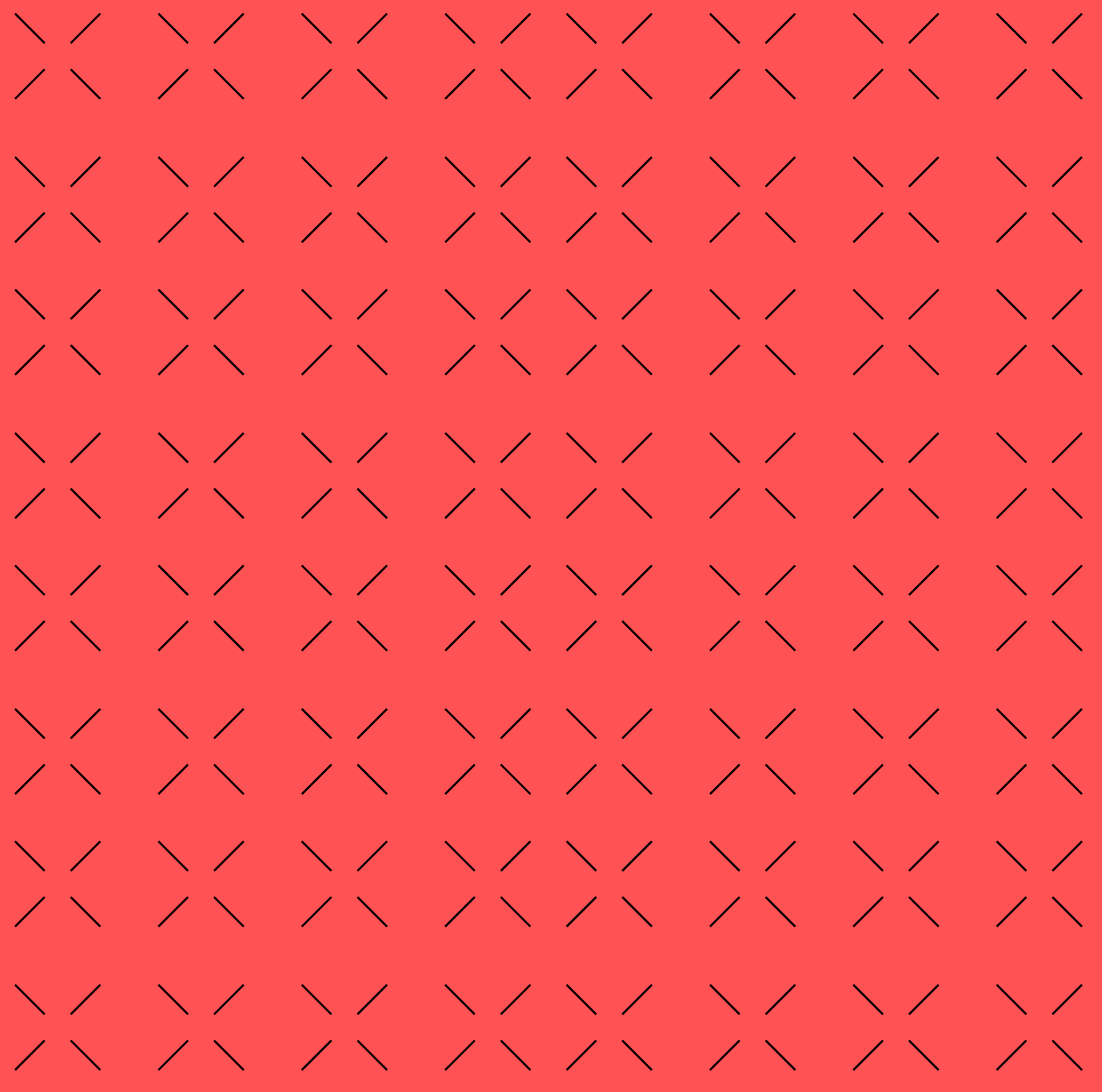


# Unidad 3.1

## Generación de servicios en red

### APIRest



# Índice

1 Servicios Web.....	4
1.1 API (Application Programming Interface).....	4
1.2 Comparación entre Desarrollo de Páginas Web y Uso de API.....	6
1.2.1 Desarrollo de Páginas Web.....	6
1.2.2 Uso de API en Programas:.....	7
1.2.3 Consideraciones Generales.....	9
2 Tipos de API.....	10
3 Los protocolos que usan las API.....	11
3.1 APIs REST.....	11
3.2 APIs SOAP.....	13
3.3 APIs RPC.....	14
3.4 GraphQL APIs.....	15
4 API REST.....	17
4.1 Uso de API de REST.....	17
4.1.1 {JSON} Placeholder.....	17
4.1.2 Openweathermap.....	18
4.1.3 OpenAI.....	18
5 Referencias.....	19

