INVOCACIÓN DE SERVICIOS: SOAP

Carlos Maldonado

5IV7

1. Introducción

En el mundo moderno de la computación distribuida y los servicios web, la comunicación entre aplicaciones se ha convertido en un aspecto fundamental. El protocolo SOAP (Simple Object Access Protocol) emerge como uno de los estándares más importantes para el intercambio de información estructurada en entornos distribuidos. Este trabajo profundiza en los conceptos, características y aplicaciones prácticas de SOAP, proporcionando una visión completa de su funcionamiento y relevancia en el desarrollo de software empresarial.

2. ¿Qué es el Protocolo SOAP?

SOAP (Simple Object Access Protocol) es un protocolo de comunicación basado en XML diseñado específicamente para el intercambio de información en entornos distribuidos y descentralizados. A diferencia de otros protocolos web, SOAP define un formato estricto y estandarizado para la estructura de los mensajes, permitiendo la comunicación entre aplicaciones a través de diferentes plataformas, lenguajes de programación y sistemas operativos.

Definición técnica: SOAP es un protocolo ligero para el intercambio de información en entornos descentralizados y distribuidos. Utiliza XML como formato de mensaje y se basa en otros protocolos de aplicación como HTTP, SMTP o TCP para la negociación y transmisión del mensaje.

Características fundamentales:

- **Independencia de plataforma:** Funciona en cualquier sistema operativo
- **Neutralidad del lenguaje:** Puede ser implementado en cualquier lenguaje de programación
- **Extensibilidad:** Permite agregar características y funcionalidades
- **Interoperabilidad:** Facilita la comunicación entre sistemas heterogéneos

3. Historia y Evolución

SOAP fue desarrollado originalmente por Microsoft en 1998 como parte de su iniciativa COM y posteriormente evolucionó hacia un estándar abierto. La versión 1.1 fue presentada al W3C en el año 2000, y la versión 1.2 se convirtió en recomendación oficial del W3C en 2003.

Linea temporal:

- **1998:** Desarrollo inicial por Microsoft

- **2000:** SOAP 1.1 presentado al W3C
- **2003:** SOAP 1.2 se convierte en recomendación W3C
- **2007:** Publicación de WS-* standards
- **Actualidad:** Continúa siendo ampliamente utilizado en entornos empresariales

4. Características Principales

4.1. Basado en XML

SOAP utiliza exclusivamente XML para formatear los mensajes, lo que garantiza:

- Legibilidad humana y máquina
- Estructura jerárquica y organizada
- Soporte para datos complejos
- Validación mediante esquemas XML

4.2. Independencia de Protocolo de Transporte

Aunque comúnmente se usa sobre HTTP, SOAP puede operar sobre:

- HTTP/HTTPS
- SMTP
- TCP
- JMS
- FTP

4.3. Orientado a Documentos

SOAP está diseñado para intercambiar documentos XML estructurados más que para invocaciones remotas simples.

4.4. Extensibilidad

Mediante el uso de headers SOAP, el protocolo puede extenderse para incluir:

- Información de seguridad
- Transacciones
- Contexto de negocio
- Metadatos

4.5. Neutralidad Lingüística

Puede ser implementado en:

- Java
- C#
- Python
- PHP
- C++
- Y muchos otros lenguajes

4.6. Soporte para Operaciones Síncronas y Asíncronas

5. Estructura del Mensaje SOAP

Un mensaje SOAP típico consta de los siguientes elementos:

5.1. Envelope

Elemento raíz que define el documento como mensaje SOAP.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
</soap:Envelope>
```

5.2. Header (Opcional)

Contiene información auxiliar como:

- Autenticación
- Información de transacciones
- Metadatos

```
<soap:Header>
 <auth:Authentication xmlns:auth="http://ejemplo.com/auth">
  <username>usuario</username>
  <password>contraseña</password>
 </auth:Authentication>
</soap:Header>
```

5.3. Body (Obligatorio)

Contiene la información principal del mensaje y la llamada al método.

```
<soap:Body>
 <m:GetTemperatura xmlns:m="http://ejemplo.com/temperatura">
  <m:ciudad>Madrid</m:ciudad>
 </m:GetTemperatura>
</soap:Body>
```

5.4. Fault (Opcional)

Para manejo de errores dentro del Body.

