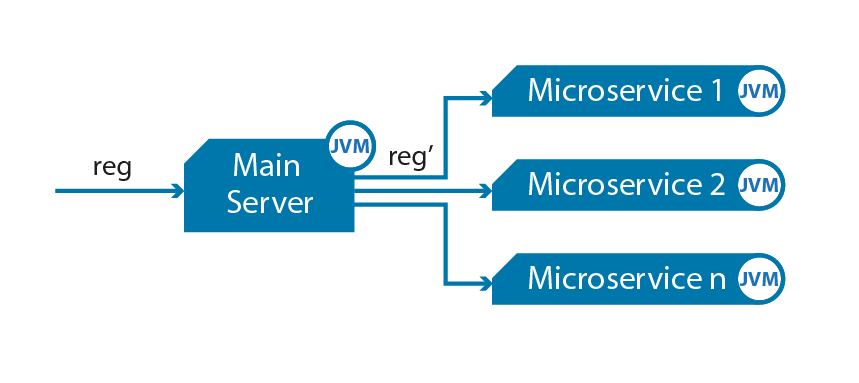
**Clase 11. API Testing**

# Pruebas de servicios

## Microservicios

Son un conjunto de servicios con funciones concretas e independientes que trabajan de forma articulada para brindar una o múltiples funcionalidades a una aplicación.



### ¿Qué son los microservicios?

Consiste en una forma para desarrollar una aplicación, es decir, un conjunto de pequeños servicios y cada uno ejecuta su propio proceso y se comunica con mecanismos ligeros, a menudo una API a través de HTTP/HTTPS.

### Tipo de prueba API

API significa Application Programming Interfaces, o en castellano Interfaz de programación de aplicaciones, y es un conjunto de definiciones y protocolos que se usa para desarrollar e integrar el software, permitiendo la comunicación entre dos aplicaciones a través de un conjunto de reglas (o endpoints).

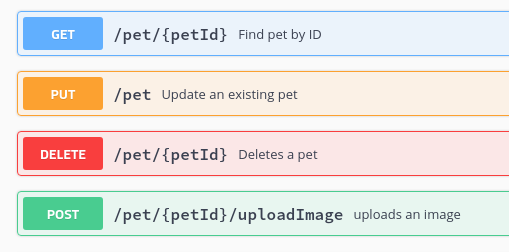
### Endpoints

Los endpoints son las URLs de una API dentro del backend que realizan llamados para generar una acción. Desde un endpoint podremos llamar a una base de datos, a un servidor de imágenes, o incluso a otro endpoint, entre otros.

### CRUD

Alude a Copy, Read, Update, Delete y son las distintas acciones que realizará un endpoint, pero tendrán los nombres GET, PUT, POST y DELETE.

Estas cuatro acciones son de las más usadas.



Referencia: Cody Reichert | [Assertible](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)

### Respuesta endpoint

La respuesta puede ser una confirmación de que algo se modificó o una información solicitada.

Junto con esto, tenemos distintos tipos de códigos de respuesta que nos indican si funciono, o no, y en este caso, las razones por las que falló el llamado.

#### 200

Solicitud aceptada; la respuesta contiene el resultado. Este es un código de respuesta general a cualquier solicitud.

* En las solicitudes GET, el recurso o datos solicitados están en el cuerpo de la respuesta.
* En las solicitudes PUT o DELETE, la solicitud fue satisfactoria y la información acerca del resultado se puede encontrar en el cuerpo de la respuesta.

#### 201

Las operaciones PUT o POST devuelven este código de respuesta e indica que se ha creado un recurso de forma satisfactoria.

El cuerpo de la respuesta podría, por ejemplo, contener información acerca de un nuevo recurso o información de validación.

#### 400

La solicitud no fue válida.

Este código se devuelve cuando el servidor ha intentado procesar la solicitud, pero algún aspecto de la misma no es válido.

La información acerca de la solicitud se proporciona en el cuerpo de la respuesta e incluye un código de error y un mensaje de error.