## Licencia



### **Reconocimiento – NoComercial – CompartirIgual (by-nc-sa):**

No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

## 1 Servicios Web

En el tejido de la interconexión digital, los servicios web bajo el protocolo HTTP han permitido la transferencia eficiente de datos entre aplicaciones a través de la web. Sin embargo, esta eficacia no ha estado exenta de desafíos inherentes, y es aquí donde entran en juego las API (Interfaz de Programación de Aplicaciones) para superar estos obstáculos.

A medida que las aplicaciones crecen en complejidad y diversidad, la necesidad de una comunicación más eficiente y estandarizada se vuelve apremiante. Los servicios web, aunque valiosos, a menudo arrastran consigo desventajas, como la falta de una interfaz clara y la complejidad asociada con la implementación de protocolos específicos.

Es en este contexto que las API emergen como una solución efectiva. Actuando como mediadoras entre sistemas dispares, las API proporcionan una interfaz estandarizada que abstrae la complejidad subyacente de la comunicación entre aplicaciones. Al hacerlo, permiten una mayor flexibilidad, colaboración y eficiencia en el desarrollo de software.

### 1.1 API (Application Programming Interface)

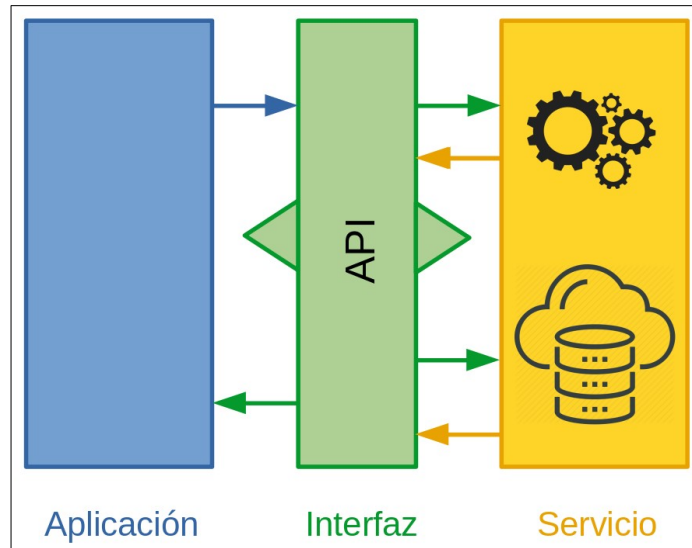
Una API es un conjunto de reglas y definiciones que permite que diferentes software se comuniquen entre sí. En términos simples, una API especifica cómo los componentes de software deben interactuar.

#### Puntos clave para entender las APIs:

- **Interfaz para la Comunicación:**
  - Una API actúa como una interfaz que permite a dos aplicaciones diferentes comunicarse entre sí.
  - Define los métodos y estructuras de datos que las aplicaciones pueden utilizar para solicitar y compartir información. Se trata de un conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios internos o remotos (a través de conexión de red).
- **Abstracción:**
  - Proporciona una capa de abstracción, permitiendo que los desarrolladores utilicen funciones de un software sin necesidad de entender la implementación interna. Existen pues ciertas rutinas basadas en cierto software o hardware sin que el desarrollador web tenga acceder al código fuente o comprender el mecanismo de trabajo interno de la API. Los desarrolladores interactúan únicamente con la API, con las funcionalidades que ofrece.

- **Estandarización:**

- Las APIs estandarizan la forma en que las aplicaciones se comunican, lo que facilita la integración de sistemas y servicios.
- Al seguir un conjunto común de reglas, diferentes desarrolladores y equipos pueden construir sobre una API sin tener que preocuparse por los detalles internos.



## 1.2 Comparación entre Desarrollo de Páginas Web y Uso de API

La elección entre desarrollo de páginas web y uso de API depende de factores como la complejidad del proyecto, las necesidades de interactividad y la preferencia del equipo de desarrollo. Cada enfoque tiene sus ventajas y desventajas, y la decisión debe basarse en los requisitos específicos del proyecto y los objetivos comerciales.