```
<soap:Fault>
 <faultcode>SOAP-ENV:Client</faultcode>
 <faultstring>Error en la solicitud</faultstring>
 <detail>
  <e:myfaultdetails xmlns:e="http://ejemplo.com/faults">
   <message>Ciudad no encontrada</message>
   <errorcode>1234</errorcode>
  </e:myfaultdetails>
 </detail>
</soap:Fault>
```

6. Componentes del Protocolo

6.1. WSDL (Web Services Description Language)

Define la interfaz del servicio web:

- **Tipos:** Definiciones de tipos de datos
- **Mensajes:** Estructura de los mensajes
- **PortType:** Operaciones disponibles
- **Binding:** Protocolo y formato de mensaje
- **Service:** Dirección del servicio

6.2. UDDI (Universal Description, Discovery and Integration)

Directorio para publicar y descubrir servicios web.

6.3. XML Schema

Define la estructura y tipos de datos de los mensajes.

7. Ventajas y Desventajas

Ventajas:

- **Estándar robusto:** Amplia especificación y soporte
- **Seguridad integrada:** WS-Security para mensajes seguros
- **Transacciones:** Soporte para transacciones distribuidas
- **Manejo de errores:** Mecanismos estandarizados de fault
- **Tooling:** Amplio soporte en herramientas de desarrollo
- **Empresarial:** Ideal para aplicaciones críticas

Desventajas:

- **Complejidad:** Mayor overhead comparado con REST
- **Verbosidad:** Mensajes XML más largos
- **Rendimiento:** Mayor consumo de ancho de banda
- **Curva de aprendizaje:** Más complejo de implementar
- **Caching:** Limitaciones en el cacheo de respuestas

8. Comparación con REST

Aspecto	*SOAP*	*REST*
Protocolo	Puede usar múltiples protocolos	Principalmente HTTP
Formato de datos	XML exclusivamente	XML, JSON, HTML, texto plano
Estándares	WS-*, WSDL, SOAP	HTTP, URI, JSON
Estado	Sin estado	Sin estado
Seguridad	WS-Security	HTTPS, OAuth
Complejidad	Alta	Baja a media
Caching	Limitado	Excelente
Casos de uso	Empresarial, transaccional	Web, móvil, IoT

9. Casos de Uso y Aplicaciones

9.1. Aplicaciones Empresariales

- Sistemas ERP (Enterprise Resource Planning)
- Integración de sistemas legacy
- Procesos transaccionales complejos

9.2. Servicios Financieros

- Transacciones bancarias
- Procesamiento de pagos
- Sistemas de compensación

9.3. Telecomunicaciones

- Sistemas de facturación
- Gestión de servicios
- Provisioning de recursos

9.4. Gobierno y Salud

- Sistemas de información hospitalaria
- Registros electrónicos
- Servicios gubernamentales

10. Ejemplo 1: Servicio Web de Clima

10.1. Descripción del Servicio

Servicio que proporciona información meteorológica para ciudades específicas.

10.2. WSDL del Servicio

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://www.ejemplo.com/clima"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.ejemplo.com/clima">

 <types>
  <xsd:schema targetNamespace="http://www.ejemplo.com/clima">
   <xsd:element name="ObtenerClimaRequest">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="ciudad" type="xsd:string"/>
      <xsd:element name="pais" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>

   <xsd:element name="ObtenerClimaResponse">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="temperatura" type="xsd:float"/>
      <xsd:element name="humedad" type="xsd:int"/>
      <xsd:element name="descripcion" type="xsd:string"/>
      <xsd:element name="fecha" type="xsd:dateTime"/>
     </xsd:sequence>
```

```
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
 </types>
 <message name="ObtenerClimaInput">
  <part name="body" element="tns:ObtenerClimaRequest"/>
 </message>

 <message name="ObtenerClimaOutput">
  <part name="body" element="tns:ObtenerClimaResponse"/>
 </message>

 <portType name="ClimaPortType">
  <operation name="ObtenerClima">
   <input message="tns:ObtenerClimaInput"/>
   <output message="tns:ObtenerClimaOutput"/>
  </operation>
 </portType>
 <binding name="ClimaBinding" type="tns:ClimaPortType">
  <soap:binding                                                    style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="ObtenerClima">
   <soap:operation soapAction="http://www.ejemplo.com/clima/ObtenerClima"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </operation>
 </binding>

 <service name="ClimaService">
  <port name="ClimaPort" binding="tns:ClimaBinding">
   <soap:address location="http://www.ejemplo.com/clima/service"/>
  </port>
 </service>
</definitions>
```