#### 401

El servidor de aplicaciones devuelve este código de respuesta cuando está habilitada la seguridad y falta la información de autorización en la solicitud

#### 404

Indica que el recurso de destino no existe. Esto podría deberse a que la URL no está bien formada o a que se ha suprimido el recurso.

#### 500

Se ha producido un error interno en el servidor.

Esto podría indicar un problema con la solicitud o un problema en el código del lado del servidor. Se puede encontrar información acerca del error en el cuerpo de respuesta.

## ¿Por qué probamos APIs?

### Funciones de APIs

Las APIs cumplen el papel fundamental de traer la información y mostrarla en la pantalla. Ésto la vuelve muy importante, ya que si fallan, serán el resultado de lo que veremos en pantalla.

### Implementación de APIs

La implementación de las pruebas, se llevarán adelante de dos formas particulares:

Mediante herramientas de consulta y modificación de APIs

A través del uso de HTTP Debug Proxy

### Pruebas de APIs

Hay una serie de herramientas que sirven específicamente para hacer pruebas de APIs, más adelante veremos cómo se comportan las mismas:

* Si las respuestas son como esperamos
* Cómo hacen distintos tipos de pruebas
* Si encontramos brechas de seguridad.

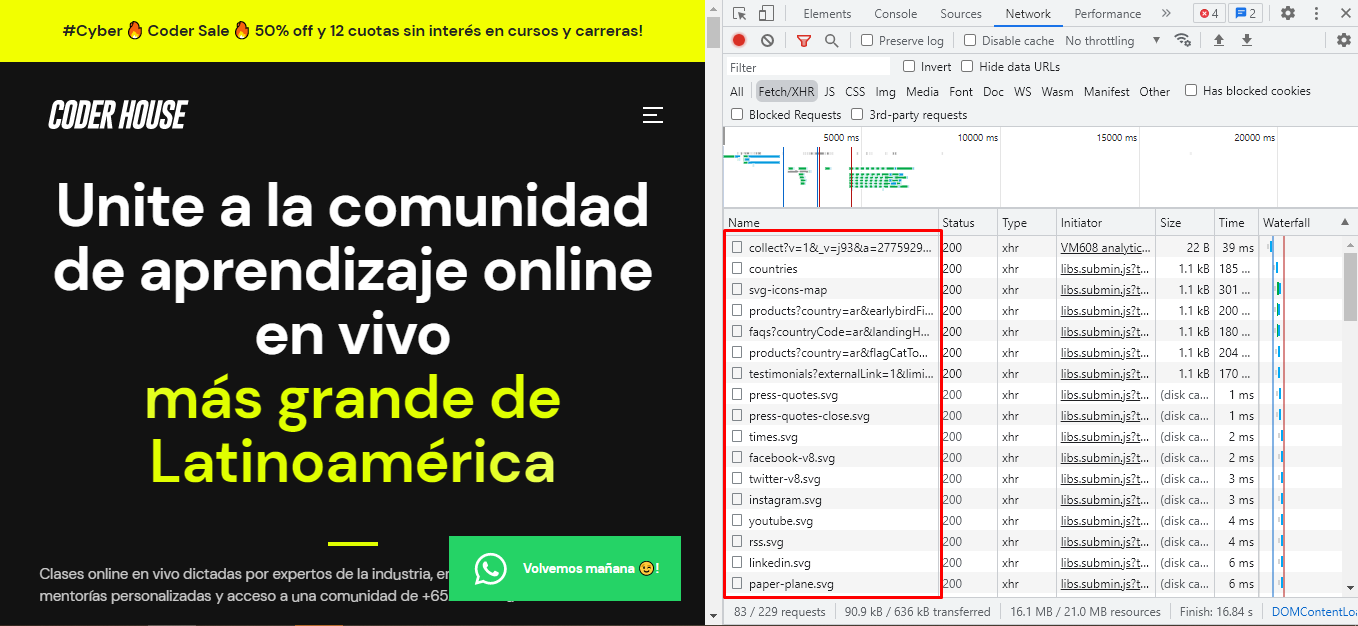
### HTTP Debug Proxy

Estas herramientas son las que interceptan toda la información que sale de las aplicaciones y nos permite modificar los endpoints para ir viendo distintos comportamientos acorde a la información que fuimos configurando.