### 1.2.1 Desarrollo de Páginas Web

- **Ventajas:**

- **Control Completo del Frontend:**

- Desarrollar páginas web permite un control total sobre la interfaz de usuario y la experiencia del usuario. Los desarrolladores pueden diseñar interfaces personalizadas según las necesidades del proyecto.

- **Interactividad Directa:**

- Las páginas web permiten una interactividad directa con los usuarios. Los eventos del lado del cliente, como clics y desplazamientos, pueden manejarse fácilmente para mejorar la experiencia del usuario.

- **Flexibilidad de Diseño:**

- Los desarrolladores tienen la flexibilidad de diseñar y estructurar la interfaz visual según sus preferencias. Pueden incorporar estilos y animaciones personalizadas para mejorar la estética.

- **SEO (Optimización para Motores de Búsqueda):**

- Las páginas web tradicionales son indexadas fácilmente por los motores de búsqueda. Las técnicas de SEO se pueden aplicar directamente para mejorar la visibilidad en los resultados de búsqueda.

- **Desventajas:**

- **Mayor Complejidad en la Implementación:**

- Desarrollar aplicaciones web completas puede ser más complejo y llevar más tiempo en comparación con el uso de API. Se necesita manejar tanto el frontend como el backend.

- **Tiempo de Carga Prolongado:**

- Páginas web grandes pueden tener tiempos de carga más largos, especialmente si contienen muchos recursos multimedia. Esto puede afectar negativamente la experiencia del usuario.

- **Requiere Conocimientos Variados:**

- El desarrollo web completo requiere habilidades en varios lenguajes y tecnologías, como HTML, CSS, JavaScript, y un backend (p. ej., Node.js, Django, Flask).

## 1.2.2      **Uso de API en Programas:**

- **Ventajas:**

- **Separación de Responsabilidades:**

- La utilización de API permite la separación clara de responsabilidades. El frontend y el backend pueden desarrollarse de forma independiente, facilitando la colaboración en proyectos grandes.

- **Eficiencia en la Transferencia de Datos:**

- Las API, especialmente las RESTful, son eficientes para la transferencia de datos a través de la web. Utilizan formatos ligeros como JSON para minimizar el ancho de banda y mejorar el rendimiento.

- **Reutilización de Recursos:**

- Las API permiten la reutilización de recursos y datos en múltiples aplicaciones. Un backend bien diseñado puede ser aprovechado por diferentes interfaces de usuario.

- **Menor Carga en el Cliente:**

- Con API, gran parte de la lógica y el procesamiento de datos se realiza en el servidor, aliviando la carga en el cliente y mejorando la velocidad de respuesta.

- **Desventajas:**

- **Dependencia de la Disponibilidad del Servicio:**


- La dependencia de una API significa que la disponibilidad del servicio es crucial. Si la API está caída o experimenta problemas, afectará directamente a las aplicaciones que la utilizan.

- **Limitaciones en la Interactividad:**

- La interactividad directa puede ser limitada en comparación con el desarrollo de páginas web tradicionales. Algunas acciones pueden requerir solicitudes adicionales a la API.

## ○ Menos Control sobre la Interfaz de Usuario:

- Al utilizar una API, los desarrolladores tienen menos control directo sobre la interfaz de usuario. La personalización visual puede estar restringida a lo que la API proporciona.



**{ APIs }**

[Overview](#)
[Generate API Key](#)
[Authentication](#)
[Recover API Key](#)
[Browse APIs](#)

One of the most popular websites at NASA is the [Astronomy Picture of the Day](#). In fact, this website is one of the [most popular websites](#) across all federal agencies. It has the popular appeal of a Justin Bieber video. This endpoint structures the APOD imagery and associated metadata so that it can be repurposed for other applications. In addition, if the `concept_tags` parameter is set to `True`, then keywords derived from the image explanation are returned. These keywords could be used as auto-generated hashtags for twitter or instagram feeds; but generally help with discoverability of relevant imagery.

The full documentation for this API can be found in the [APOD API Github repository](#).

**Example image:**



Credit: Hubble, NASA, ESA, & D. O. Wang (U. Mass./Amherst);  
Sperner, T. A. A. J. & V. Stetson (SDSS/Abell)

**HTTP Request**

GET `https://api.nasa.gov/planetary/apod`

`concept_tags` are now disabled in this service. Also, an optional return parameter `copyright` is returned if the image is not public domain.

