

QUÉ ES UN API

Interfaz de programación de aplicaciones

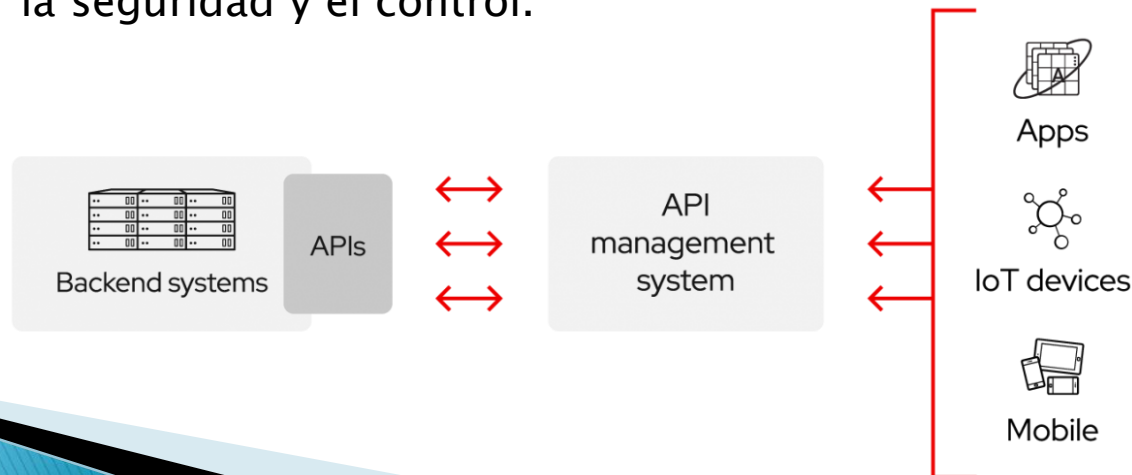
Elías Pretel

Juan Silva

Alejandra Rodríguez

API – Características

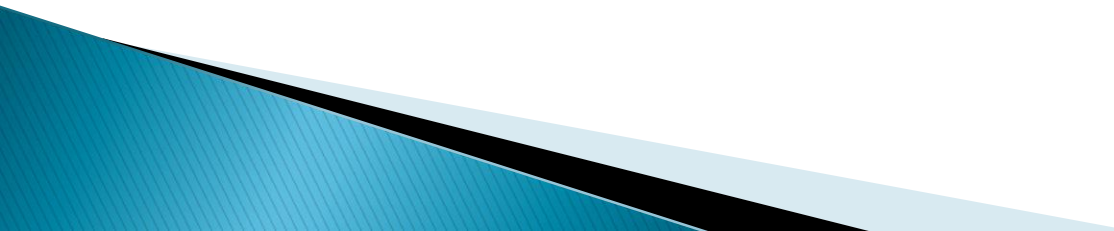
- Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados.
- Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación.
- Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos.
- Las API le permiten habilitar el acceso a sus recursos, al mismo tiempo, mantener la seguridad y el control.



APIS LOCALES

- Cámara y micrófono, para acceder al audio y video del dispositivo.
- Giroscopio, utilizado para VR y saber la posición del dispositivo.
- Bluetooth

APIS REMOTAS

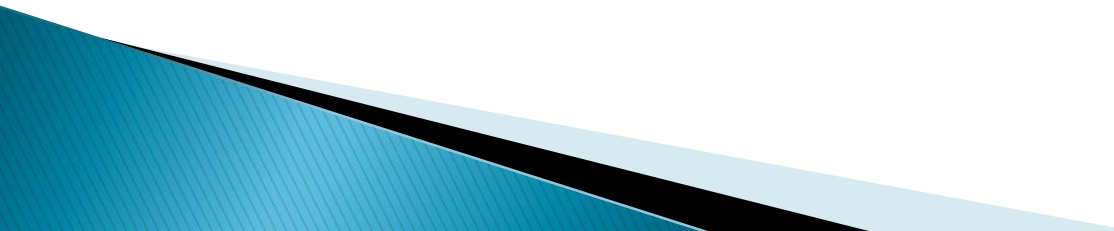
- Google Maps
 - Api de spotify
 - Firebase Notifications; la utilizan las mayoría de las apps para las notificaciones push y demás.
- 

REST – ACERCAMIENTO

REST cambió por completo la ingeniería de software a partir del 2000.

Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por Roy Fielding, el padre de la especificación HTTP y uno de los referentes internacionales en todo lo relacionado con la Arquitectura de Redes, en su disertación 'Architectural Styles and the Design of Network-based Software Architectures'.

En el campo de las APIs, REST (Representational State Transfer– Transferencia de Estado Representacional) es, al día de hoy, el alfa y omega del desarrollo de servicios de aplicaciones.



API REST

REST se entiende como protocolo de intercambio y manipulación de datos en los servicios de internet que cambió por completo el desarrollo de software a partir de 2000. Ya casi toda empresa o aplicación dispone de una API REST para creación de negocio.

CARACTERÍSTICAS

- **Protocolo cliente/servidor sin estado:** cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché.
- Se configura lo que se conoce como protocolo **cliente-caché-servidor sin estado**: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable.
- **Interfaz uniforme:** Para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con la información.

VENTAJAS

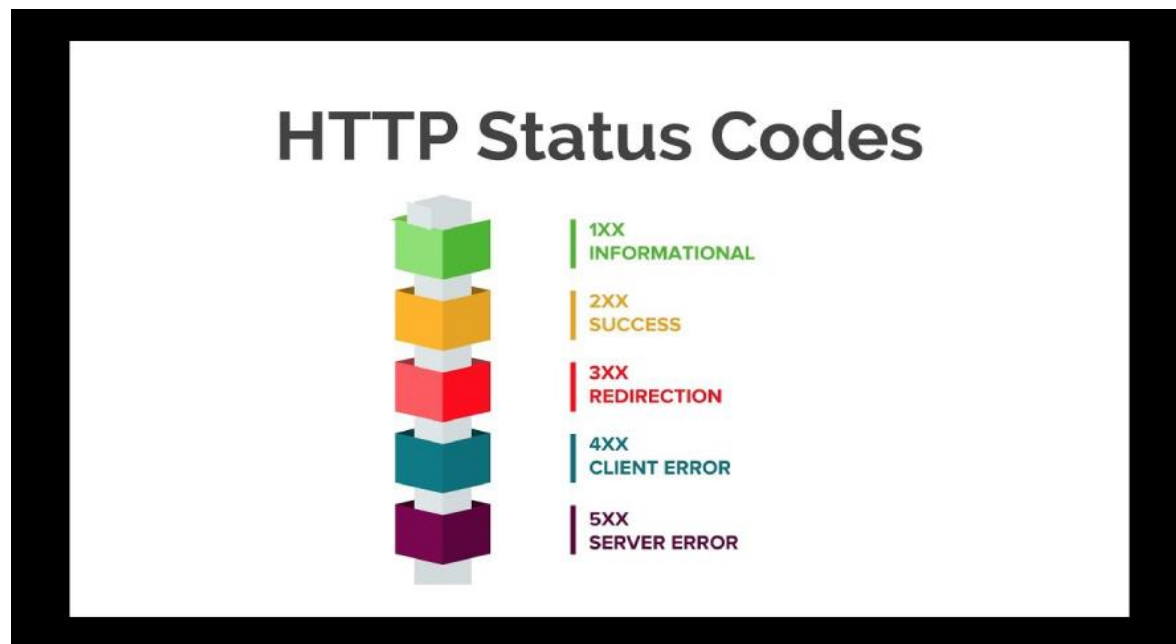
- **Separación entre el cliente y el servidor:** el protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos.
- **Visibilidad, fiabilidad y escalabilidad:** La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta.
- ▶ **La API REST siempre es independiente del tipo de plataformas o lenguajes:** Siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando.
- ▶ Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente **XML** o **JSON**.

