**Alternativas a los servicios Web basados en SOAP**

Los servicios web basados en SOAP funcionan bien en ciertas situaciones:

* Proporcionan una plataforma de comunicaciones interoperable entre sistemas informáticos.
* Comunicación entre diferentes partes de un motor de procesos de negocios y soluciones de gestión de procesos de negocios (BPM).

Las estructuras de datos basadas en XML tienden a ser más detalladas de lo necesario para clientes simples, tal como las aplicaciones JavaScript en un escenario de navegador web a servidor.

**Web SOA (Web 2.0) versus Enterprise SOA**

Web 2.0 no hace referencia a una actualización real de ninguna especificación técnica, pero refleja un cambio en la forma en que se desarrollan y utilizan las páginas web. También refleja un conjunto diferente de herramientas y lenguajes de desarrollo.

Imagen que contiene texto, mapa

Descripción generada automáticamente

La SOA empresarial y la SOA web tienen diferentes estándares y protocolos. A veces, un cliente Web 2.0 necesita acceder a la SOA de la empresa, por lo que es necesario algún código de puente.

**Web SOA protocols and standards**

* Protocolos y técnicas de transporte e invocación.
* Hypertext Transfer Protocol (HTTP, HTTPS)
* Representational State Transfer (REST)
* Comet

Estándares y técnicas de formato de datos :

* Extensible Markup Language (XML)–Plain old XML over HTTP (POX/HTTP)
* JavaScript Object Notation (JSON)
* JSON-RPC
* Bayeux
* Really Simple Syndication (RSS)
* Atom

**Introducción a REST**

La transferencia de estado representacional (REST) es una arquitectura para acceder a los recursos a través de una red:

* El estado y las funciones de la aplicación se dividen en recursos.
* Cada recurso es direccionable de manera única con sintaxis universal.
* Todos los recursos son accesibles con una interfaz uniforme y genérica.
* Cada solicitud del cliente al servidor debe contener toda la información necesaria para comprender la solicitud.
* Las arquitecturas REST no pueden confiar en el contexto almacenado en el servidor.
* REST es un patrón de diseño, no un **estándar.**
* En un contexto de Web 2.0, REST describe una forma de diseñar aplicaciones web que:
* Aborden recursos a través de URI
* Acceda a recursos a través de métodos HTTP

Operaciones

* GET
* PUT
* DELETE
* POST

Imagen que contiene captura de pantalla

Descripción generada automáticamente

La primera interacción muestra una versión RESTfull de crear un nuevo empleado utilizando un HTTP POST.  
La respuesta HTTP incluye el número de identificación del nuevo empleado.  
La segunda interacción utiliza un GET para recuperar información sobre el empleado 101.  
La última interacción usa DELETE para eliminar al empleado 101.

**JOSE** es un elemento crítico para proporcionar integridad y confidencialidad del mensaje en los escenarios REST y JSON.

Algunas partes de las especificaciones JOSE respaldan esta necesidad al requerir una codificación "segura para URL"

Algunos tipos de cargas útiles deben asegurarse para la confidencialidad y la integridad, SSL / TLS proporciona algo de eso, pero solo durante la transacción generalmente no proporciona no repudio.

La seguridad de los servicios web (WS-Security) proporciona un buen modelo para proteger las cargas útiles(payloads).