**Query Parameters**

Parameter	Type	Default	Description
date	YYYY-MM-DD	today	The date of the APOD image to retrieve
start_date	YYYY-MM-DD	none	The start of a date range, when requesting date for a range of dates. Cannot be used with <code>date</code> .
end_date	YYYY-MM-DD	today	The end of the date range, when used with <code>start_date</code> .
count	int	none	If this is specified then <code>count</code> randomly chosen images will be returned. Cannot be used with <code>date</code> or <code>start_date</code> and <code>end_date</code> .
thumbs	bool	False	Return the URL of video thumbnail. If an APOD is not a video, this parameter is ignored.
api_key	string	DEMO_KEY	api.nasa.gov key for expanded usage

By NASA Open Innovation Team - <https://api.nasa.gov/>, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=124347117>



### 1.2.3 Consideraciones Generales

- **Escenario de Uso:**
  - El enfoque a elegir depende del escenario de uso. Si se requiere una aplicación altamente interactiva y personalizada, el desarrollo web completo puede ser más apropiado. Para aplicaciones que se centran en la eficiencia y la reutilización de datos, el uso de API puede ser más beneficioso.
- **Complejidad del Proyecto:**
  - Proyectos más complejos pueden beneficiarse de la separación de frontend y backend que ofrecen las API. Para proyectos más simples, el desarrollo web completo puede ser más directo.
- **Ecosistema Tecnológico:**
  - La elección entre desarrollo web y uso de API también depende del ecosistema tecnológico preferido y de las habilidades del equipo de desarrollo.

## 2 Tipos de API

Se distinguen cuatro tipos de API comúnmente empleadas en servicios web: públicas, de socios, internas y compuestas. En este contexto, el "tipo" de API indica el alcance previsto de su uso.

- **APIs Públicas:** Una API pública está abierta y disponible para cualquier desarrollador o negocio externo. Empresas que buscan compartir sus aplicaciones y datos con otras empresas desarrollan y ofrecen APIs públicas. También conocidas como APIs abiertas o externas, generalmente involucran autenticación y autorización moderadas. Además, las empresas podrían buscar monetizar la API mediante un costo por cada llamada realizada.
- **APIs de Socios:** Las APIs de socios solo están disponibles para desarrolladores externos o consumidores de API específicamente seleccionados y autorizados. Estas facilitan actividades comerciales entre empresas, como compartir datos de clientes con empresas externas de CRM. Los socios tienen derechos y licencias claros, y estas APIs suelen incorporar mecanismos sólidos de autenticación y autorización. A menudo, las empresas no buscan monetizar directamente estas APIs, ya que los socios son remunerados por sus servicios en lugar del uso de la API.
- **APIs Internas:** Las APIs internas o privadas están destinadas únicamente para uso interno, conectando sistemas y datos dentro de la empresa. Por ejemplo, una API interna podría conectar sistemas de nómina y recursos humanos. Estas APIs suelen presentar niveles de seguridad y autenticación débiles o nulos, ya que se asume que están destinadas para uso interno y que la seguridad se maneja mediante otras políticas. Sin embargo, esta tendencia está cambiando debido a una mayor conciencia de amenazas y demandas de cumplimiento normativo.
- **APIs Compuestas:** Las APIs compuestas combinan dos o más APIs para crear secuencias de operaciones relacionadas o interdependientes. Estas APIs son beneficiosas para abordar comportamientos complejos o estrechamente relacionados, y en ocasiones, pueden mejorar la velocidad y el rendimiento en comparación con APIs individuales.

### 3 Los protocolos que usan las API

Entender el protocolo que utiliza una API es tan crucial como conocer su tipo, ya que el protocolo define cómo la API se conecta a Internet y cómo comunica información. La elección del protocolo influye en el diseño, construcción y mantenimiento de la API, por lo que es esencial comprender las ventajas y desventajas de cada opción.

