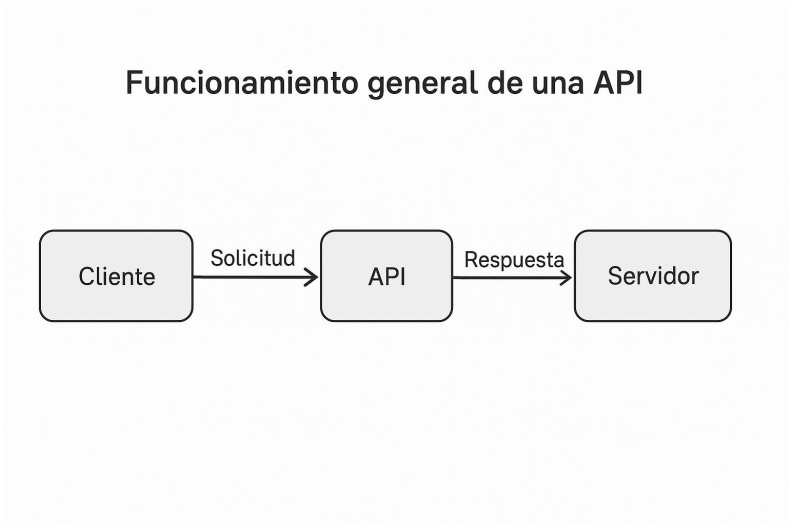
M5. Minería de Texto

Clase 5b. Introducción a APIs

¿Qué es?

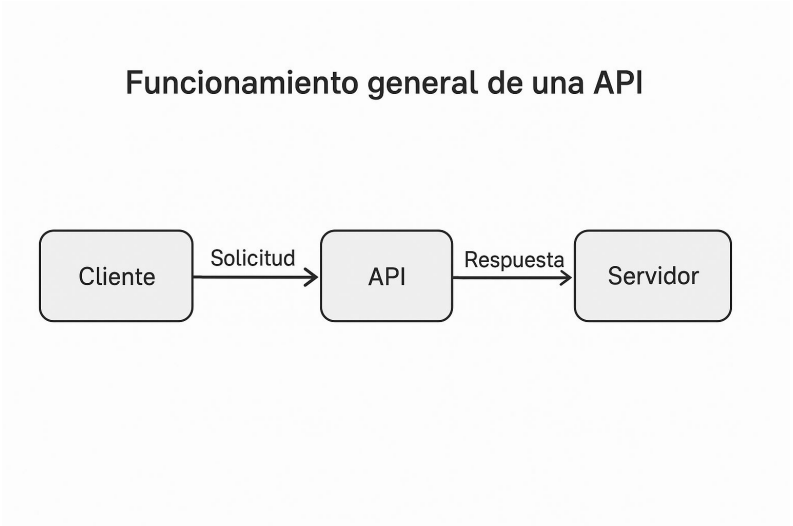
- Una API (Application Programming Interface) es un conjunto de reglas y protocolos que permite que diferentes programas se comuniquen entre sí.

- Elementos centrales:
  - Solicitud (request): lo que pedimos al servicio.
  - Respuesta (response): lo que el servicio nos devuelve.
  - Formato estandarizado: generalmente JSON o XML.



Tipos de solicitudes: GET y POST

- En este curso vamos a usar solamente dos requests:
  - GET: se usa cuando queremos obtener información existente.
    - Ejemplo: traer datos de la API del Banco Mundial (Ejercicio 2).
    - Las variables suelen ir en la URL (parámetros de consulta).
  - POST: se usa cuando queremos enviar datos para que el servidor procese.
    - Ejemplo: enviar un prompt a un LLM.
    - El contenido (prompt, parámetros, etc.) se manda en el cuerpo de la request (body).



Tipos de solicitudes: GET y POST

Característica	GET	POST
Uso principal	Obtener información existente	Enviar datos para que sean procesados
Dónde viajan los datos	En la **URL** (parámetros de consulta)	En el **cuerpo (body)** de la solicitud
Ejemplo típico	Descargar datos de un censo, series estadísticas, datos climáticos	Enviar un *prompt* a un LLM, cargar datos a un servidor
Visibilidad	Parámetros visibles en la URL (menos seguro para info sensible)	Datos ocultos en el body (más seguro para inputs largos o confidenciales)
Tamaño de datos	Limitado (las URLs no pueden ser muy largas)	Puede manejar volúmenes más grandes
Idempotencia (misma request varias veces)	Sí: pedir lo mismo devuelve siempre lo mismo	No necesariamente: puede generar resultados distintos
Ejemplo en R	`GET("https://api.worldbank.org/v2/...")`	`POST(url, body = toJSON(prompt))`

