Obligatorio 2: Web Services y ESB

Requerimientos

La empresa TicketInco es una empresa reconocida en el mercado de organización de espectáculos y eventos. Desde su sitio web, es posible consultar la disponibilidad de entradas para un espectáculo, realizar una reserva, realizar el pago y posterior emisión de entradas. Este negocio le da muchos resultados a TicketInco pero hace unos meses tomó la decisión de ampliar sus fronteras de negocios, permitiendo que sus partners sean capaces de vender las entradas de los espectáculos que promociona. Para alcanzar este objetivo, comenzó un proceso de migración de su proceso de venta y emisión de entradas actual basada en una arquitectura JEE a una arquitectura SOA, basada en web services, que le permita una comunicación estandarizada con ellos. Es así, que se diseñaron fachadas de Web Services sobre las funcionalidades Java existentes, que permiten reutilizar gran parte de sus sistemas.

Por otro lado, TicketInco planea ampliar sus posibilidades de medios de pago pasando de un único medio de pagos actual propio, a utilizar medios de pago provistos por terceros. En ese sentido, el primer partner de negocio será PagosYa!, el cual disponibilizó un servicio Rest sincrónico con este propósito.

El sistema a desarrollar deberá implementar el nuevo proceso de negocios basado en la composición de Web Services SOAP. La mayoría de los Web Services son provistos por la aplicación JEE sobre la cual se implementó una fachada de Web Services, salvo el sistema de medios de pago, el cual es un antiguo sistema basado en Cobol que permite únicamente una integración basada en colas de mensajes JMS. Este sistema cuenta con una única cola de mensajes para recibir los pagos de las ventas de entrada. Se asumen que todos los pagos son procesados, por lo que no hay respuesta a esta solicitud. El procesamiento de errores queda por fuera del alcance de este obligatorio.

La empresa tomó la decisión estratégica de implementar la nueva arquitectura SOA utilizando únicamente Web Services SOAP, con el objetivo de estandarizar las tecnologías de integración y que la adaptación a tecnología sea más rápida y económica, dada la antigüedad de sus sistemas. Migrar a una arquitectura SOA basada en servicios Rest, implicaría una actualización tecnológica de los sistemas existentes que no está en los planes estratégicos de la empresa. Para mitigar la heterogeneidad tecnológica entre los Web Services SOAP y los medios de pago (tanto local como tercero), TicketInco planea reutilizar el ESB adquirido hace unos años para ofrecer una fachada de Web Services SOAP sobre estos, permitiendo que la composición de servicios esté basada únicamente en Web Services SOAP. De esta forma, la composición de servicios siempre se comunica con Web Services SOAP.

En el anexo 1 y 2 se presenta una descripción gráfica del proceso de negocio a implementar.

Se pide:

1. Desarrollo de Web Services sobre el sistema Java para:
    a) consulta de entradas disponibles
    b) reserva de entradas
    c) consulta de estado de reserva
    d) confirmación de reserva de venta de entradas
    e) anulación de venta de entradas
2. Diseño e implementación de una API Rest sincrónica del partner de negocios PagosYa! para la confirmación y anulación de pagos con el medio de pago externo, siguiendo los lineamientos vistos en el curso.
3. Desarrollo de 2 fachadas de Web Services SOAP utilizando el ESB para los siguientes sistemas:
    a) API rest PagosYa! (medio de pago externo) para confirmación y anulación de pagos
    b) Medio de pago basado en mensajería para confirmación y anulación de pagos propio a TicketInco.
4. Desarrollo del Web Service de "Callback" para la recepción de confirmación de pagos y emisión de entradas por parte de los partners.
5. Utilizar SOAPUI como aplicación cliente de Web Services.
6. Para todas las APIs a desarrollar, se debe cumplir con los contratos especificados en el anexo 3.

