# Intro API rest

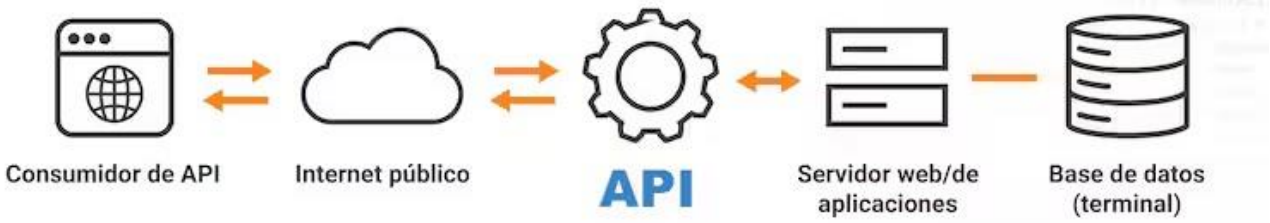
1. ¿Qué es una API?
    1. Tipos de API web
    2. Diferencias entre las API y los terminales
2. ¿Qué se necesita para crear una API REST)

# 1. ¿Qué es una API?

Una API (Application Programming Interface) es un conjunto de reglas que permite a diferentes aplicaciones comunicarse entre sí. En el contexto de la web, las APIs permiten que aplicaciones o servicios intercambien datos a través de internet.

Una API web es una **interfaz programática** que consta de uno o más terminales expuestos públicamente a un **sistema de mensajes de SOLICITUD-RESPUESTA** definido (req, res), normalmente expresado en formato **JSON o XML**, que se expone a través de la Web, normalmente mediante un **servidor web basado en HTTP.**

Piensa en una API como un mesero en un restaurante: el cliente (tu aplicación) hace un pedido (request), el mesero (la API) lo lleva a la cocina (el servidor) y regresa con lo que pediste (response).

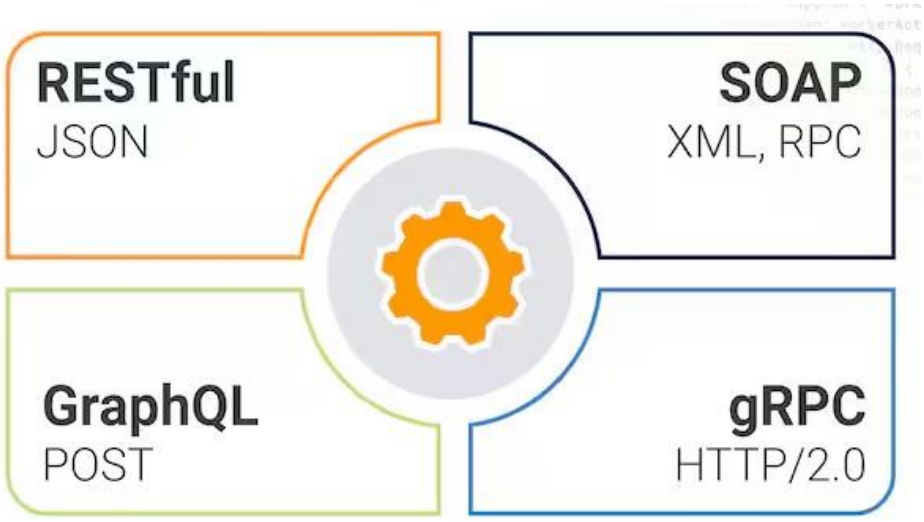



## Conceptos clave:

- **Recurso**: Representa algo que la API maneja (como usuarios, productos, o pedidos).
- **Ruta**: Es la dirección específica (URI) donde está disponible un recurso.
- **URI (Identificador Uniforme de Recursos)**: Es una dirección única para identificar un recurso, como `https://api.ejemplo.com/usuarios` .
- **Métodos HTTP**: Son las acciones que puedes realizar sobre un recurso:
    - `GET` : Obtener información.
    - `POST` : Crear un nuevo recurso.
    - `PUT` : Actualizar un recurso existente.
    - `DELETE` : Eliminar un recurso.