10.3. Solicitud SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
        xmlns:cli="http://www.ejemplo.com/clima">
 <soap:Header>
  <cli:Authentication>
   <apiKey>ABC123XYZ</apiKey>
  </cli:Authentication>
 </soap:Header>
 <soap:Body>
  <cli:ObtenerClimaRequest>
   <ciudad>Madrid</ciudad>
   <pais>España</pais>
  </cli:ObtenerClimaRequest>
 </soap:Body>
</soap:Envelope>
```

10.4. Respuesta SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
        xmlns:cli="http://www.ejemplo.com/clima">
 <soap:Body>
  <cli:ObtenerClimaResponse>
   <temperatura>22.5</temperatura>
   <humedad>65</humedad>
   <descripcion>Parcialmente nublado</descripcion>
   <fecha>2024-01-15T14:30:00Z</fecha>
  </cli:ObtenerClimaResponse>
 </soap:Body>
</soap:Envelope>
```

10.5. Implementación en Java

```
@WebService(targetNamespace = "http://www.ejemplo.com/clima")
public class ServicioClima {

  @WebMethod
  public ObtenerClimaResponse obtenerClima(ObtenerClimaRequest request) {
    ObtenerClimaResponse response = new ObtenerClimaResponse();

    // Lógica de negocio para obtener el clima
    ClimaData datosClima = ClimaService.obtenerClima(
      request.getCiudad(),
      request.getPais()
    );

    response.setTemperatura(datosClima.getTemperatura());
    response.setHumedad(datosClima.getHumedad());
    response.setDescripcion(datosClima.getDescripcion());
    response.setFecha(datosClima.getFecha());

    return response;
  }
}
```

11. Ejemplo 2: Sistema Bancario

11.1. Descripción del Servicio

Sistema transaccional bancario que maneja transferencias, consultas de saldo y procesamiento de pagos.

11.2. WSDL del Servicio Bancario

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://www.ejemplo.com/banco"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.ejemplo.com/banco">

 <types>
  <xsd:schema targetNamespace="http://www.ejemplo.com/banco">

   <!-- Transferencia Request -->
   <xsd:element name="TransferenciaRequest">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="cuentaOrigen" type="xsd:string"/>
      <xsd:element name="cuentaDestino" type="xsd:string"/>
      <xsd:element name="monto" type="xsd:decimal"/>
      <xsd:element name="moneda" type="xsd:string"/>
      <xsd:element name="concepto" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>

   <!-- Transferencia Response -->
   <xsd:element name="TransferenciaResponse">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="numeroTransaccion" type="xsd:string"/>
      <xsd:element name="estado" type="xsd:string"/>
      <xsd:element name="fechaProcesamiento" type="xsd:dateTime"/>
      <xsd:element name="saldoActual" type="xsd:decimal"/>
```

```
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>

  <!-- Consulta Saldo Request -->
  <xsd:element name="ConsultaSaldoRequest">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="numeroCuenta" type="xsd:string"/>
     <xsd:element name="tipoConsulta" type="xsd:string"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>

  <!-- Consulta Saldo Response -->
  <xsd:element name="ConsultaSaldoResponse">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="saldoDisponible" type="xsd:decimal"/>
     <xsd:element name="saldoContable" type="xsd:decimal"/>
     <xsd:element name="moneda" type="xsd:string"/>
     <xsd:element name="fechaConsulta" type="xsd:dateTime"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
 </xsd:schema>
</types>

<!-- Mensajes para Transferencia -->
<message name="TransferenciaInput">
 <part name="body" element="tns:TransferenciaRequest"/>
</message>
<message name="TransferenciaOutput">
 <part name="body" element="tns:TransferenciaResponse"/>
</message>

