# Obligatorio 2: Web Services y ESB

**Requerimientos**

La mutualista ProSalud comenzó el proceso de desarrollo de una nueva aplicación móvil con el objetivo de proveerle mejores servicios a sus afiliados. La nueva aplicación tiene como principal servicio la gestión de la agenda médica y consulta de medicación de los afiliados, incluyendo la consulta, alta y baja de cita médica.

El proceso de alta de cita médica comienza con la selección de la especialidad a solicitar donde luego de seleccionarla, se le despliegan al usuario los médicos disponibles para esa especialidad. Una vez seleccionado el médico, se le presentan al usuario las fechas y horarios disponibles, para luego confirmar la selección de médico, fecha y hora. El afiliado también tiene la posibilidad de seleccionar una agenda rápida donde luego de elegir especialista, selecciona agenda rápida y se le retorna una lista con las primeras 4 opciones disponibles más inmediatas donde está definido médico, fecha y hora. Inmediatamente de presentadas estas opciones, el afiliado puede confirmar una de ellas. En todo momento el afiliado puede consultar sus citas agendadas o dar de baja una cita.

Para llevar a cabo el proceso de agenda, la aplicación móvil se comunica con el sistema de Agenda vía un conjunto de APIs Restful que ofrece y que mantiene la información de disponibilidad de los médicos. La lista de especialidades, lista de médicos por especialidad y consulta de medicación se mantienen en otro sistema (SysDoc) y se ofrecen mediante tres Web Services SOAP. En ese sentido, ProSalud definió utilizar su ESB para transformar estos Web Services en APIs Restful y así ofrecer una interfaz uniforme a la aplicación. A su vez, definió aplicar el patrón Gateway en su ESB y todas las APIs Restful ofrecidas a la aplicación móvil deben estar publicadas en él y requieren de un token de seguridad emitido por un servidor OAuth2. Por lo tanto, la aplicación móvil primero necesita solicitar un token al servidor OAuth2 para luego consumir las APIs Restful a través del ESB, quien valida la presencia del token. Se utiliza la capacidad de caché del ESB para la consulta de especialidades y no tener que invocar cada vez al Web Service de especialidades.

Para la atención de los pacientes, los médicos utilizan el sistema SysDoc que les permite registrar los datos del diagnóstico, el cual puede incluir la emisión de una o varias recetas de medicación y la realización de estudios médicos. En caso de que el diagnóstico incluya la realización de estudios médicos, el sistema SysDoc debe comunicarse con el sistema de CliniDoc para registrar el alta del estudio. Dado que el registro del estudio puede llevar varios minutos se utiliza WS-Addressing para gestionar el pedido asincrónicamente. CliniDoc responde asincrónicamente a SysDoc el registro del estudio con un identificador del estudio. Los médicos utilizan este identificador para consultar por el resultado, donde un Web Service con MTOM permite obtener los resultados para los estudios que incluyan documentos adjuntos (ej: imagenología). La medicación indicada por el médico debe poder visualizarse en la aplicación móvil.

Por último, el sistema de facturación de ProSalud es un viejo ERP que solo permite una comunicación con archivos, por lo que se utiliza el ESB para leer los archivos de contabilidad y enviar la información contable a los Web Services de DGI que requieren WS-Security.

**Recomendación:** Se recomienda una vez desarrollados los Web Services con WS-Security, consumirlos primero con SOAPUI previo el desarrollo del cliente, con el objetivo de validar su configuración/desarrollo.