#### 3.1 APIs REST

##### Descripción:

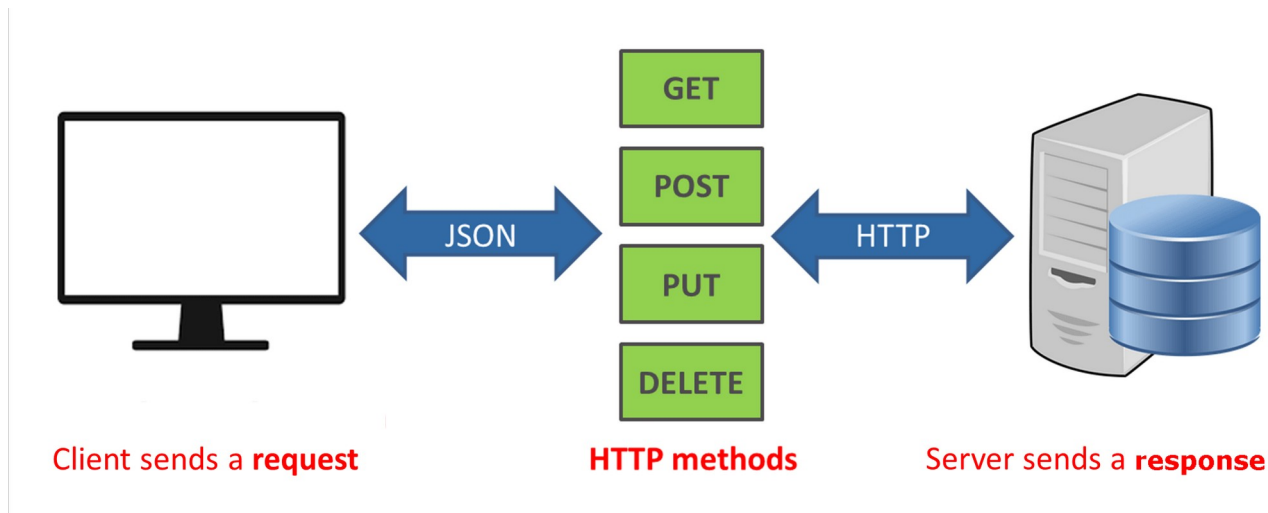
El protocolo Representational State Transfer (REST o RESTful) es ampliamente reconocido. Destaca por su simplicidad al definir rutas mediante una URL, a diferencia de otros que requieren que el desarrollador empaquete rutas con XML.

Por contra, sólo pueden transmitir información a través del protocolo HTTP, limitándose a enviar principalmente texto. Aunque las APIs REST pueden utilizar parámetros de formato para transmitir información más compleja, como imágenes o archivos de audio, estos archivos siguen siendo codificados como texto, generalmente en formato JSON o XML.

A pesar de estas limitaciones, las APIs REST se utilizan en una amplia gama de funciones, aunque requiere creatividad trabajar dentro de las restricciones de REST y HTTP.

API REST debe ajustarse a los límites de la arquitectura REST y por ello debe cumplir ciertas reglas:

- **Arquitectura cliente-servidor:** La interfaz de la API permanece en el cliente y está separada de los datos almacenados en el servidor.
- **Sin estado:** Cada solicitud hecha con la API es independiente de las demás, y las llamadas se realizan de forma independiente.
- **Almacenable en caché:** Una respuesta de la API REST puede recuperar datos en caché, pero es necesario especificar si las respuestas se pueden almacenar en caché.
- **Estratificado:** La API funciona de la misma manera, ya sea que interactúe directamente con el servidor o si hay otras capas, como un equilibrador de carga o un servicio de proxy, entre el cliente y el servidor.



BY: <https://aprendiendoarduino.wordpress.com/tag/arquitectura-api/>

### Pros de API REST:

- **Simplicidad:** El diseño simple y la facilidad de comprensión son características clave de REST, facilitando su implementación y adopción.
- **Desacoplamiento:** Cliente y servidor están desacoplados, lo que permite una mayor flexibilidad y escalabilidad en el desarrollo y la evolución del sistema.
- **Operaciones HTTP Estándar:** Utiliza métodos HTTP estándar como GET, POST, PUT y DELETE para operaciones, facilitando la integración y el uso en aplicaciones web.
- **Escalabilidad:** Al estar basado en HTTP, puede aprovechar las características de escalabilidad de la web.
- **Ampliamente Utilizado:** Es un estándar ampliamente reconocido y utilizado en diversas aplicaciones y servicios web.