<!-- Mensajes para Consulta Saldo -->
<message name="ConsultaSaldoInput">
```

```
  <part name="body" element="tns:ConsultaSaldoRequest"/>
 </message>
 <message name="ConsultaSaldoOutput">
  <part name="body" element="tns:ConsultaSaldoResponse"/>
 </message>
 <portType name="BancoPortType">
  <operation name="RealizarTransferencia">
   <input message="tns:TransferenciaInput"/>
   <output message="tns:TransferenciaOutput"/>
  </operation>
  <operation name="ConsultarSaldo">
   <input message="tns:ConsultaSaldoInput"/>
   <output message="tns:ConsultaSaldoOutput"/>
  </operation>
 </portType>
 <binding name="BancoBinding" type="tns:BancoPortType">
  <soap:binding                                                    style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="RealizarTransferencia">
   <soap:operation
soapAction="http://www.ejemplo.com/banco/RealizarTransferencia"/>
   <input><soap:body use="literal"/></input>
   <output><soap:body use="literal"/></output>
  </operation>
  <operation name="ConsultarSaldo">
   <soap:operation
soapAction="http://www.ejemplo.com/banco/ConsultarSaldo"/>
   <input><soap:body use="literal"/></input>
   <output><soap:body use="literal"/></output>
  </operation>
 </binding>
 <service name="BancoService">
  <port name="BancoPort" binding="tns:BancoBinding">
   <soap:address location="https://www.ejemplo.com/banco/service"/>
  </port>
 </service>
</definitions>
```

11.3. Solicitud de Transferencia

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:ban="http://www.ejemplo.com/banco">
 <soap:Header>
  <ban:Security>
   <ban:UsernameToken>
    <ban:Username>usuario_bancario</ban:Username>
    <ban:Password Type="ban:PasswordText">clave_segura</ban:Password>
   </ban:UsernameToken>
  </ban:Security>
  <ban:TransactionID>TX987654321</ban:TransactionID>
 </soap:Header>
 <soap:Body>
  <ban:TransferenciaRequest>
   <cuentaOrigen>ES1234567890123456789012</cuentaOrigen>
   <cuentaDestino>ES9876543210987654321098</cuentaDestino>
   <monto>1500.00</monto>
   <moneda>EUR</moneda>
   <concepto>Transferencia mensual</concepto>
  </ban:TransferenciaRequest>
 </soap:Body>
</soap:Envelope>
```

11.4. Respuesta de Transferencia

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:ban="http://www.ejemplo.com/banco">
 <soap:Body>
  <ban:TransferenciaResponse>
   <numeroTransaccion>TXN123456789</numeroTransaccion>
   <estado>COMPLETADA</estado>
   <fechaProcesamiento>2024-01-15T10:30:45Z</fechaProcesamiento>
   <saldoActual>8500.00</saldoActual>
  </ban:TransferenciaResponse>
 </soap:Body>
</soap:Envelope>
```

11.5. Implementación en C#

```
[WebService(Namespace = "http://www.ejemplo.com/banco")]
public class ServicioBancario : WebService
{
  [WebMethod]
  [SoapHeader("Security", Direction = SoapHeaderDirection.In)]
  public TransferenciaResponse RealizarTransferencia(TransferenciaRequest request)
  {
    // Validar seguridad
    if (!ValidarCredenciales(Security))
    {
      throw new SoapException("Credenciales inválidas",
        SoapException.ClientFaultCode);
    }

    // Procesar transferencia
    var resultado = ProcesadorTransferencias.Procesar(
      request.CuentaOrigen,
      request.CuentaDestino,
      request.Monto,
      request.Moneda,
      request.Concepto
    );

    return new TransferenciaResponse
    {
      NumeroTransaccion = resultado.NumeroTransaccion,
      Estado = resultado.Estado,
      FechaProcesamiento = resultado.FechaProcesamiento,
      SaldoActual = resultado.SaldoActual
    };
  }

  [WebMethod]
  [SoapHeader("Security", Direction = SoapHeaderDirection.In)]
  public ConsultaSaldoResponse ConsultarSaldo(ConsultaSaldoRequest request)
  {
```

```
        if (!ValidarCredenciales(Security))
        {
            throw new SoapException("Credenciales inválidas",
                SoapException.ClientFaultCode);
        }

        var saldo = ServicioSaldos.ObtenerSaldo(
            request.NumeroCuenta,
            request.TipoConsulta
        );

        return new ConsultaSaldoResponse
        {
            SaldoDisponible = saldo.SaldoDisponible,
            SaldoContable = saldo.SaldoContable,
            Moneda = saldo.Moneda,
            FechaConsulta = DateTime.UtcNow
        };
    }

