



Laboratorio Nacional de Políticas Públicas



CENTRO DE INVESTIGACIÓN
Y DOCENCIA ECONÓMICAS A.C.

APIs

Periodismo de Datos
Abril, 2021

M.C. JORGE JUVENAL CAMPOS FERREIRA.

Investigador Asociado.

Laboratorio Nacional de Políticas Públicas

CIDE



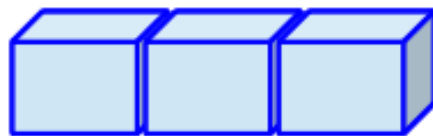
1. Revisión de conceptos básicos:

- Repaso de conceptos
- Qué es un API.
- Donde se usa un API
- Qué es un Cliente de API para R.
- Que es una llamada a una API

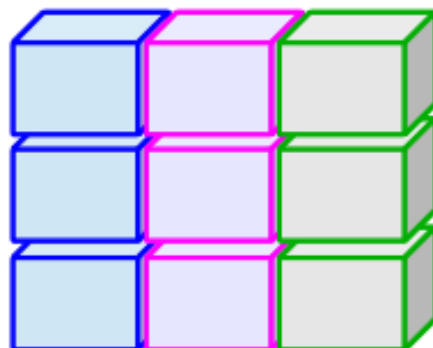
2. Ejemplo práctico. Acceso a datos a través de API:

Listas

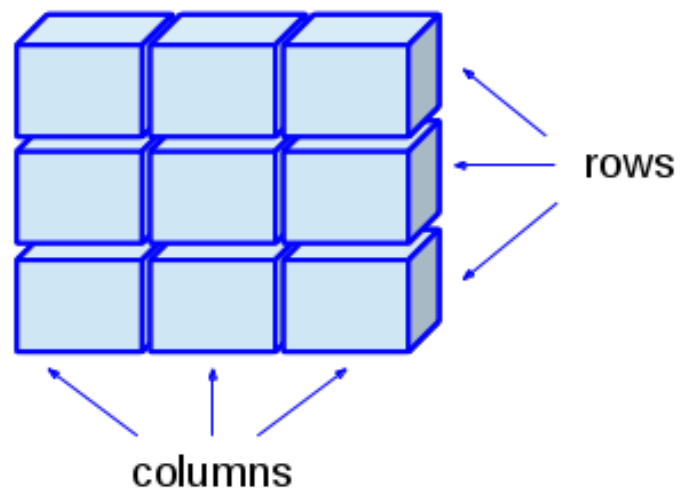
Vector



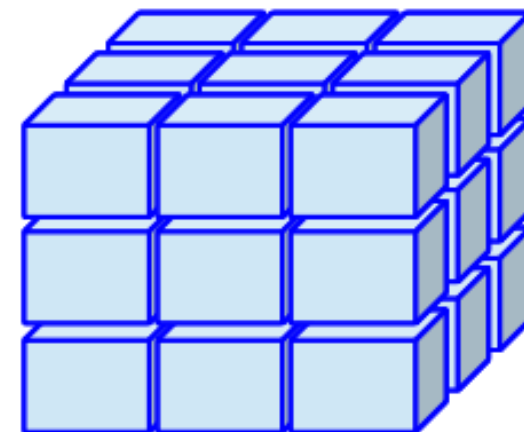
Data Frame
(Table)



