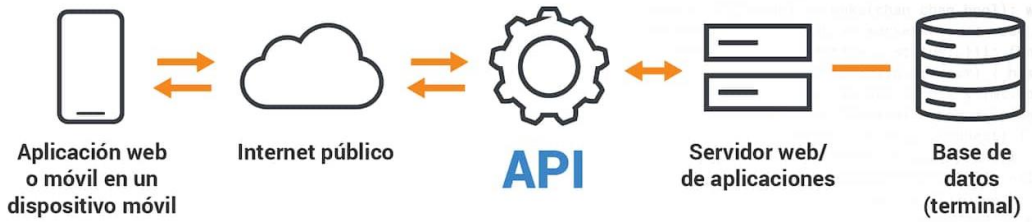
APIs

Una Interfaz de Programación de Aplicaciones (API) es un conjunto de reglas, protocolos y herramientas que permiten a los desarrolladores de software crear aplicaciones que interactúen con un sistema de software específico, servicio o plataforma. En esencia, una API define cómo los componentes de software deben comunicarse entre sí. Proporciona una interfaz clara y estructurada para que los desarrolladores interactúen con una aplicación o servicio.

Su propósito es permitir la integración y la interoperabilidad entre diferentes sistemas de software, simplificando el desarrollo de aplicaciones y permitiendo la reutilización de funcionalidades existentes.



Tipos de APIs:

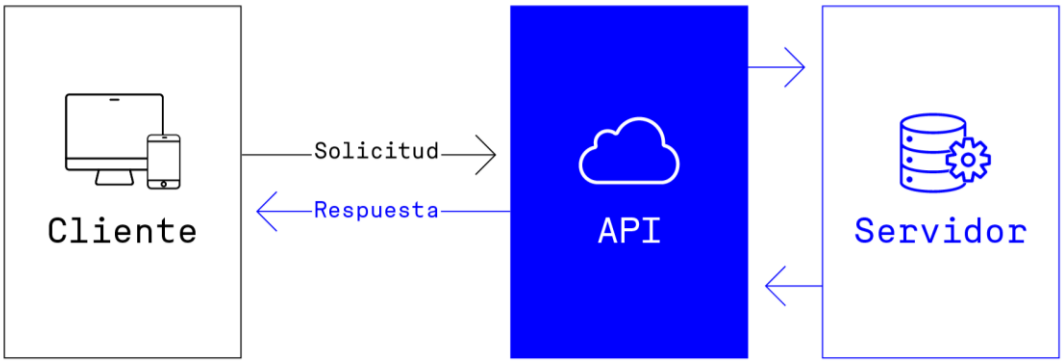
- APIs Web: Permiten la comunicación a través de la web utilizando protocolos como HTTP y HTTPS. Son comunes en servicios en la nube y plataformas web.
- APIs de Bibliotecas o SDKs: Proporcionan funciones y procedimientos que pueden ser invocados desde el código de una aplicación para acceder a funcionalidades específicas de una biblioteca o conjunto de herramientas.
- APIs de Sistemas Operativos: Permiten a las aplicaciones acceder a funciones del sistema operativo subyacente, como la gestión de archivos, la red y los dispositivos de hardware.
- APIs de Bases de Datos: Permiten a las aplicaciones acceder y manipular datos almacenados en una base de datos.

Estructura y Funcionamiento:

Una API define un conjunto de endpoints, o puntos finales, que representan las operaciones que se pueden realizar.

Cada endpoint tiene una URL única y acepta solicitudes en un formato específico (por ejemplo, JSON o XML) y devuelve respuestas también en un formato específico.

Las operaciones comunes incluyen la obtención de datos (mediante el método GET), la creación de nuevos datos (mediante el método POST), la actualización de datos existentes (mediante el método PUT o PATCH) y la eliminación de datos (mediante el método DELETE).



Autenticación y Autorización:

Las APIs suelen requerir algún tipo de autenticación para verificar la identidad del usuario o de la aplicación que realiza la solicitud.

Además, pueden utilizar sistemas de autorización para controlar el acceso a determinados recursos o funcionalidades basados en roles o permisos específicos.

