**Universidad Politécnica de Tlaxcala Región Poniente**

**(Uptrep)**

**TECNOLOGÍAS Y APLICACIONES EN INTERNET**

**“REPORTE DE API’s”**

**Nombre:**  
Jeni Pamela Espinoza Rivera

**Maestro(a):**  
Vanesa Tenopala Zavala

**10/MARZO/2025**

**Índice**

1. Introducción
2. API de SOAP 3
   * Definición
   * Características
   * Ejemplo de implementación
3. API de RPC 4
   * Definición
   * Características
   * Ejemplo de implementación
4. API de WebSocket 5
   * Definición
   * Características
   * Ejemplo de implementación
5. API de REST 5
   * Definición
   * Características
   * Ejemplo de implementación
6. Conclusión 6
7. Bibliografía 6

**1. Introducción**

Las APIs (Interfaz de Programación de Aplicaciones) permiten la comunicación entre sistemas y aplicaciones. En este informe se analizarán cuatro tipos principales de APIs utilizadas en el desarrollo de aplicaciones modernas: **SOAP, RPC, WebSocket y REST**.

**2. API de SOAP**

**Definición**

SOAP (Simple Object Access Protocol) es un protocolo basado en XML para el intercambio de información estructurada en redes.

**Características**

* Utiliza **XML** para la estructura de los mensajes.
* Requiere **WSDL** (Web Services Description Language) para su definición.
* Compatible con **HTTP, SMTP y otros protocolos de transporte**.
* Seguridad mediante **WS-Security**.

**Ejemplo de implementación en Python (Cliente SOAP)**

python

CopyEdit

from zeep import Client

client = Client('http://www.dneonline.com/calculator.asmx?WSDL')

result = client.service.Add(10, 5)

print("Resultado de la suma:", result)

**3. API de RPC**

**Definición**

RPC (Remote Procedure Call) permite ejecutar procedimientos en un servidor remoto como si fueran funciones locales.

**Características**

* Usa **JSON-RPC o XML-RPC** como formato de comunicación.
* Ideal para **microservicios**.
* Puede operar sobre **HTTP o WebSockets**.

**Ejemplo de implementación en Python (JSON-RPC)**

Servidor:

from jsonrpclib.SimpleJSONRPCServer import SimpleJSONRPCServer

def sumar(a, b):

return a + b

server = SimpleJSONRPCServer(('localhost', 5000))

server.register\_function(sumar, 'sumar')

print("Servidor RPC activo en el puerto 5000")

server.serve\_forever()

Cliente:

import jsonrpclib

server = jsonrpclib.ServerProxy('http://localhost:5000')

print("Resultado de la suma:", server.sumar(10, 5))

**4. API de WebSocket**

**Definición**

WebSocket es un protocolo que permite **comunicación bidireccional** en tiempo real entre el cliente y el servidor.

**Características**

* Usa el **protocolo ws:// o wss://**.
* Permite **envío y recepción de datos simultáneamente**.
* Ideal para **chat en vivo, juegos en línea y notificaciones en tiempo real**.

**Ejemplo de implementación en JavaScript**

Servidor (Node.js con WebSocket):

const WebSocket = require('ws');

const server = new WebSocket.Server({ port: 8080 });

server.on('connection', socket => {

console.log('Cliente conectado');

socket.on('message', message => {

console.log('Mensaje recibido:', message);

socket.send('Echo: ' + message);

});

});

Cliente (HTML y JavaScript):

<script>

let ws = new WebSocket("ws://localhost:8080");

ws.onopen = () => ws.send("Hola servidor!");

ws.onmessage = event => console.log("Respuesta:", event.data);

</script>

**5. API de REST**

**Definición**

REST (Representational State Transfer) es un estilo de arquitectura para desarrollar **servicios web basados en HTTP**.

**Características**

* Utiliza **métodos HTTP** como GET, POST, PUT y DELETE.
* Formato de respuesta comúnmente en **JSON o XML**.
* No mantiene estado (**stateless**).
* Compatible con **cualquier lenguaje de programación**.

**Ejemplo de implementación en Node.js (Express.js)**

const express = require('express');

const app = express();

app.use(express.json());

app.get('/saludo', (req, res) => {

res.json({ mensaje: "Hola, bienvenido a la API REST" });

});

app.listen(3000, () => console.log("Servidor REST corriendo en puerto 3000"));

Cliente (JavaScript con Fetch API):

fetch("http://localhost:3000/saludo")

.then(res => res.json())

.then(data => console.log(data.mensaje));

**6. Conclusión**

Las APIs juegan un papel clave en la comunicación entre aplicaciones. **SOAP** es robusto pero más pesado, **RPC** es eficiente para llamados de procedimientos, **WebSocket** es ideal para comunicación en tiempo real, y **REST** es el estándar más flexible y utilizado en la web actual.

**7. Bibliografía**

* Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures.
* W3C. (2024). Simple Object Access Protocol (SOAP) 1.2.
* MDN Web Docs. (2024). WebSockets.