### Visor llamados

Desde la consola de cada explorador, podremos observar los llamados que hace el sistema para traer toda la información que vemos en pantalla.

#### Ejemplo visor llamados



# SOAP & REST

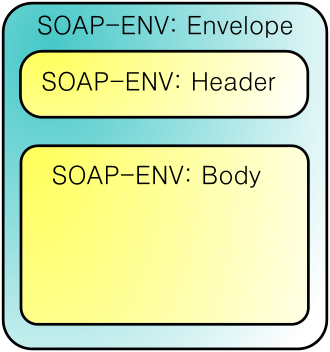
## ¿Qué son?

REST y SOAP son dos formas de transmisión de datos online. Ambas definen cómo programar una API

## SOAP

### ¿Qué es SOAP?

SOAP (Simple Object Access Protocol) es un protocolo estándar diseñado originalmente para que dos aplicaciones construidas en diferentes lenguajes y en diferentes plataformas se pueden comunicar. Al ser un protocolo, impone reglas de armado que lo hacen complejo, pesado y de carga por la cantidad de información que lo compone.



Referencia: [Diferencias entre REST y SOAP | redhat.com](https://www.redhat.com/es/topics/integration/whats-the-difference-between-soap-rest)

### Características de SOAP

Al ser un protocolo armado, provee características que son propiedades asegurables para transacciones confiables en bases de datos, tales como:

* Seguridad
* Atomicidad
* Consistencia
* Aislamiento
* Durabilidad

### Web services comunes

* Web services security (WS-security): Estandariza como los mensajes son seguros y transferidos a través de identificadores únicos llamados tokens.
* WS-ReliableMessaging: Estandariza el manejo de errores entre los mensajes transferidos entre infraestructura de IT poco confiables.
* Web services addressing (WS-addressing): Empaqueta la información de enrutamiento como metadatos dentro del header.
* Web services description language (WSDL): Describe qué hace un webservice y donde arranca y termina ese servicio.

### Info extra

Los llamados enviados de SOAP pueden ser en distintas capas de aplicaciones como:



La respuesta del mismo será en documentos XML que es un lenguaje tanto humano como de máquina.

### Historia de SOAP

En 1998 SOAP fue diseñado como un protocolo de acceso a objetos por Dave Winer, Don Box, Bob Atkinson y Mohsen Al-Ghosein por Microsoft.

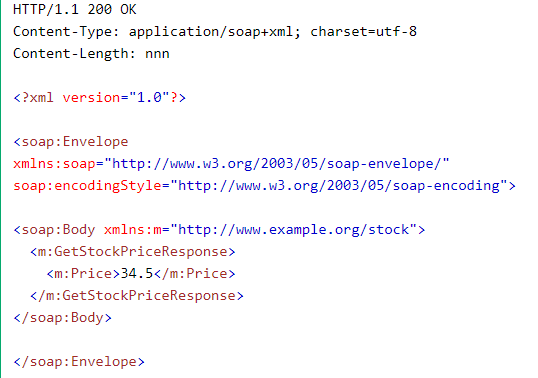
La especificación SOAP actualmente es mantenida por el XML Protocol Working Group del World Wide Web Consortium.

Referencia: [Simple Object Access Protocol | Wikipedia](https://es.wikipedia.org/wiki/Simple_Object_Access_Protocol)

### Ejemplo llamado

Se está llamando a “IBM” para saber el valor de la bolsa

### Ejemplo respuesta

El valor es 34.4

### Video ejemplo

Fuente:[enlace](http://www.youtube.com/watch?v=gmqjAennHbc)

## REST

### ¿Qué es REST?

REST (Representational State Transfer) es un grupo de principios de arquitectura para el desarrollo de APIs para servicios web, como así también de aplicaciones móviles de conectividad ligera.