**Se pide:**

1. Diseño de las APIs Restful requeridas por la aplicación móvil.
2. Diseño e implementación de los Web Services SOAP del sistema SysDoc:
    a. Consulta de especialidades
    b. Consulta de médicos por especialidad
    c. Consulta de medicación
    d. Atención de paciente
    e. Recepción de callback ws-addressing
3. Diseño e implementación de los Web Services SOAP del sistema CliniDoc:
    a. Alta de estudio médico: implementado con WS-Addressing y respuestas desacopladas
    b. Consulta de estado de estudio médico: retorna si un estudio médico está disponible.
    c. Obtener estudio médico: implementado con MTOM para resultados que requieran adjuntos (PDF, imágenes, etc).
4. Diseño e implementación del Web Service de DGI con WS-Security.
5. Implementación en el ESB de flujo de integración entre el sistema ERP de facturación y Web Service de DGI.
6. Implementación de las APIs Restful a partir de los Web Services disponibles, utilizando transformaciones JSON-SOAP en el ESB de la organización.
7. Implementación de las APIs Restful del sistema de Agenda
    a. Consulta de agenda del afiliado.
    b. Alta y baja de cita.
8. Catálogo de APIs Restful en el ESB documentadas con Open API.
9. Todos los sistemas deben respetar los modelos de datos definidos en los anexos.

**Requerimientos no funcionales**

1. Los Web Services SOAP deberán implementarse con las librerías JAX-WS. Como servidor de aplicaciones se podrá usar Tomcat. Los Web Services deberán loggear todos los pedidos procesados en consola o en un archivo. Se debe utilizar Apache CXF como tecnología de Web Services.
2. Las APIs Restful se deben implementar con las librerías JAX-RS. Las APIs deberán loggear todos los mensajes recibidos.
3. El Web Service de DGI 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. Se puede usar un cliente SOAPUI para mostrar la evidencia.
4. Se debe utilizar MuleESB como implementación de ESB
5. En caso de utilizar una base de datos, se debe utilizar PostgreSQL
6. Se debe utilizar Java 11 para todas las implementaciones.
7. Se debe utilizar Postman o SOAPUI como emulador de la aplicación móvil
8. Se debe utilizar Postman o SOAPUI como emulador de la interfaz de usuario del sistema SysDoc donde se podrán consumir los Web Services de SysDoc y Clinidoc.
9. Se debe implementar el servidor OAuth2 de tokens utilizando Mule ESB. Se debe utilizar el ESB para validar estos tokens en las operaciones solicitadas.
10. Se deben usar los conectores de MuleESB para la integración con los Web Services de ProSalud, DGI y sistema de facturación.

**Entregables:**

1. Código fuente de todos los sistemas desarrollados
2. Documentación describiendo la arquitectura del sistema.
3. Documentación con casos de prueba ejecutados sobre el sistema

**Forma de entrega:** Para la entrega se usará el gitlab de Facultad, donde los grupos deberán subir el código al repositorio asignado. No se tomarán en cuenta entregas pasada la hora de entrega.

**Fecha de entrega:** 14 de noviembre hasta las 23:59hrs. No se aceptarán entregas pasada esta fecha/hora.

**Defensa:** Las fechas de las defensas serán el 16/11 y 18/11 a las 19hrs.

**Anexo 1: Modelo de datos**

Una cita médica debe incluir los siguientes datos:

- Fecha
- Hora
- Médico

Un diagnóstico debe incluir los siguientes datos:

- Médico tratante
- Paciente
- Descripción del diagnóstico
- Lista de recetas prescritas. Cada receta debe incluir
    - Nombre de medicamento
    - Indicaciones
- Lista de estudios a realizar
    - Nombre de estudio
    - Confirmado o pendiente.

**Anexo 2: Formato de archivo CSV**

El archivo CSV a generar para el sistema de facturación debe contener una línea por factura generada y cuya cabecera debe indicar el nombre de los campos:

- bill_id: identificador de la factura
- bill_type: tipo de factura. CR equivale a crédito y CO contado.
- date: fecha de la factura en formato ddMMyyyy
- amount_digits: monto completo con decimales sin separador.
- digits: cantidad de dígitos después de la coma
- tax_type: tipo de impuestos. 1 equivale a IVA y 2 IMESI
- tax_amount: monto asociado al impuesto
- tax_digits
- currency: moneda asociada al monto (858 para moneda pesos y 840 para dólares)

*Bill_id, bill_type ,date,type,amount,amount_digits, tax_amount,tax_type,currency*