### Contras de API REST:

- **Limitaciones en la Transmisión de Datos:** REST está limitado a la transmisión de datos a través de protocolos de texto, lo que puede afectar el rendimiento y la eficiencia al manejar tipos de contenido más complejos, como imágenes o archivos de audio.
- **Rigidez en la Estructura de Datos:** La estructura de los datos está predefinida por el servidor. Esto puede llevar a situaciones de overfetching, donde se recuperan más datos de los necesarios.

## 3.2 APIs SOAP

### Descripción:

El protocolo Simple Object Access Protocol (SOAP) desarrollado a finales de la década de 1990 por Microsoft e IBM, define una estructura de mensajes XML para facilitar la comunicación entre sistemas distribuidos a través de protocolos como HTTP, SMTP y otros. Aunque más verboso en comparación con REST, SOAP es más restrictivo dado que solo pueden trabajar con datos XML y tienen requisitos más rígidos para las solicitudes.

La estandarización de SOAP permite que estas APIs se comuniquen de manera más confiable con datos complejos y los entreguen a través de más canales que no solo HTTP. En general, SOAP se adapta mejor a aplicaciones más sofisticadas, donde la confiabilidad es más importante que la velocidad.

### Pros:

- **Agnóstico del Lenguaje:** SOAP es agnóstico con respecto al lenguaje de programación y puede ser utilizado con protocolos de transporte más allá de HTTP, como TCP o SMTP.
- **Manejo de Errores Estandarizado:** Incorpora un manejo de errores nativo y estandarizado, lo que facilita la identificación y resolución de problemas en las transacciones.
- **Solicitudes con Estado:** Permite solicitudes con estado, lo que significa que puede rastrear el estado de las transacciones, brindando una mayor confiabilidad en comparación con REST, que es sin estado.
- **Estándares Bien Definidos:** SOAP define claramente los estándares, incluyendo la necesidad de archivos de metadatos (WSDL), lo que hace que las interacciones sean más predecibles.
- **Seguridad:** Proporciona capas adicionales de seguridad, lo que lo hace adecuado para industrias donde la seguridad es de suma importancia, como servicios financieros y de salud.

### Contras:

- **Verbosidad:** SOAP tiende a ser más verboso en comparación con REST debido al formato XML, lo que resulta en una mayor sobrecarga de datos y mayor complejidad.
- **Menor Popularidad:** A medida que los desarrolladores buscan soluciones más livianas y simples, SOAP ha perdido popularidad en comparación con REST.

- **Complejidad de Construcción y Mantenimiento:** La construcción y el mantenimiento de código SOAP pueden ser más complejos y requerir más recursos en comparación con REST.
- **Ancho de Banda:** Las solicitudes SOAP suelen requerir más ancho de banda debido a su formato XML más pesado, lo que puede afectar el rendimiento, especialmente en entornos con restricciones de ancho de banda.

### 3.3 APIs RPC

#### Descripción:

El protocolo Remote Procedure Call (RPC) permite que las aplicaciones puedan invocar funciones o procedimientos en un sistema remoto como si estuvieran ejecutándose localmente. A diferencia de una API REST, que trabaja con recursos, una API RPC trabaja con acciones. Esto quiere decir que a diferencia de REST, que devuelve en su respuesta un fichero, RPC devuelve la confirmación de que la función se activó o un error indicando por qué no se ejecutó.

Las APIs RPC rara vez son APIs públicas, ya que activar métodos en servidores remotos no es algo que la mayoría de las empresas deseen permitir al público en general. Llamar a un servidor RPC realmente cambia el estado del servidor, por lo que va más allá de la distinción sin estado/estado de REST y SOAP. Como resultado, las APIs RPC deben tener un alto nivel de seguridad y confianza entre productores y consumidores, por lo que son más a menudo APIs privadas.

En 2015, Google introdujo un tipo de RPC llamado gRPC, que utiliza Protocol Buffers para serializar y analizar datos. gRPC está construido sobre HTTP/2, una actualización de HTTP introducida en 2015. Por motivos de integración, si quieres trabajar con gRPC en el navegador, necesitará un servicio de proxy como Envoy, lo que puede hacerlo menos útil para sitios web o aplicaciones basadas en el navegador.

#### Pros:

- **Simplicidad:** RPC se destaca por su simplicidad. Utiliza métodos como GET para obtener información y POST para realizar otras operaciones, lo que facilita su implementación y uso.
- **Excelente Rendimiento:** Debido a su simplicidad y enfoque en funciones específicas, RPC puede ofrecer un rendimiento excepcional, especialmente para cargas útiles ligeras.
- **Configurabilidad Infinita:** La capacidad de definir cualquier tipo de función lo hace infinitamente configurable según las necesidades específicas de la aplicación.