Al ser una guía, se deja la implementación a los desarrolladores.

### Características de REST

Cuando se hace un llamado a un REST API, habitualmente es a través de HTTP. Luego, la respuesta puede volver en varios formatos, como:

* HTML,
* XML,
* texto plano
* JSON (suele ser el más usado ya que puede ser leído por cualquier lenguaje de programación, humano o computadora).

De esta forma, REST es más flexible y fácil de armar.

### Guía de arquitectura

Una aplicación de tipo REST debe contener estas 6 características:

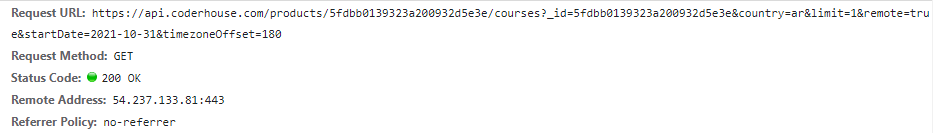
1. Una arquitectura cliente-servidor compuesta por clientes, servidores y recursos.
2. Comunicación cliente-servidor sin estado, lo que significa que no se almacena contenido del cliente en el servidor entre solicitudes pero sí en la sesión del cliente.
3. Datos que se pueden almacenar en caché para eliminar la necesidad de algunas interacciones cliente-servidor.
4. Una interfaz uniforme entre los componentes para que la información se transfiera de forma estandarizada.
5. Una restricción del sistema en capas, donde las interacciones cliente-servidor pueden ser mediadas por capas jerárquicas.
6. Código a pedido, lo que permite a los servidores ampliar la funcionalidad de un cliente mediante la transferencia de código ejecutable.

### Historia de REST

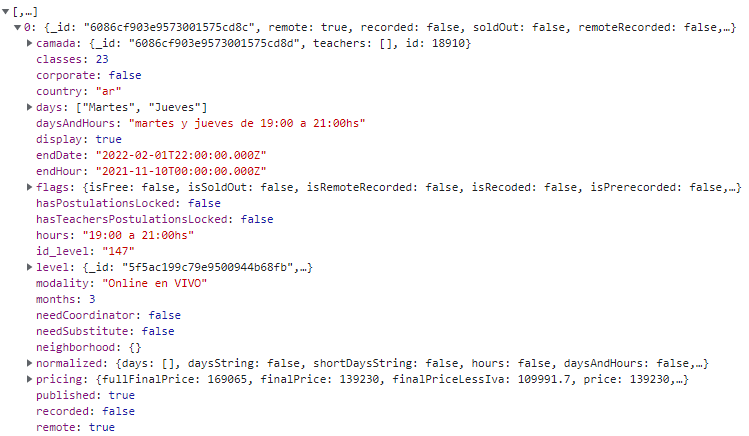
Los grupos W3C y del IETF comenzaron a trabajar en la creación de descripciones formales de los tres estándares principales de la Web: URI, HTTP y HTML.

Roy Fielding estuvo involucrado en la generación de estos estándares y durante los siguientes seis años desarrolló el estilo arquitectónico REST. Fielding defendió REST en su PhD en los 2000, siendo ese el comienzo de su uso

### Ejemplo llamado



### Ejemplo respuesta



# SOAP vs REST

### SOAP

* Muchos sistemas legendarios utilizan SOAP, al ser más antiguo.
* SOAP es un protocolo específicamente trabajando en mensajería XML.
* SOAP ofrece seguridad y cumplimiento de transacciones, que se adapta a las necesidades de algunas empresas, pero es más pesado.