Requerimientos no funcionales

1. Los Web Services SOAP deberán implementarse con las librerías JAX-WS. Como servidor de aplicaciones se podrá usar Tomcat o Wildfly. Los Web Services deberán loggear todos los pedidos procesados en consola o en un archivo. En caso de utilizar Tomcat, se debe utilizar Apache CXF como tecnología de Web Services.
2. Los Web Services REST deberán implementarse usando las librerías JAX-RS. Como servidor de aplicaciones se podrá usar Tomcat o Wildfly. Los Web Services deberán loggear todos los pedidos procesados en consola o en un archivo.
3. Se debe usar MuleESB como producto de tipo ESB.
4. El Web Service de reserva de entradas (1.b) requiere que los mensajes vengan firmados utilizando WS-Security. Los grupos deberán mostrar evidencia en la defensa que se está validando la firma de estos mensajes.
5. El Web Service de confirmación de reserva de venta de entradas (1.d) requiere que los mensajes vengan cifrados con WS-Security. Los grupos deberán mostrar evidencia en la defensa que se están cifrando los mensajes. Por ejemplo, logging del mensaje soap.
6. El Web Service de confirmación de reserva de venta de entradas (1.d) es asincrónico y se debe utilizar WS-Addressing para la implementación de respuestas desacopladas.
7. Las entradas confirmadas (respuestas del Web Service confirmación de pago (pto 5)) deberán enviarse utilizando MTOM.
8. La comunicación con el servicio REST debe utilizar httpbasic y ssl. Se deben usar los repositorios de usuarios de Tomcat o Wildfly.
9. Se debe usar ActiveMQ como servidor de colas de mensajes
10. Se debe utilizar Java 7 o superior para todas las implementaciones
11. Se debe utilizar SOAPUI como cliente de la composición de servicios
12. Debe ser posible tener más de un proceso de venta de entradas en ejecución

13. En caso de usar una base de datos para almacenar el estado de la composición, se debe utilizar PostgreSQL

Sugerencia: para la implementación del timer se sugiere usar Quartz o Spring scheduler

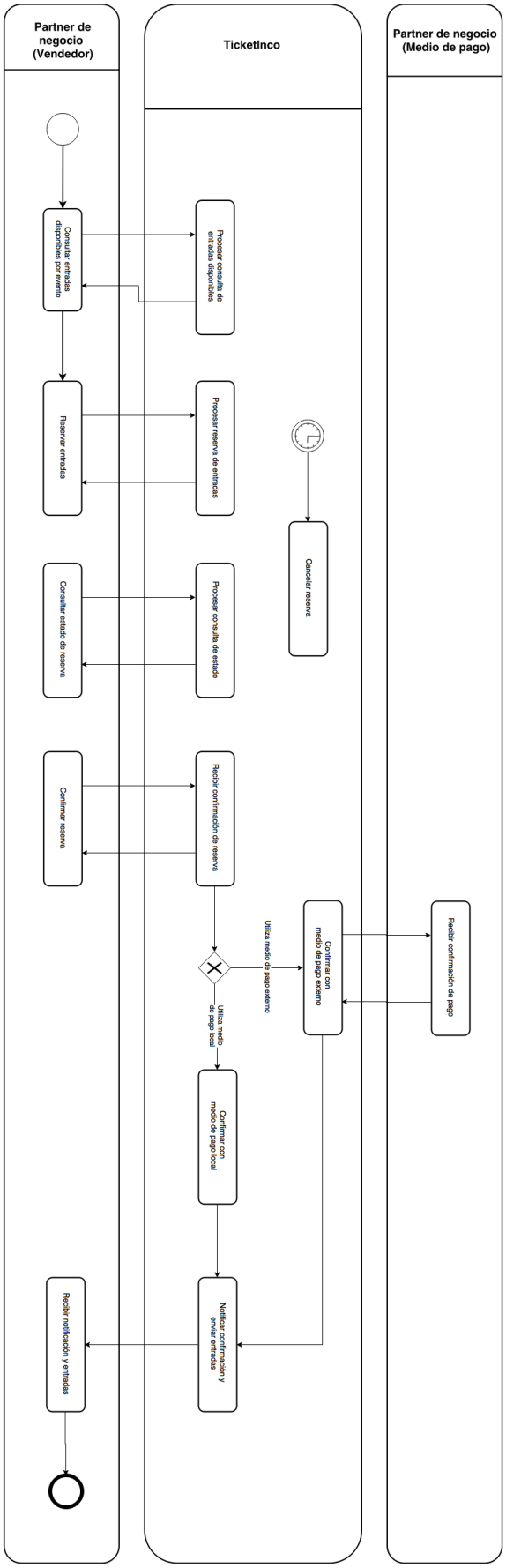
Entregables:

1. Código fuente de todos los sistemas desarrollados
2. Documentación describiendo la arquitectura del sistema.

Fecha de entrega: 13 de noviembre hasta las 23:59hrs

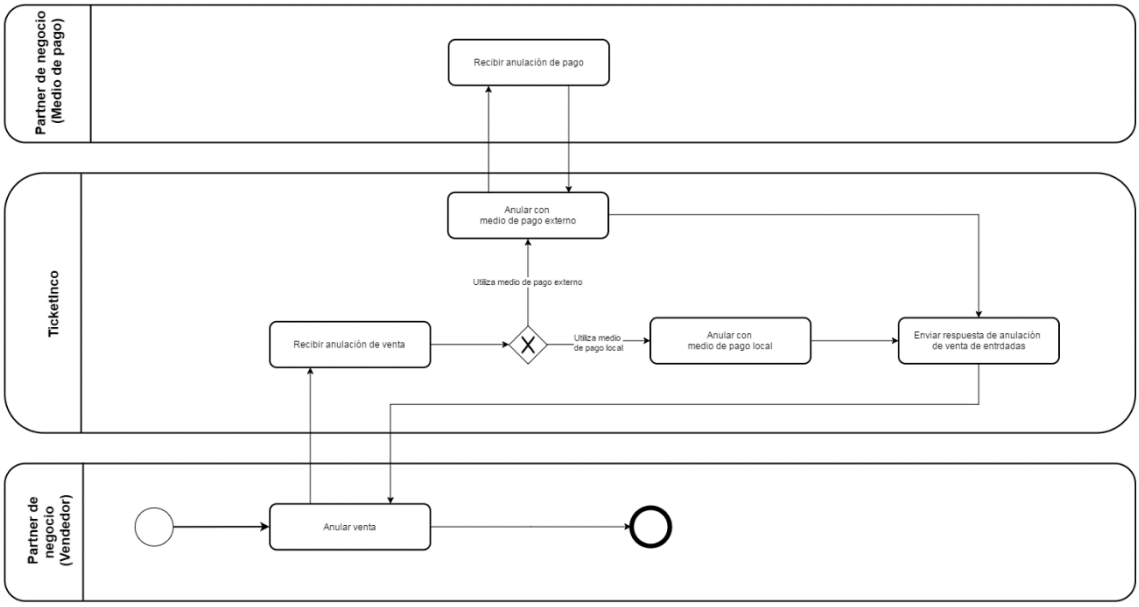
Defensa: Las fechas de las defensas serán el 14/11, 16/11 y 21/11 en el horario del curso.

Anexo 1: Proceso de negocio de reserva y venta de entradas



Anexo 2: Proceso de negocio de anulación de venta de entradas

Luego de realizada la venta y emisión de ventas de entradas, el cliente tendrá la posibilidad de anular la venta y devolver las entradas. Este proceso deberá anular los pagos, marcar la venta como anulada y marcar las entradas vendidas como disponibles nuevamente. En casos que el medio de pago sea local, se debe enviar un mensaje a la cola de mensajes de este sistema y en casos donde el medio de pago sea externo, se debe invocar al servicio rest de anulación de pagos correspondiente.



Anexo 3: Formatos y estructuras de datos

Web Service para consulta de entradas disponibles por evento		
Descripción	Dado un evento y una fecha, retorna una lista de horarios, donde cada horario posee una lista de disponibilidades. Una disponibilidad está compuesta de un sector, precio y cantidad de asientos disponibles.	
Datos de entrada	Tipo de datos	Descripción
Identificador del evento	Integer	N/A
Fecha del evento	Date	N/A
Datos de salida	Tipo de datos	Descripción
Horario	Datetime	N/A
Disponibilidad		
Sector	String	El nombre del sector disponible
Precio	Double	El precio de la entrada en el sector asociado
Cantidad	Integer	La cantidad de entradas disponibles en el sector asociado.

