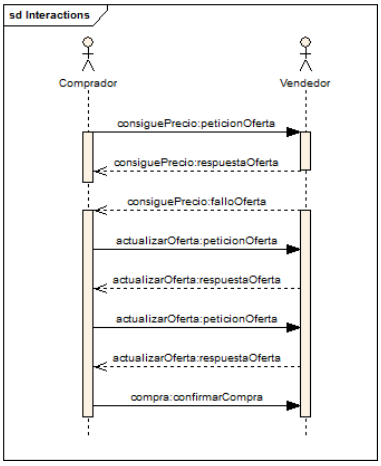
Entregables:

1. Un compuesto SOA totalmente funcional (SOA Composite Application):
    - Composite.xml: Define los servicios, referencias y componentes en el compuesto SOA.
    - Procesos BPEL:
        - *Empleado.bpel*: Determina el tipo de empleado (primera clase o turista) en función de la entrada.
        - *Iberia.bpel*: Gestiona la tarificación de vuelos en función del tipo de empleado y aeropuerto.
        - *Vueling.bpel*: Similar a Iberia.bpel con reglas de tarificación ajustadas.
    - Archivos WSDL: Definiciones de servicios web para cada servicio (*Gestor*, *Empleado*, *Iberia*, *Vueling*).
2. Casos de prueba:
    - Archivos de solicitud y respuesta SOAP: Ejemplos de casos de prueba que muestran cómo interactuar con los servicios, cubriendo diferentes escenarios para cada servicio.
3. Instrucciones de despliegue:
    - Unas instrucciones que expliquen cómo desplegar los procesos SOA composite y BPEL en Oracle WebLogic Server o SOA_DEV.
4. Crear documentación:
    - Redactar un breve documento explicativo:
        - La estructura del compuesto SOA.
        - Los roles de cada proceso BPEL (Empleado, Iberia, Vueling).
        - Cómo se conectan los servicios entre sí.
        - Cómo probar los servicios (ejemplos de peticiones SOAP y resultados esperados).

Práctica 2: "Orquestación de servicios Web utilizando WS-BPEL"

2. Regateo de precios entre comprador y vendedor

Orquestar la actividad de comercialización (regateo de precios) entre un comprador y un vendedor de un producto solicitado, según el diagrama de interacción que se muestra en la figura siguiente:



Enunciado: El comprador comienza preguntando un precio al vendedor y éste responde dándole el precio del producto o planteando una excepción si no conoce el producto solicitado o no está disponible en ese momento en las existencias del almacén. El comprador sigue pidiendo un precio al vendedor y entra en un comportamiento repetitivo (con actualizaciones del precio del artículo) hasta que el comprador decide aceptar un precio ofrecido por el vendedor, que considera el mejor precio posible para el producto. En esta tarea, se nos pide que desarrollemos la descripción completa de la orquestación anteriormente descrita, que tiene lugar entre el comprador y el vendedor del producto".

Parte 1: Configurar el entorno (10 minutos)

1. Abrir JDeveloper:
    - Asegurarse que JDeveloper 12c está instalado con Oracle SOA Suite y WebLogic Server configurados.
    - Abrir JDeveloper y cree una nueva aplicación SOA:
        - File → New → Applications → Aplicación SOA.
        - Nombrar la aplicación (por ejemplo, ShoppingServiceProject).
2. Crear un nuevo compuesto SOA:
    - Después de crear la aplicación, cree un nuevo Proyecto SOA dentro de la aplicación:
        - Seleccionar Composite con BPEL.
        - Nombre el compuesto (por ejemplo, ShoppingComposite).
    - Asegurarse que el composite tiene activada la funcionalidad BPEL.

Parte 2: Crear el VerCantidad.bpel (Proceso de verificación de existencias) (20 minutos)

3. Añadir un nuevo proceso BPEL:
    - Haga clic con el botón derecho del ratón en el el composite → New → BPEL Process.
    - Nombrar el proceso *VerCantidad* y seleccionar Synchronous BPEL Process (ya que devolverá la disponibilidad y cantidad de stock).
4. Definir variables de entrada/salida:

- Definir mensajes de entrada y salida (VerCantidadRequestMessage, VerCantidadResponseMessage) utilizando un WSDL simple. Estos deben llevar el nombre del producto y devolver la cantidad de existencias y el estado de disponibilidad.

5. Implementar la lógica de negocio (comprobación de existencias):
   - Utilizar <receive> para obtener el nombre del producto.
   - Implementar condiciones <if> para comprobar los nombres de los productos (por ejemplo, CAMISA, TRAJE, ZAPATOS).
     - Si el producto existe, fijar la cantidad y el flag de disponibilidad (existencia = true).
     - En caso contrario, establecer existencia = false.
   - Utilizar una actividad <assign> para asignar cantidades de productos y devolver la respuesta.
6. Desplegar y probar el proceso *VerCantidad*:
   - Desplegar el proceso en WebLogic o SOA_DEV y probarlo con peticiones SOAP sencillas (por ejemplo, envíe nombres de productos y verifique la respuesta de disponibilidad de existencias).