Agregar nuevas funciones es sencillo en un entorno RPC, lo que permite una escalabilidad y flexibilidad notables.

- **Gestión de Microservicios:** RPC se utiliza comúnmente en arquitecturas de microservicios para gestionar llamadas remotas entre servicios distribuidos.

#### Contras:

- **Acoplamiento Estrecho:** RPC tiende a estar más acoplado al sistema subyacente, lo que puede reducir su reutilización en diferentes contextos.
- **Seguridad y Confianza:** Al permitir la ejecución remota de funciones, se necesita un alto nivel de seguridad y confianza entre los sistemas que se comunican. También la falta de una capa de abstracción entre la API y las funciones del sistema, planteando preocupaciones de seguridad.
- **Menos Popular que REST:** Aunque eficiente, RPC es menos popular que REST, lo que podría afectar la disponibilidad de recursos y bibliotecas de desarrollo.
- **Menos Orientado a Recursos:** Si bien REST se centra en los recursos, RPC se centra en las acciones. Esto puede ser una desventaja si la aplicación requiere una gestión más centrada en los datos.

### 3.4 GraphQL APIs

#### Descripción:

Aunque GraphQL no es realmente un protocolo separado, es un lenguaje de consulta distintivo con mejores prácticas para su uso. Al igual que una API REST, GraphQL utiliza HTTP para transmitir datos de texto en la carga útil de cada solicitud, pero su enfoque es diferente.

GraphQL surge como respuesta a las API REST, con el objetivo de ejecutar sintaxis precisa para recuperar solo lo necesario, optimizando el payload y simplificando el proceso. El usuario de GraphQL debe conocer los campos de datos disponibles, pero puede escribir una consulta que combine esos campos en el orden que desee. Las consultas se envían en la carga útil de una solicitud HTTP POST, y los datos se devuelven en la forma del esquema especificado por la consulta.

#### Pros:

- **Consulta Precisa:** Los clientes pueden solicitar solo la información necesaria, evitando la sobre o infra-solicitud de datos. Esto mejora la eficiencia de las transferencias de datos, especialmente en entornos con ancho de banda limitado.
- **Documentación Clara:** Las consultas de API son transparentes y están muy bien documentadas.

- **Flexibilidad en la Estructura de Datos:** A diferencia de REST, donde la estructura de los datos está predefinida, GraphQL permite a los clientes definir la estructura de los datos según sus necesidades específicas.
- **Reducción de Overfetching y Underfetching:** Elimina el problema de overfetching (recuperar más datos de los necesarios) y underfetching (no recuperar suficientes datos) al permitir a los clientes especificar exactamente qué datos necesitan.

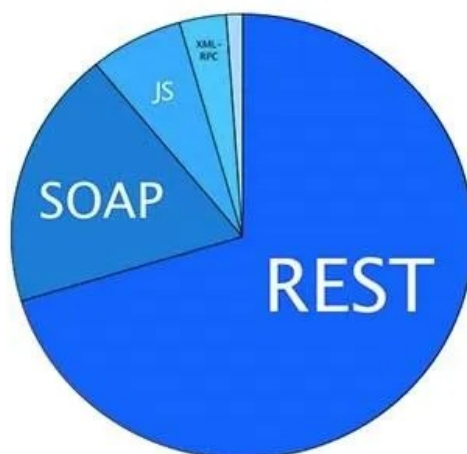
#### Contras:

- **Aprendizaje Inicial:** Puede haber una curva de aprendizaje inicial para los desarrolladores que no están familiarizados con GraphQL, ya que difiere significativamente de REST en términos de paradigmas de consulta.
- **Caché de Datos Desafiante:** No reutiliza las convenciones estándar de almacenamiento en caché HTTP, lo que requiere que la gestión de la caché de datos sea más desafiante, ya que es responsabilidad del consumidor mantener una sintaxis de consulta coherente.
- **Rendimiento con Consultas Anidadas:** El rendimiento puede verse afectado si hay muchas consultas anidadas o si se solicita una gran cantidad de datos en una sola consulta.
- **Menos Visibilidad de Endpoints:** A diferencia de REST, donde los endpoints son explícitos, GraphQL a menudo tiene menos visibilidad de los puntos finales, lo que puede hacer que los consumidores de la API tengan que depender más de la documentación.
- **Posible Exposición de Datos Sensibles:** Si no se configura correctamente, GraphQL puede exponer datos sensibles debido a su capacidad de recuperar datos de manera muy específica.