<https://www.coldings.com/eaglecustomer/Home/Index?prm=2>

<https://www.coldings.com/eaglecustomer/Home/Index/2>

Parámetros por URL

HTTP Status Code



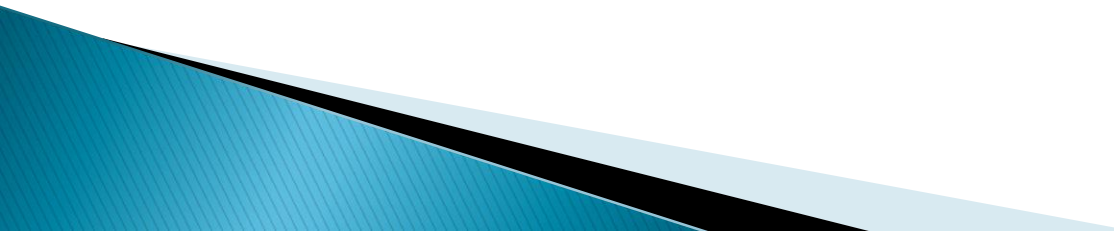
<https://developer.mozilla.org/es/docs/Web/HTTP/Status>

JSON (JavaScript Object Notation o Notación de Objetos)

- ▶ Es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.
- ▶ **JSON** es un formato de texto completamente independiente de lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores, entre ellos: C, C++, C#, Java, JavaScript, Perl, Python; entre otros. Estas propiedades hacen de **JSON** un formato de intercambio de datos ideal para usar con API REST o AJAX. A menudo se usa en lugar de XML, debido a su estructura ligera y compacta.

```
[
  {
    "description": "quarter",
    "mode": "REQUIRED",
    "name": "qtr",
    "type": "STRING"
  },
  {
    "description": "sales representative",
    "mode": "NULLABLE",
    "name": "rep",
    "type": "STRING"
  },
  {
    "description": "total sales",
    "mode": "NULLABLE",
    "name": "sales",
    "type": "INTEGER"
  }
]
```

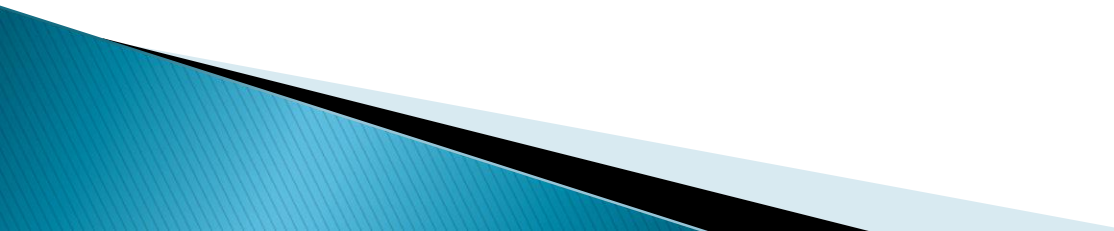

JSON – Características

- JSON es solo un formato de datos.
 - Requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas.
 - Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione.
 - Puede tomar la forma de cualquier tipo de datos que sea válido para ser incluido en un JSON, no solo arreglos u objetos. Así, por ejemplo, una cadena o un número único podrían ser objetos JSON válidos.
- 

JSON – Ventajas

- Es auto descriptivo y fácil de entender. Un archivo de configuración en JSON son mucho mas sencillos de entender, pero no se pueden tipar.
- Su sencillez le ha permitido posicionarse como alternativa a XML.
- Es más rápido en cualquier navegador.
- Es más fácil de leer que XML.
- Es más ligero (bytes) en las transmisiones.
- Se parsea más rápido.
- Velocidad de procesamiento alta.
- Puede ser entendido de forma nativa por los analizadores de JavaScript.

JSON – Desventajas

- Algunos desarrolladores encuentran su escueta notación algo confusa.
 - No cuenta con una característica que posee XML: “extensibilidad”.
 - No soporta grandes cargas, solo datos comunes.
 - Para la seguridad requiere de mecanismos externos como expresiones regulares.
- 