Parte 3: Crear el Comprador.bpel (20 minutos)

7. Añadir un nuevo proceso BPEL:
   - Añada otro Proceso BPEL llamado Comprador.
   - Será un Synchronous BPEL Process para gestionar las ofertas de precio de los compradores.
8. Definir variables de entrada/salida:
   - Definir variables de entrada (CompradorRequestMessage) para recibir precio_orig y ofrecido.
   - Definir variables de salida (CompradorResponseMessage) para devolver si la oferta es aceptable (flag aceptable).
9. Aplicar la lógica de comparación de precios:
   - Utilizar una actividad <recibir> para obtener la oferta del comprador.
   - Utilizar una actividad <if> para comparar el precio ofrecido frente al 70% del precio original (precio_orig * 0,7).
   - Si la oferta es aceptable, asignar la variable de salida aceptable = true; en caso contrario, false.
   - Terminar con una actividad <reply> para devolver el resultado al comprador.
10. Desplegar y Probar el Proceso Comprador:
    - Desplegar el proceso y pruébelo utilizando peticiones SOAP con diferentes entradas de precios.

Parte 4: Crear el Vendedor.bpel (20 minutos)

11. Añadir el Proceso BPEL Vendedor:
    - Hacer clic con el botón derecho del ratón en el composite → New → BPEL Process.
    - Nómbrelo Vendedor y seleccionar Synchronous BPEL Process.
12. Definir variables de entrada/salida:
    - Definir variables de entrada (VendedorRequestMessage) para aceptar la oferta del comprador.
    - Definir variables de salida (VendedorResponseMessage) para devolver la contraoferta.
13. Aplicar la lógica de la contraoferta:
    - Utilizar una actividad <assign> para multiplicar la oferta del comprador por 0,9 (un 10% de descuento) y devolver el nuevo precio.

- Utilizar una actividad <reply> para enviar la contraoferta de vuelta al comprador.

14. Implementar y probar:
    - Desplegar y probar el proceso Vendedor con varias ofertas de precios.

Parte 5: Desarrollar el Gestor.bpel para Coordinar Servicios (25 minutos)

15. Crear el proceso Gestor.bpel:
    - Añadir otro Proceso BPEL llamado Gestor, que actuará como coordinador principal.
16. Definir variables y entradas:
    - Configurar variables para interactuar con los servicios VerCantidad, Comprador y Vendedor.
17. Implementar el flujo de negocio:
    - Recibir solicitud de producto:
        - Utilizar una actividad <receive> para aceptar el nombre del producto del cliente.
        - Llamar al proceso VerCantidad utilizando una actividad <invoke> para comprobar si el producto está disponible.
    - Compruebe la disponibilidad del producto:
        - Si el producto está disponible (existencia = true), iniciar la negociación del precio con el proceso Comprador.
        - Si el producto no está disponible, devolver un mensaje indicando que el producto no se encuentra.
    - Negociación de precios:
        - Enviar el precio original al proceso Comprador y comprobar si la oferta del comprador es aceptable.
        - Si la oferta del comprador es inaceptable, invocar el proceso Vendedor para obtener una contraoferta y continuar la negociación.
        - Repetir el bucle (<while>) hasta que el comprador acepte el precio o fracase la negociación.
    - Finalizar la transacción:
        - Una vez que el comprador acepta el precio, devolver un mensaje de éxito y finalizar la compra.
18. Desplegar y probar el proceso Gestor:
    - Desplegar el composite completo y probar de principio a fin interactuando con el proceso Gestor.
    - Utilizar solicitudes SOAP para enviar nombres de productos y ofertas de precios, asegurándose de que el control del proceso fluye a través de la comprobación de existencias, la negociación del comprador y las contraofertas del vendedor.

Parte 6: Pruebas finales y depuración (15 minutos)

19. Pruebas exhaustivas:
    - Pruebe todos los servicios juntos enviando peticiones SOAP al servicio Gestor. Compruébalo:
        - Se comprueba la disponibilidad de existencias.
        - La negociación del precio se produce correctamente con ofertas aceptables/inaceptables.
        - La contraoferta la genera el vendedor cuando la necesita.
        - El mensaje final y el resultado de la transacción se devuelven correctamente.

20. Depuración:
    - Utilice los registros de WebLogic y las herramientas de depuración de JDeveloper para identificar y resolver cualquier problema en el flujo del proceso o en la asignación de variables.

Entregables:

1. Aplicación compuesta SOA:
    - Composite.xml que conecta todos los procesos BPEL.
2. Procesos BPEL:
    - VerCantidad.bpel: Proceso de verificación de existencias.
    - Comprador.bpel: Evaluación de la oferta del comprador.
    - Vendedor.bpel: Proceso de contraoferta del vendedor.
    - Gestor.bpel: Coordinador de procesos para la negociación y finalización de transacciones.
3. Archivos WSDL:
    - Ficheros WSDL para cada servicio (VerCantidad, Comprador, Vendedor, Gestor).
4. Casos de prueba:
    - Archivos de solicitud/respuesta SOAP para probar cada proceso.
5. Instrucciones de despliegue:
    - Guía para desplegar el composite y ejecutar las pruebas.
6. Cree documentación:
    - Resumen: Resumen de los procesos SOA composite y BPEL (VerCantidad, Comprador, Vendedor, Gestor).
    - Descripciones de procesos: Breve explicación de cada proceso BPEL y su función en la aplicación.
    - WSDL: Descripción de interfaces de servicio.
    - Instrucciones de prueba: Pasos para probar cada proceso utilizando peticiones SOAP.
    - Guía de despliegue: Instrucciones para desplegar el proyecto en WebLogic o SOA_DEV.