## 4 API REST

API REST es la más popular y dado que se basa en el protocolo HTTP para transmitir información y usa la estructura cliente servidor que hemos ido trabajando en las diferentes unidades, haremos varias prácticas con este tipo de API.



By: <https://lacliniquewp.com/api-rest/>

### 4.1 Uso de API de REST

Cada interfaz tiene su semántica propia y habrá que leer la documentación de la interfaz que vamos a usar.

#### 4.1.1 {JSON} Placeholder

Placeholder JSON es una plataforma de servicios gratuita que ofrece datos ficticios diseñados para diversas situaciones. Se utiliza con el propósito de explorar nuevas bibliotecas, perfeccionar habilidades de programación o como un recurso temporal hasta la finalización de tu propia API.

Esta solución te brinda acceso a información simulada que puedes incorporar durante el desarrollo y las fases de prueba. Esto simplifica la creación de entornos de trabajo previos a la implementación completa de la funcionalidad de tu API personalizada. Placeholder JSON resulta práctico para cubrir temporalmente la necesidad de datos mientras trabajas en el desarrollo de tu aplicación, ofreciendo una herramienta eficaz para este propósito.

- **URL:** <https://jsonplaceholder.typicode.com/>
- **Manual:** <https://jsonplaceholder.typicode.com/guide/>

**Importante para modificaciones de los datos:** el recurso no se actualizará realmente en el servidor, pero se simulará como si lo hiciera.

### 4.1.2 Openweathermap

OpenWeatherMap es un servicio en línea que proporciona datos meteorológicos y pronósticos del tiempo a nivel mundial. Ofrece una API (Interfaz de Programación de Aplicaciones) que permite a los desarrolladores acceder y utilizar esta información en sus propias aplicaciones y sitios web. Los datos proporcionados incluyen detalles sobre condiciones climáticas actuales, pronósticos a corto y largo plazo, información sobre viento, humedad, temperatura y más.

Para interactuar con la API de OpenWeatherMap, los desarrolladores pueden enviar solicitudes HTTP con parámetros específicos, como la ubicación geográfica o el intervalo de tiempo para el pronóstico. La API responderá con datos meteorológicos en formato JSON que luego pueden ser procesados y utilizados en la aplicación del desarrollador.

En resumen, OpenWeatherMap es una plataforma que facilita el acceso a datos meteorológicos actualizados y pronósticos, lo que resulta útil para una variedad de aplicaciones que requieren información climática, como aplicaciones móviles, sitios web y otros proyectos relacionados con el clima.

- **URL:** <https://api.openweathermap.org>
- **Manual:** <https://openweathermap.org/current>

### 4.1.3 OpenAI

OpenAI es una empresa de investigación en inteligencia artificial que tiene como objetivo avanzar en el desarrollo de IA. Han sido conocidos por desarrollar modelos de lenguaje avanzados, como GPT-3 (Generative Pre-trained Transformer 3), que son capaces de comprender y generar texto de manera notable. Estos modelos se han utilizado en una variedad de aplicaciones, desde asistentes virtuales hasta generación de contenido.

La plataforma de OpenAI, que puede ser accesible a través de su sitio web o API, permite a los desarrolladores integrar y utilizar estos modelos en sus propias aplicaciones y proyectos. Los desarrolladores pueden enviar solicitudes a la API para realizar tareas específicas, como generar texto, responder preguntas, traducir idiomas, entre otras capacidades basadas en procesamiento del lenguaje natural.

- **URL:** <https://platform.openai.com/>
- **Manual breve:** <https://platform.openai.com/docs/quickstart?context=python>
- **Manual completo:** <https://platform.openai.com/docs/guides/text-generation/chat-completions-api>

## 5 Referencias

<https://en.wikipedia.org/wiki/API>

<https://aprendiendoarduino.wordpress.com/tag/arquitectura-api/>

<https://www.mulesoft.com/es/resources/api/types-of-apis>

<https://www.techtarget.com/searchapparchitecture/tip/What-are-the-types-of-APIs-and-their-differences>

<https://stoplight.io/api-types>

<https://blog.postman.com/different-types-of-apis/>