Documentación y Versionado:

Una buena documentación es crucial para que los desarrolladores comprendan cómo utilizar una API correctamente. Esto incluye descripciones detalladas de los endpoints, los parámetros de las solicitudes y las respuestas esperadas.

El versionado de la API garantiza la compatibilidad hacia atrás y permite realizar cambios en la API sin afectar a los usuarios existentes. Cada versión de la API puede tener su propia documentación y políticas de soporte.

Ejemplos de Aplicación:

Las APIs se utilizan en una amplia variedad de aplicaciones y servicios, desde redes sociales y plataformas de comercio electrónico hasta sistemas de gestión empresarial y dispositivos IoT.

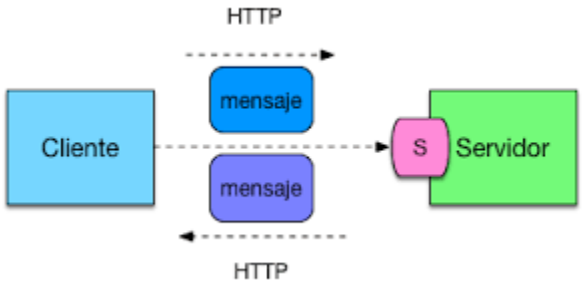
Ejemplos populares incluyen la API de Twitter para acceder a datos de tweets, la API de Google Maps para integrar mapas en aplicaciones, y la API de Stripe para procesar pagos en línea.

REST

La Transferencia de Estado Representacional (REST) es una arquitectura de software que establece pautas para el diseño de API, inicialmente concebida para gestionar la comunicación en redes complejas como Internet.

Se basa en la arquitectura de la World Wide Web y proporciona un marco para implementar comunicaciones confiables y de alto rendimiento a gran escala, siendo flexible y modificable, lo que garantiza visibilidad y portabilidad entre plataformas para cualquier sistema de API.

Las API que siguen los principios de REST se denominan API REST, y los servicios web que implementan REST se conocen como servicios web RESTful.



- Recursos: En REST, todo es considerado un recurso. Un recurso puede ser cualquier cosa que pueda ser nombrada, como un documento, una imagen, un usuario, etc. Cada recurso tiene un identificador único (URI) y es manipulado a través de la representación de su estado.
- Operaciones CRUD: REST utiliza las operaciones CRUD (Crear, Leer, Actualizar, Borrar) para interactuar con los recursos. Estas operaciones se mapean a los métodos HTTP: GET para leer, POST para crear, PUT o PATCH para actualizar y DELETE para borrar.
- Interfaz Uniforme: Una de las restricciones principales de REST es su interfaz uniforme, que estandariza el uso de URI's (identificadores uniformes) para manipular recursos con el uso de operaciones CRUD. El servidor transfiere información en un formato estándar llamado "representación".
- Formatos de Representación: REST utiliza formatos de representación estándar como JSON o XML para intercambiar datos entre el cliente y el servidor. Estos formatos son legibles tanto por humanos como por máquinas, lo que facilita la interoperabilidad entre diferentes sistemas.
- Tecnología sin Estado: Implica que el servidor no almacena ningún estado de cliente entre las solicitudes. Cada solicitud que realiza un cliente al servidor debe contener toda la información necesaria para que el servidor comprenda y procese la solicitud, el servidor no debe mantener ningún tipo de estado de sesión entre las solicitudes del cliente. Esto hace que las aplicaciones sean más escalables y fáciles de mantener.
- Estado de la Solicitud: Las respuestas de las solicitudes REST suelen incluir un código de estado que indica si la solicitud fue exitosa o si ocurrió algún error.
- Sistema por Capas: El cliente y el servidor son independientes entre sí, lo que permite que cada uno evolucione de forma independiente. Los clientes se comunican con los

servidores a través de intermediarios autorizados, facilitando la modularidad y la escalabilidad del sistema.

- Almacenamiento en Caché: Permite almacenar en caché las respuestas del servidor para mejorar el rendimiento y la eficiencia de la red, reduciendo la necesidad de repetir solicitudes para recursos estáticos.
- Código Bajo Demanda: Permite a los servidores transferir temporalmente código de programación al cliente para extender o personalizar su funcionalidad, lo que puede mejorar la experiencia del usuario y la interactividad de la aplicación web.