## 1.1 Tipos de API web

- **API RESTful**: La más común. Utiliza métodos HTTP y trabaja con recursos representados en formato JSON o XML.
  - Las API RESTful son fáciles de utilizar por los marcos de front-end modernos (por ejemplo, React y React Native) y facilitan el desarrollo de aplicaciones web y móviles.
    - Ejemplo: `GET /usuarios` devuelve una lista de usuarios.
- **SOAP**: Más estructurada y basada en XML (lenguaje de marcado extensible). Se usa en aplicaciones corporativas.
  - Ejemplo: Realizar operaciones bancarias.
- **GraphQL**: Permite que el cliente especifique exactamente qué datos necesita, lo que reduce la cantidad de datos transferidos.
  - Ejemplo: Solicitar sólo el nombre y correo electrónico de un usuario.
  - Desarrollado por Facebook, proporciona **acceso a las bases de datos** a través de un **único terminal POST** (normalmente /graphql). Resuelve un problema común de las API RESTful, que requieren varias llamadas para rellenar una sola página de interfaz de usuario, además de introducir otros problemas adicionales.
- **API gRPC**: un nuevo protocolo binario (lenguaje más rápido y compacto) de alto rendimiento desarrollado por Google sobre HTTP/2.0 (versión moderna de envío de datos por internet), que se utiliza principalmente para la comunicación este-oeste.
  - reduce el tiempo que toma enviar y recibir datos entre programas, especialmente cuando hay mucha información o se necesitan muchas respuestas rápidas.
  - "Comunicación este-oeste" es cuando los programas dentro de la misma empresa (o en el mismo sistema) hablan entre ellos. Por ejemplo, el programa de ventas necesita pedir información al programa de contabilidad para saber si hay presupuesto para algo.
- **WebSocket**: Para comunicación bidireccional en tiempo real, como en chats o videojuegos.



## 1.2 Diferencias entre APIs y terminales

- **API**:
  - Es el conjunto completo de reglas y recursos.
  - Proporciona acceso estructurado a varias funcionalidades o datos de un sistema.
  - Ejemplo: Toda la funcionalidad de `https://api.ejemplo.com` .
- **Terminal o Endpoint**:
  - Los terminales constan de **rutas de acceso a recursos**
    - las **operaciones** que se pueden realizar en esos **recursos** y la definición de los **datos** de recursos (en JSON, XML u otro formato).
  - Un terminal o endpoint es una **dirección específica de una API** donde puedes acceder a un **recurso particular**.
  - Ejemplo: `https://api.ejemplo.com/usuarios/123` (este endpoint da acceso al usuario con ID 123).

En resumen, una API es como un menú completo, y un terminal es un plato específico que puedes pedir del menú.

