

# Laboratorio de datos: web scraping y Procesamiento de Lenguaje Natural

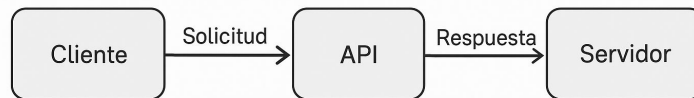
## Clase 8c. Introducción a APIs



# ¿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.

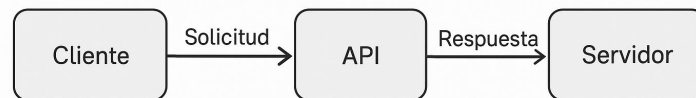
## Funcionamiento general de una API



# 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).

## Funcionamiento general de una API



# Tipos de solicitudes: GET y POST

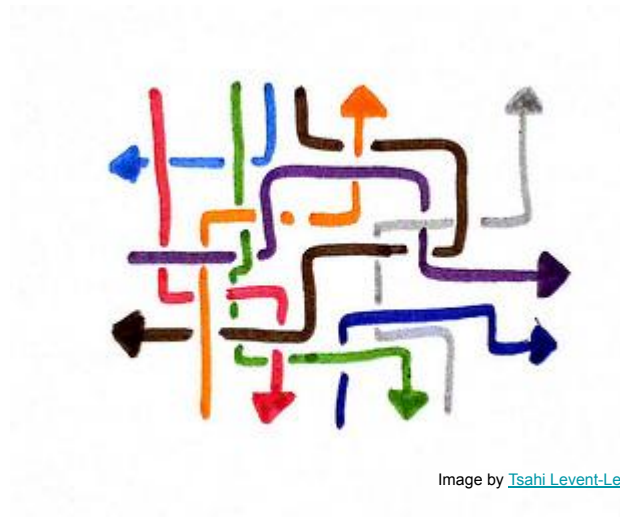
Característica	GET	POST
<b>Uso principal</b>	Obtener información existente	Enviar datos para que sean procesados
<b>Dónde viajan los datos</b>	En la <b>URL</b> (parámetros de consulta)	En el <b>cuerpo (body)</b> de la solicitud
<b>Ejemplo típico</b>	Descargar datos de un censo, series estadísticas, datos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

# Otras solicitudes

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

# Usos posibles

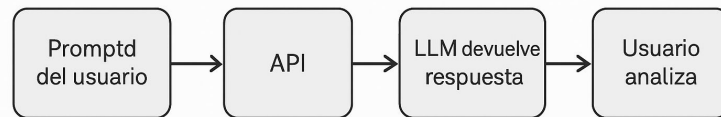
- **Acceso a datos:** bases estadísticas, censos, encuestas, información meteorológica, redes sociales.
- **Automatización de tareas:** por ejemplo, descargar diariamente un dataset sin intervención manual.
- **Servicios de terceros:** traducción automática, geolocalización, procesamiento de imágenes,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



**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

```
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**