    private bool ValidarCredenciales(SecurityHeader security)
    {
        // Lógica de validación
        return ServicioAutenticacion.ValidarUsuario(
            security.UsernameToken.Username,
            security.UsernameToken.Password
        );
    }
}
```

12. Implementación y Herramientas

12.1. Herramientas de Desarrollo

- **Java:** JAX-WS, Apache CXF, Spring WS
- **.NET:** WCF, ASP.NET Web Services
- **Python:** Zeep, SUDS
- **PHP:** NuSOAP, PHP SOAP extension

12.2. Mejores Prácticas

- Diseñar WSDL primero (Contract-first)
- Usar namespaces apropiados
- Implementar manejo robusto de errores
- Considerar versionamiento de servicios
- Documentar exhaustivamente

12.3. Patrones Comunes

- **Document Literal:** Estilo preferido para interoperabilidad
- **RPC Literal:** Para compatibilidad con sistemas legacy
- **WS-Addressing:** Para enrutamiento de mensajes
- **MTOM:** Para transferencia de archivos binarios

13. Seguridad en SOAP

13.1. WS-Security

Estándar para asegurar mensajes SOAP:

- **Autenticación:** UsernameToken, X.509 certificates
- **Confidencialidad:** Encryption XML
- **Integridad:** Digital signatures
- **Timestamp:** Prevención de replay attacks

13.2. Ejemplo de Seguridad

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
 <soap:Header>
  <wsse:Security>
   <wsse:UsernameToken wsu:Id="UsernameToken-1">
    <wsse:Username>usuario</wsse:Username>
    <wsse:Password     Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">contraseña</wsse:Password>
   </wsse:UsernameToken>
   <wsu:Timestamp wsu:Id="Timestamp-1">
    <wsu:Created>2024-01-15T10:00:00Z</wsu:Created>
    <wsu:Expires>2024-01-15T10:05:00Z</wsu:Expires>
   </wsu:Timestamp>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <!-- Contenido del mensaje -->
 </soap:Body>
</soap:Envelope>
```

14. Futuro de SOAP

Aunque REST y GraphQL han ganado popularidad, SOAP mantiene su relevancia en:

- **Aplicaciones críticas que requieren transacciones ACID**
- **Entornos con requisitos de seguridad complejos**
- **Integración de sistemas heterogéneos**

Tendencias actuales:

- Microservicios con SOAP para servicios core
- Hybrid approaches (SOAP + REST)
- Cloud integration
- API modernization

15. Conclusión

SOAP representa un protocolo maduro y robusto para la implementación de servicios web en entornos empresariales. Su fortaleza reside en la estandarización, seguridad y capacidades transaccionales que ofrece. Aunque puede ser más complejo que alternativas como REST, sigue siendo la elección preferida para aplicaciones que requieren alta confiabilidad, seguridad y soporte para operaciones transaccionales complejas.

Los ejemplos presentados demuestran la versatilidad de SOAP en diferentes dominios, desde servicios simples de consulta hasta sistemas transaccionales críticos. La decisión entre SOAP y otras arquitecturas debe basarse en los requisitos específicos del proyecto, considerando factores como la complejidad, seguridad, interoperabilidad y mantenimiento a largo plazo.

BIBLIOGRAFÍA

- Wikipedia — *Simple Object Access Protocol (SOAP)*. Artículo explicativo con estructura del mensaje, características y ejemplos de mensajes SOAP (petición/respuesta). Wikipedia
- W3C — *SOAP Version 1.2 Part 0: Primer (Second Edition)*. Documento primer (tutorial) de la especificación SOAP 1.2; muy útil para entender el estándar y casos de uso. W3C
- IBM Docs (es) — *¿Qué es SOAP?* Explicación técnica, versiones (1.1 / 1.2) y referencias a WSDL/WS-*; incluye notas de implementación. IBM
- IBM Docs (es) — *La estructura de un mensaje SOAP* (ejemplos de Envelope, Header, Body y Fault). **(Recomendado como ejemplo #1)**. IBM
- Arsys (blog) — *¿Qué es SOAP (Simple Object Access Protocol)?* Resumen claro y actual sobre uso e historia; bueno para la introducción y características. Arsys
- IONOS DigitalGuide (es) — *SOAP: explicación del protocolo de red*. Guía práctica y comparativa con REST; útil para la sección de usos y ventajas/desventajas. IONOS
- IBM (docs generales) — *SOAP 1.1 y 1.2* (diferencias y notas sobre la recomendación W3C). Buen recurso para hablar de versiones y compatibilidades. IBM
- Google Developers (es) — *Encabezados de respuesta y solicitud XML de SOAP* (ejemplos prácticos de cabeceras SOAP usados en APIs). Útil para ejemplos concretos de headers y estilos (document/literal). Google for Developers
- Automation Anywhere (docs en español) — *Ejemplo de uso: acción servicio web SOAP* (ejemplo práctico: pasar dos valores y devolver suma). **(Recomendado como ejemplo #2 — ejemplo paso a paso práctico).** Documentación de Automation Anywhere
- Informatica (docs en español) — *Respuesta y solicitud SOAP de ejemplo* (ejemplos de solicitud/respuesta en contexto empresarial). Útil si quieres ejemplos más formales/enterprise.