*123456,NC,22062022,14850,2,1,3267,2,858*

**Anexo 3: Web Service DGI**

| | **Tipo de datos** | **Descripción** |
|---|---|---|
| Id_factura | String | Número de factura |
| Tipo_factura | Integer | 1: crédito<br>2: contado |
| Fecha | DateTime | |
| Monto nominal | Double | |
| Tipo impuesto | Integer | 1: IVA<br>2: IMESI |
| Mono impuesto | Double | |
| Moneda | Integer | 858: Pesos<br>840: Dólares |

- bill_id: identificador de la factura
- bill_type: tipo de factura. CR equivale a crédito y CO contado.
- date: fecha de la factura en formato ddMMyyyy
- amount: monto completo con decimales pero separador.
- amount_digits: cantidad de dígitos después de la coma asociado al monto
- tax_type: tipo de impuestos. 1 equivale a IVA y 2 IMESI
- tax_amount: monto asociado al impuesto
- tax_digits: cantidad de dígitos después de la coma asociado a los impuestos
- currency: moneda asociada al monto (858 para moneda pesos y 840 para dólares)

## Referencias


1. MuleESB:
    a. Anypoint Studio: https://www.mulesoft.com/platform/studio
    b. Anypoint Studio java troubleshooting: https://docs.mulesoft.com/studio/7.10/faq-default-browser-config
    c. Anypoint is cripplingly slow: https://help.mulesoft.com/s/question/0D52T00004mXXrISAW/anypoint-studio-7-is-cripplingly-slow
    d. Designing your First API: https://developer.mulesoft.com/guides/quick-start/designing-your-first-api
    e. Developing your First application: https://developer.mulesoft.com/tutorials-and-howtos/quick-start/developing-your-first-mule-application
    f. Mule App development tutorial: https://docs.mulesoft.com/mule-runtime/4.3/mule-app-tutorial
    g. Consume API Rest: https://docs.mulesoft.com/mule-runtime/4.3/consume-data-from-an-api
    h. Web Service Consumer Connector: https://docs.mulesoft.com/web-service-consumer-connector/1.6/
    i. Web Service Consumer Tutorial: https://www.youtube.com/watch?v=ZCySANp4SIo
    j. HTTP Connector: https://docs.mulesoft.com/http-connector/1.5/
    k. Lookup data in a CSV File: https://docs.mulesoft.com/mule-runtime/4.3/dataweave-cookbook-csv-lookup
    l. Transform XML to JSON: https://docs.mulesoft.com/mule-runtime/4.3/dataweave-cookbook-perform-basic-transformation
    m. XML to CSV mapping: https://www.youtube.com/watch?v=M8rC0sktwyY
    n. Import API from Desing Center into Project: https://docs.mulesoft.com/studio/7.10/import-api-specification
    o. Cache Scope: https://docs.mulesoft.com/mule-runtime/4.3/cache-scope
    p. JMS Connector: https://docs.mulesoft.com/jms-connector/1.7/
    q. OAuth2 Provider Module (Utilizar version 1.0.5):
        i. https://docs.mulesoft.com/oauth2-provider-module/1.0/
        ii. https://www.mulesoft.com/exchange/com.mulesoft.modules/mule-oauth2-provider-module/
        iii. https://apisero.com/implementing-mulesoft-as-oauth-provider-for-securing-mule-application/
    r. HTTP Error Status Code and reason phrase: https://docs.mulesoft.com/http-connector/1.5/http-error-status-reason-phrase-task
2. Postman: https://www.getpostman.com/
3. SOAPUI: http://www.soapui.org/
4. Keytool: https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html
5. CXF-WSSecurity: http://cxf.apache.org/docs/ws-security.html
6. CXF-WSAddressing: https://cxf.apache.org/docs/ws-addressing.html
7. Java Addressing: https://docs.oracle.com/cd/E24329_01/web.1211/e24965/wsaddressing.htm#WSADV691

8. MTOM: https://docs.oracle.com/cd/E12839_01/web.1111/e13734/mtom.htm#WSADV138
9. Deploy JAX-WS Web Service on Tomcat: https://www.mkyong.com/webservices/jax-ws/deploy-jax-ws-web-services-on-tomcat/
10. JAX-RS:
    a. https://docs.oracle.com/javaee/6/tutorial/doc/giepu.html
    b. https://www.baeldung.com/jax-rs-spec-and-implementations