#### { SOAP }

* Únicamente XML
* Más lento
* Más antiguo
* Más complejo
* Es protocolo
* Más segura

### REST

* En la actualidad, REST pasó a ser más usado debido a su rápida alternativa.
* REST es una guía flexible de implementación.
* REST es ligero, siendo ideales para los nuevos contextos de internet como tecnología IoT, aplicaciones mobile y computación serverless.

#### { REST }

* Múltiples lenguajes
* Más rápido
* Más popular
* Más sencillo
* No es protocolo
* Menos segura

# Herramientas de API Testing

## Introducción

La gran mayoría de herramientas de pruebas de API sirven tanto para SOAP como para REST, por ende debemos encontrar la que más nos guste. A continuación, se explican cuáles son las más usadas.

## Soap UI

Es una herramienta más usada para las pruebas de SOAP. Se caracteriza por ser de código abierto, desarrollado por SmartBear.

Provee muchas herramientas y vistas para hacer el trabajo más sencillo.

Referencia: [API Testing Features | SoapUI](https://www.soapui.org/)

### Vídeo Soap UI

[Enlace](https://youtu.be/rfIi9hdT-vk)

* Instalación 0.28 a 2.15
* Test de funcionalidad 2.18 a 4.34
* test driven development 4.43 a 6.30
* Test de carga 6.31 a 9.30

## Postman

Es la herramienta de prueba de APIs más usada de todas debido a su sencillez y usabilidad. Contiene herramientas extras que veremos posteriormente.

Referencia: [Referencia Postman API](http://www.postman.com)

### Info extra de Postman

#### RUNNER & MONITOR

Dentro de todas las herramientas que trae Postman, hay dos que son claves: el runner y monitor.

Runner corre una cantidad de llamados en orden, la cantidad de veces marcadas.

Monitor genera llamadas automáticas en distintos momentos, acorde a lo que dejemos marcado.

## SWAGGER

* Una API pierde su sentido si no es accesible y si no tenemos una documentación que nos ayude a entenderla.
* Uno de los mayores problemas de las APIs es que en muchos casos, la documentación que les acompaña es inútil. Swagger nace con la intención de solucionar este problema. Su objetivo es estandarizar el vocabulario que utilizan las APIs. Es el diccionario API.
* Swagger es una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs
* De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger nos ayuda a crear documentación que todo el mundo entienda.
* Utilizando el Swagger UI para exponer la documentación de nuestra API, podemos ahorrarnos mucho tiempo

## Otras herramientas

Existen otras herramientas para investigar, entre las cuales se distingue:

* Insomnia REST Client
* HttpMaster
* RESTClient
* HttpRequester
* RESTer
* Hoppscotch

## Glosario

**API:**  Interfaz de Programación de Aplicaciones.

**Microservicios:** conjunto de pequeños servicios en donde cada uno va a ejecutar su propio proceso y se va a comunicar con una API.

**JSON:** JavaScript Object Notation. El formato JSON se utiliza para estructurar datos en forma de texto y permite el intercambio de información entre aplicaciones de manera sencilla, liviana y rápida

**SOAP**: el protocolo simple de acceso a objetos (SOAP) es un protocolo oficial, cuyo mantenimiento está a cargo del Consorcio World Wide Web.

**REST:** La transferencia de estado representacional (REST) es un conjunto de principios arquitectónicos.

**Endpoint:** los endpoints son las URLs de un API o un backend que responden a una petición.

#### ¿QUIERES SABER MÁS? TE DEJAMOS MATERIAL AMPLIADO DE LA CLASE

* [Assertible - Información Extra Protocolo HTTP](https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them)
* [W3 School - Información extra de SOAP](https://www.w3schools.com/xml/xml_soap.asp)
* [Wikipedia - REST](https://en.wikipedia.org/wiki/Representational_state_transfer#History)
* [Lista de APIs para probar](https://ichi.pro/es/una-lista-seleccionada-de-100-api-publicas-geniales-y-divertidas-para-inspirar-su-proximo-proyecto-8109114517939)

#DEMOCRATIZANDOLAEDUCACIÓN