Imagen que contiene captura de pantalla

Descripción generada automáticamente

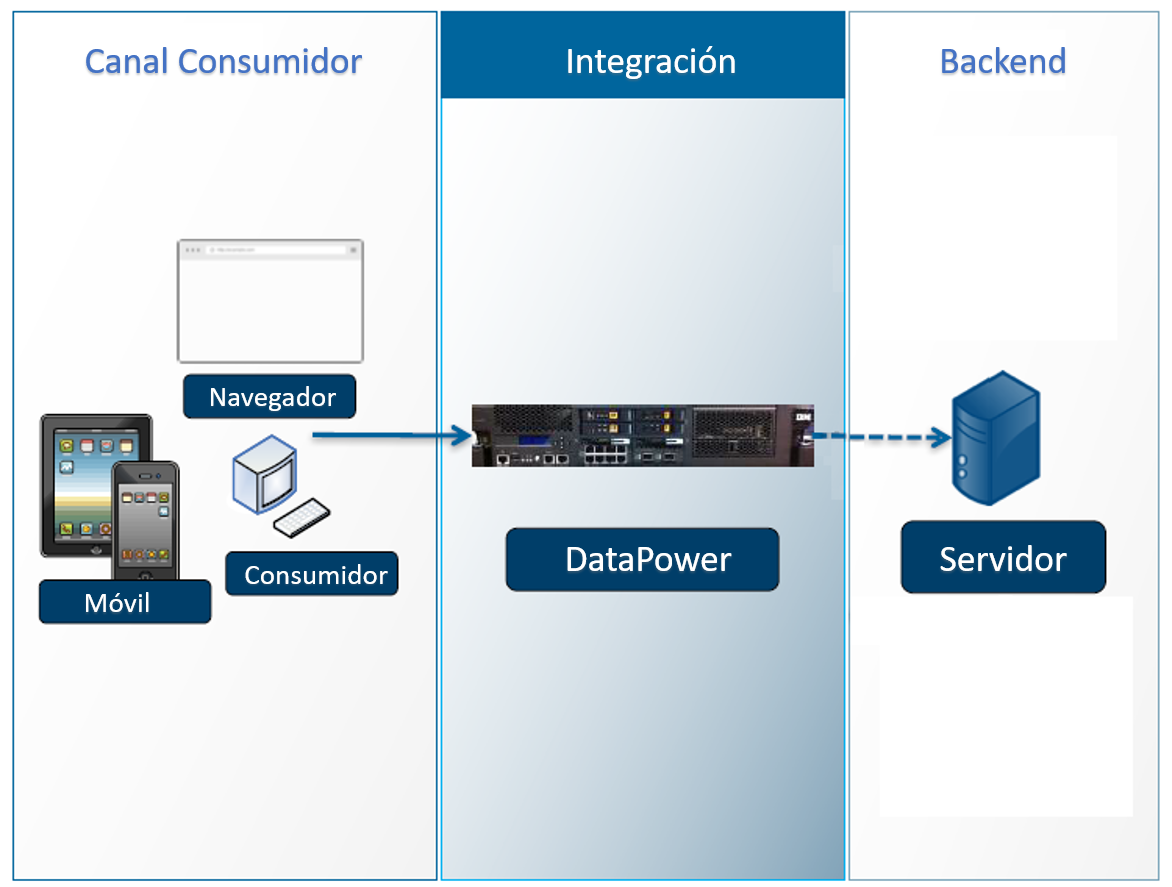
**JWA “JSON Web Algorithms”:** Registra algoritmos e identificadores criptográficos para ser utilizados con las especificaciones JWS, JWE y JWK. Define varios registros de la IANA para estos identificadores.

**JWK “JSON Web Key”**: Es una estructura de datos JSON que representa una clave criptográfica.

**JWS “JSON Web Signature”:** Asegura el contenido con firmas digitales o códigos de autenticación de mensajes mediante el uso de estructuras de datos JSON.

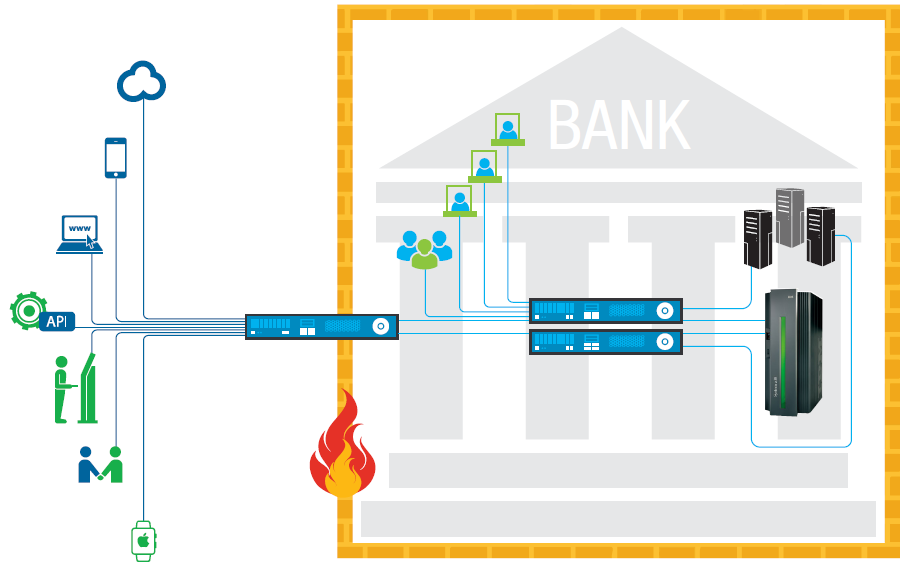
**JWE “JSON Web Encryption”:** Representa contenido cifrado mediante el uso de estructuras de datos JSON.

**JWT “JSON Web Token”**: Es un objeto JSON que representa reclamos entre dos partes.  
Es una carga útil de una estructura JWS o como el texto sin formato de una estructura JWE, lo que permite que las reclamaciones se firmen digitalmente o se cifren o se cifren, *No es parte de JOSE, pero usa JOSE para proteger el token.*

**

Durante más de una década, la mayoría de las instituciones financieras más grandes del mundo han confiado en IBM DataPower Gateway para cumplir con sus objetivos de seguridad e integración, lo que les permite adaptarse de forma segura y rápida a las tendencias tecnológicas a medida que ocurren.

La enterprise es tanto un consumidor como un proveedor de información, y las API son la lengua franca para el intercambio de información. IBM DataPower Gateway admite los últimos estándares de la industria para ayudarlo a participar en la economía API.