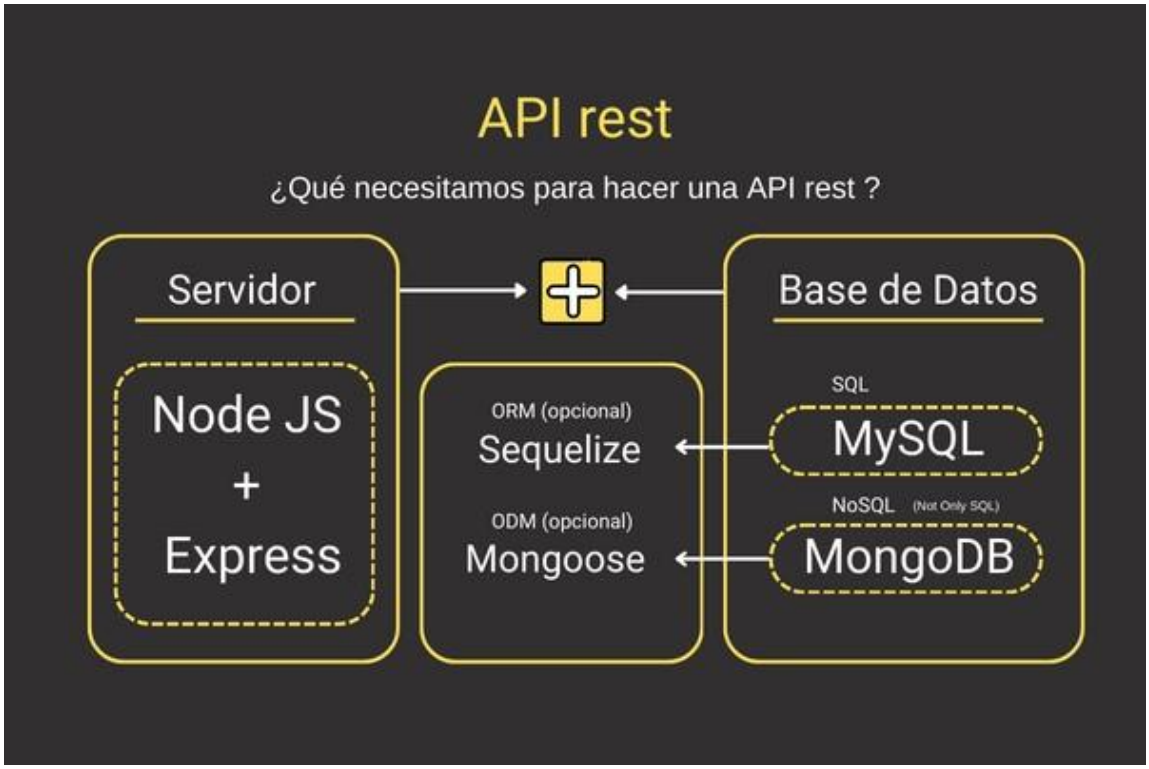
# 2. ¿Qué se necesita para crear una API REST)

Lo mínimo necesario para crear una API REST (Representational State Transfer) incluye:

1. **Un servidor HTTP**: Necesitarás un servidor web que pueda manejar solicitudes HTTP. Puedes usar frameworks como Express.js en Node.js, Flask en Python, o Spring Boot en Java para crear este servidor.
2. **Endpoints**: Define los puntos finales (endpoints) de tu API, que son las URLs a través de las cuales los clientes pueden acceder a los recursos. Por ejemplo, `/usuarios` , `/productos` , etc.
3. **Métodos HTTP**: Define qué métodos HTTP serán permitidos en cada endpoint. Los métodos comunes son GET para obtener datos, POST para agregar nuevos datos, PUT o PATCH para actualizar datos existentes, y DELETE para eliminar datos.
4. **Lógica de negocio**: Implementa la lógica necesaria para manejar las solicitudes recibidas en cada endpoint. Esto puede incluir acceso a una base de datos, validación de datos, autenticación y autorización, entre otras cosas.
5. **Formato de datos**: Decide qué formato de datos utilizarás para intercambiar información con los clientes. JSON es el formato más comúnmente utilizado en las APIs REST debido a su simplicidad y legibilidad.
6. **Documentación**: Es importante documentar tu API para que los clientes sepan cómo usarla correctamente. Puedes utilizar herramientas como Swagger o Postman para generar documentación automáticamente a partir de tu código.

Con estos elementos básicos, puedes crear una API REST funcional que permita a los clientes interactuar con tus recursos a través de solicitudes HTTP estándar.

## API rest y Bases de Datos




# Relación entre API rest y Base de Datos

Las APIs REST actúan como una capa intermedia entre las aplicaciones y las bases de datos, permitiendo que las aplicaciones accedan y manipulen datos de manera controlada a través de solicitudes HTTP estándar.

Las API REST definen endpoints que representan recursos en la base de datos, como usuarios o productos, y utilizan métodos HTTP (GET, POST, PUT, DELETE) para realizar operaciones CRUD en estos recursos.

La API interpreta las solicitudes entrantes y las traduce en consultas de base de datos, interactuando con la base de datos para recuperar, crear, actualizar o eliminar datos según sea necesario. Esto proporciona una separación clara entre la lógica de la aplicación y la lógica de acceso a datos, lo que facilita la escalabilidad y el mantenimiento del sistema.

Además, las API REST pueden implementar lógica adicional, como la autenticación y la autorización, para garantizar la seguridad y la integridad de los datos.