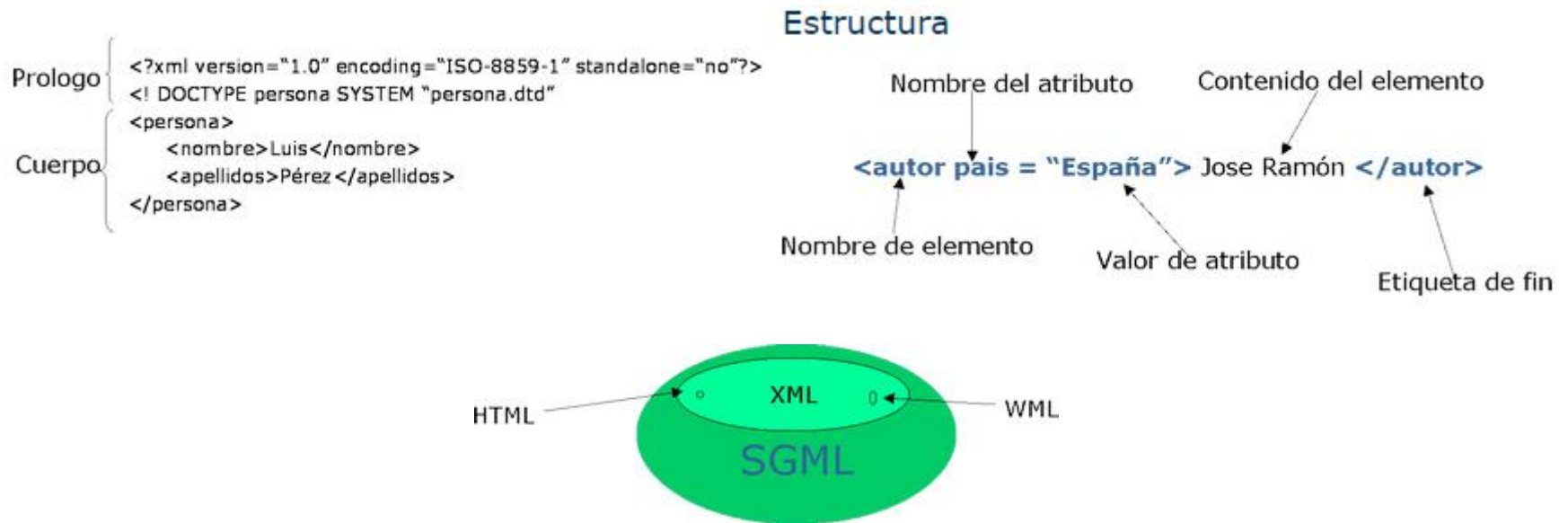
XML

- Especificación para diseñar lenguajes de marcado, que permite definir etiquetas personalizadas para descripción y organización de datos.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Libro">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Título" type="xsd:string"/>
        <xsd:element name="Autores" type="xsd:string" maxOccurs="10"/>
        <xsd:element name="Editorial" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="precio" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Estructura de un documento XML

- Un documento XML está formado por **datos de caracteres y marcado**, el marcado lo forman las etiquetas:

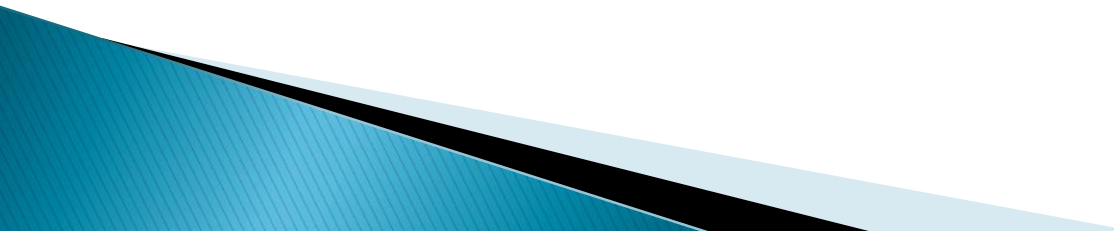


- XML no es ,como su nombre puede sugerir, un lenguaje de marcado.
- XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados.

XML– Ventajas

- Tiene un formato estructurado y fácil de comprender.
- Separa radicalmente la información o el contenido de su presentación o formato.
- Está diseñado para ser utilizado en cualquier lenguaje o alfabeto.
- Su análisis sintáctico es fácil debido a las estrictas reglas que rigen la composición de un documento.
- Tiene soporte a cualquier tipo de datos.
- Se pueden definir estructuras complejas y reutilizables.

XML– Desventajas

- ▶ El formato es sumamente estricto.
 - ▶ Lleva más tiempo procesarlo.
 - ▶ Complejidad de analizador (parser).
 - ▶ Un error en cualquier parte del formato puede hacer que todo el documento sea inválido.
- 



SoapUI



 **Progress**[®]
Telerik[®] Fiddler[™]



POSTMAN