Web Service para reserva de entradas		
Descripción	Dado un evento, una fecha y un conjunto de disponibilidades, retorna un identificador indicando que se realizó correctamente la reserva.	
Características especiales	Utiliza WS-Security para firmar los mensajes.	
Datos de entrada	Tipo de datos	Descripción
Identificador del evento	Integer	N/A
Fecha del evento	Date	N/A
Lista de horarios*		Posee la misma estructura que el resultado de la consulta de entradas disponibles
Datos de salida	Tipo de datos	Descripción
Identificador de reserva	Long	

Web Service para consulta de estado de reserva		
Descripción	Dado un id de reserva se retorna el estado de la misma.	
Datos de entrada	Tipo de datos	Descripción
Identificador de la reserva	Long	N/A
Datos de salida	Tipo de datos	Descripción
Estado	Integer	0 = cancelado 1 = pendiente 2 = confirmado
Errores	No existe la reserva	

Web Service para confirmación de una reserva		
Descripción	Dado un id de reserva, un identificador de medio de pago y la información de pago, se lanza el proceso de confirmación de venta de entradas.	
Características especiales	Utiliza WS-Addressing para proveer asincronismo. Las respuestas de este Web Service son enviadas al Web Service de notificación de confirmaciones. Utiliza cifrado con WS-Security para proveer confidencialidad en la comunicación.	
Datos de entrada	Tipo de datos	Descripción
Identificador de la reserva	Long	N/A
Identificador de medio de pago	Long	N/A
Número de tarjeta	String	
Fecha de vencimiento	Date	
Dígito verificador	Int	
Errores	No existe la reserva	

Web Service para notificación de confirmaciones de una reserva y entradas		
Descripción	Dado un id de reserva y una lista de entradas, se procesa la recepción de las mismas.	
Características especiales	Utiliza MTOM para recibir las entradas en formato binario.	
Datos de entrada	Tipo de datos	Descripción
Identificador de la reserva	Long	N/A
Lista de entradas		
Entrada	Binario	Imagen de la entrada

Web Service para anulación de una reserva		
Descripción	Dado un id de confirmación de pago y un id de medio de pago, se anula la venta y se marcan como disponibles las entradas vendidas.	
Características especiales	Este servicio es de tipo Request-Response	
Datos de entrada	Tipo de datos	Descripción
Identificador de la confirmación	Long	N/A
Identificador de medio de pago	Long	N/A
Datos de salida	Tipo de datos	Descripción
Identificador de confirmación de la anulación	Integer	Identificador asociado a la anulación.
Errores	No existe el pago con ese identificador	
API REST para recepción de confirmación de pagos		
Descripción	Dada la información de la tarjeta y un monto, retorna si se acepta o no el pago.	
Datos de entrada	Tipo de datos	Descripción
Número de tarjeta	Long	N/A
Fecha de vencimiento	Datetime	N/A
Dígito verificador	Integer	N/A
Monto	Double	N/A

Datos de salida	Tipo de datos	Descripción
Id confirmación de pago	Long	

API REST para anulación de pagos		
Descripción	Dado un id de confirmación de pago, se retorna si se acepta o no la anulación del pago.	
Datos de entrada	Tipo de datos	Descripción
Id confirmación de pago	Long	Id confirmación de pago
Datos de salida	Tipo de datos	Descripción
Id confirmación de anulación	Long	

Formato para la comunicación con el medio de pago local		
Datos de entrada	Tipo de datos	Descripción
Número de tarjeta	String	Cadena de caracteres de 16 dígitos
Fecha de vencimiento	String	Formato dd-mm-yyy hh:mm
Dígito verificador	String	Formato NNN
Monto	String	Formato NNNN.NN

Formato para la comunicación con el medio de pago local para anulación de pagos		
Datos de entrada	Tipo de datos	Descripción
Id confirmación de pago	String	Id confirmación de pago

Referencias

1. Mule ESB: https://docs.mulesoft.com/mule-fundamentals/v/3.7/first-30-minutes-with-mule
2. Mule ESB download: https://developer.mulesoft.com/download-mule-esb-runtime
3. Publish SOAP API: https://docs.mulesoft.com/mule-user-guide/v/3.7/publishing-a-soap-api
4. Wildfly: http://wildfly.org/
5. SOAPUI: http://www.soapui.org/
6. Keytool: https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html
7. CXF-WSSecurity: http://cxf.apache.org/docs/ws-security.html
8. JAX-RS: http://docs.oracle.com/javaee/6/tutorial/doc/giepu.html
9. Quartz: http://www.quartz-scheduler.org/
10. Spring Scheduling: https://spring.io/guides/gs/scheduling-tasks/