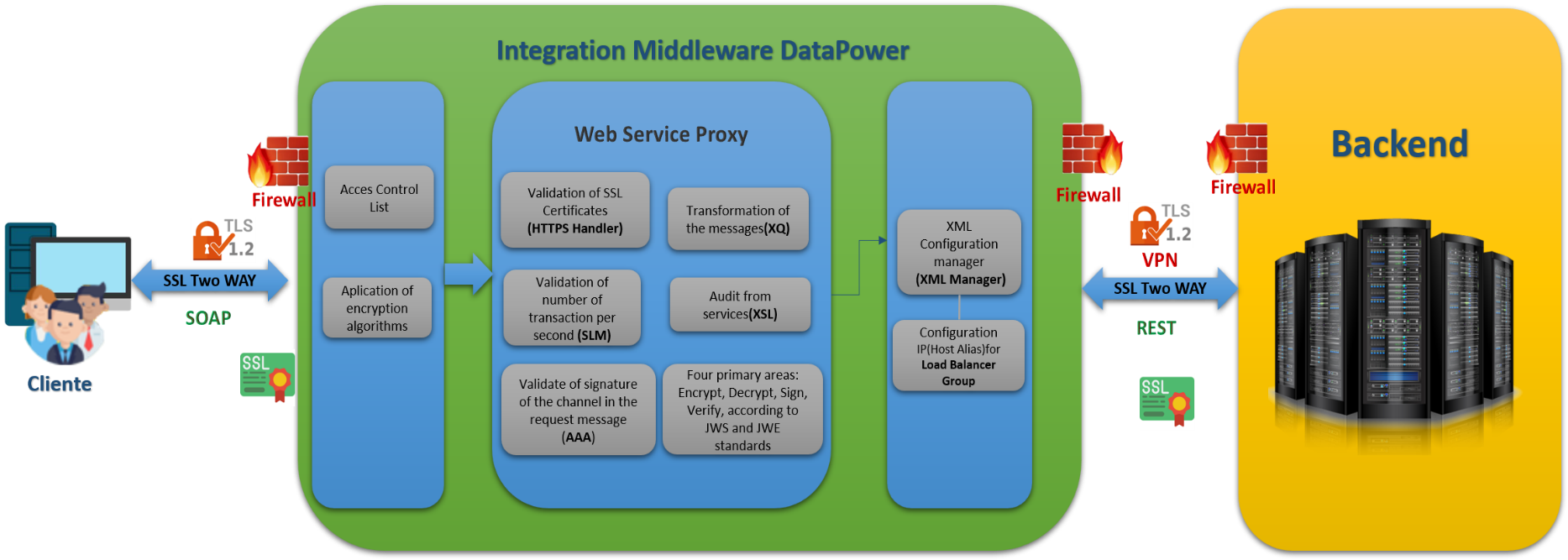
¿Qué tipo de servicio deberías usar?

Si está enfocado en WSDL y servicios web, elija el proxy del servicio web

Presente un único WSDL virtual a los clientes, estará compuesto por múltiples operaciones en el back-end.

**Escenarios de PRUEBA**

El flujo de negocio que se desea resolver esta especificado en el siguiente diagrama:



**1Parte: Orquestador**

**Web Service Proxy**

Los aspectos relevantes en transporte para consumo son:

Exposición para consumo en Datapower Externo.

Backend HTTPS TLS 1.2 mutual (two way), cliente y servidor presentan certificados. Respecto a la mensajería SOAP enviada desde Cliente se tienen los siguientes aspectos:

Se remueven headers de WS-Security antes de enviar a clientes.

En mensajería SOAP response enviada desde backend.

Firma digital WS-Security con algoritmo de firma rsa-sha512 http://www.w3.org/2001/04/xmldsig-more#rsa-sha512, algoritmo de huella (digest) de mensaje http://www.w3.org/2001/04/xmlenc#sha512, referencia de token BinarySecurityToken. La regla de layout a aplicar al encabezado de seguridad (Security Header Layout) será strict. Se firmará header de timestamp y body de mensaje SOAP.

Validación de credenciales extraídas de header WS-SEC UserNameToken con password de tipo PasswordText con LDAP Tivoli TDS actual.

**2Parte:**

**Request** REST

**Response** SOAP

**Pendiente**

**--- REST JOSE**

**Creating and verifying a JWS**

Cifrado de mensaje WS-Security utilizando el certificado público. Se utilizará el algoritmo AES256-CBC para el cifrado, referencia de token X509SubjectKeyIdentifier, algoritmo para transportar la llave RSA-PKCS1, tipo de llave de cifrado: llave efímera transportada por algoritmo asimétrico. Se cifrará Body con el certificado público.

Descifrado de mensaje con WS-Security, utilizando la llave privada. Validación de firma digital utilizando el certificado público

DIAGRAMA FUNCIONAL