Otras solicitudes

Método	Función principal	Ejemplo de uso
GET	Obtener información ya existente.	Consultar datos de un censo desde una API.
POST	Enviar datos para que se procesen o creen nuevos recursos.	Enviar un *prompt* a un LLM.
PUT	Reemplazar completamente un recurso existente.	Actualizar un registro completo de un individuo en una base de encuestas.
PATCH	Modificar parcialmente un recurso existente.	Cambiar solo la edad de un registro en una encuesta sin tocar el resto de los datos.
DELETE	Eliminar un recurso existente.	Borrar un registro de prueba cargado en una base de datos vía API.
HEAD	Igual que GET, pero solo devuelve los *headers* (metadatos), no el contenido.	Verificar si un dataset está disponible y su fecha de última actualización sin descargarlo.
OPTIONS	Devuelve qué métodos están permitidos en un recurso.	Preguntar a una API si admite `GET` y `POST`.

Usos posibles

- **Acceso a datos:** bases estadísticas, censos, encuestas, información meteorológica, redes sociales.
- **Automatización de tareas:** por ejemplo, descargar diariamente un dataset sin intervención manual.
- **Servicios de terceros:** traducción automática, geolocalización, procesamiento de imágenes, modelos de lenguaje.
- **Ventaja principal:** las APIs permiten trabajar con información y servicios sin necesidad de descargar programas completos o manejar manualmente los datos.

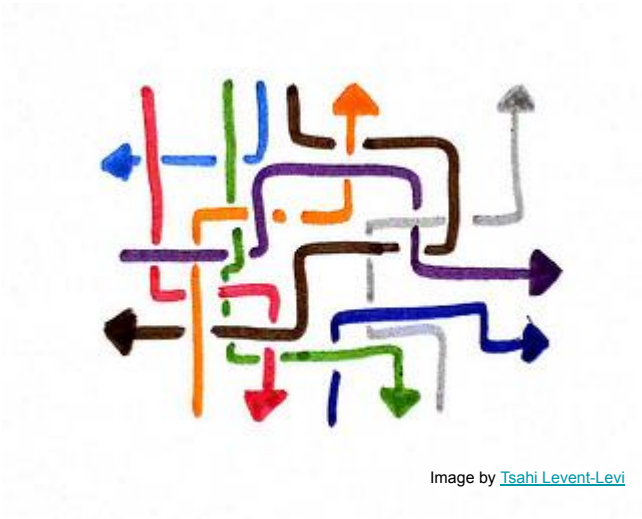
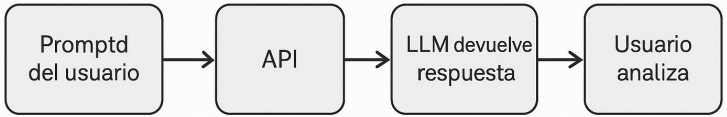


Image by Tsahi Levent-Levi

Vamos al Notebook

APIs y LLMs

- Los LLMs (como GPT, Gemini, Claude, LLaMA) exponen una API que recibe texto de entrada (prompt) y devuelve texto generado como salida (completion).
- Elementos clave en la consulta a un LLM mediante API:
  - **Endpoint:** la URL a la que se envía la solicitud.
  - **Clave de autenticación:** código personal para acceder al servicio.
  - **Parámetros:** prompt, temperatura, número máximo de tokens, etc.



Gemini (LLM de Google)

- Familia de LLMs accesibles mediante la Google AI Studio API.
- Uso de clave de autenticación de Google Cloud.

- Respuestas en formato JSON.
- Posibilidad de controlar parámetros como temperature, maxOutputTokens, etc.

Una consulta a un LLM genérico

```r
library(httr)
library(jsonlite)

url <- "https://api.llm-ejemplo.com/v1/chat"
api_key <- "TU_API_KEY"

prompt <- list(
        model = "llm-demo",
        messages = list(
                list(
                role = "user",
                content = "Explicá qué es una clase social en términos sencillos.")
        )
)

respuesta <- POST(
        url,
        add_headers(Authorization = paste("Bearer", api_key)),
        body = toJSON(prompt, auto_unbox = TRUE),
        encode = "json"
)

resultado <- fromJSON(content(respuesta, "text"))
cat(resultado$choices[[1]]$message$content)
```

Vamos al Notebook