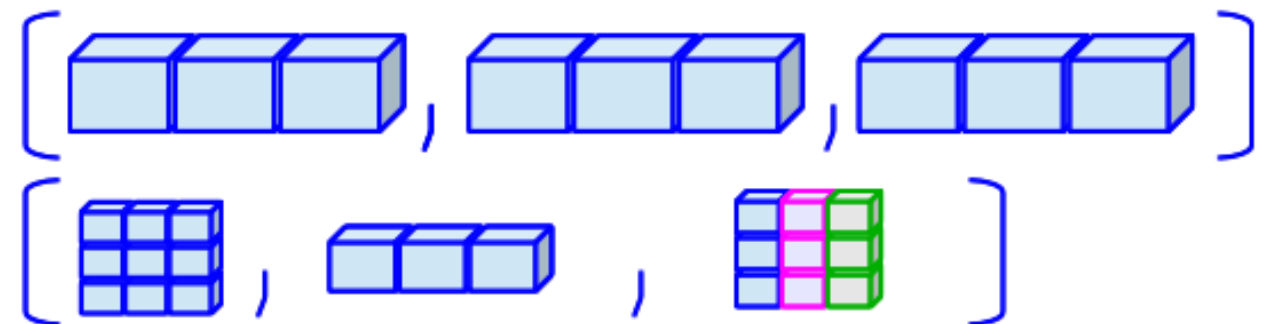
Matrix



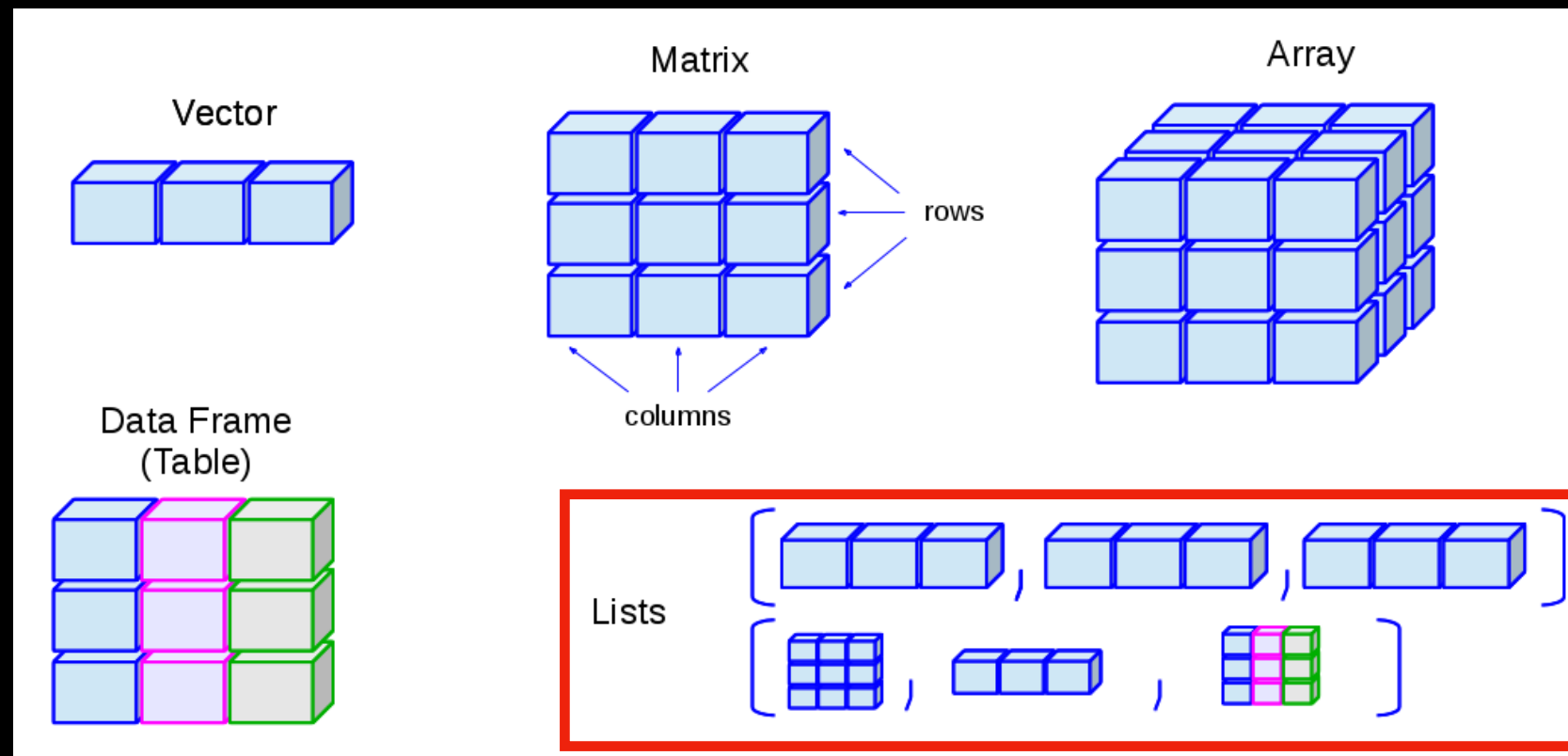
Array



Lists



Listas



Una lista es como un vector, en donde el contenido de cada casilla puede ser de **cualquier** tipo (otra lista adentro, un vector, una tibble o dataframe, etc.).

stringr::str_glue()



```
str_glue(..., .sep = "", .envir =  
parent.frame())
```

Función para pegar objetos dentro de una cadena de texto.

```
> # Ejemplo:  
> nombre = "Juvenal"  
> lugar_trabajo = "el CIDE"  
> ocupacion = "ingeniero"  
> str_glue("Mi nombre es {nombre}, trabajo en {lugar_trabajo} y soy {ocupacion}")  
Mi nombre es Juvenal, trabajo en el CIDE y soy ingeniero
```

Formato JSON



```
{
  "users": [
    {
      "userId": 1,
      "firstName": "Chris",
      "lastName": "Lee",
      "phoneNumber": "555-555-5555",
      "emailAddress": "clee@fileinfo.com"
    },
    {
      "userId": 2,
      "firstName": "Action",
      "lastName": "Jackson",
      "phoneNumber": "555-555-5556",
      "emailAddress": "ajackson@fileinfo.com"
    }
  ]
}
```

```
"store": {
  "book": [
    {
      "category": "reference",
      "author": "Nigel Rees",
      "title": "Sayings of the Century",
      "price": 8.95
    },
    {
      "category": "fiction",
      "author": "Evelyn Waugh",
      "title": "Sword of Honour",
      "price": 12.99
    },
    {
      "category": "fiction",
      "author": "Herman Melville",
      "title": "Moby Dick",
      "isbn": "0-553-21311-3",
      "price": 8.99
    }
  ]
}
```

Ejemplos:

https://github.com/JuveCampos/Shapes_Resiliencia_CDMX_CIDE/raw/master/Infraestructura/Pozos_Sacmex2.geojson;
Ver archivo carpeta local.

Formato JSON



Un archivo *.json (JavaScript Object Notation) es un **archivo que almacena estructuras simples** de datos.

- Este formato es un **estándar para el intercambio de información**, y se usa para transmitir información entre una aplicación y un servidor.
- Estos archivos son **ligeros, basados en texto, legibles por humanos** y pueden ser editados por un editor de texto.
- Es el **formato** que se utiliza para **dar información proveniente de un API**.
- Para leerlos **en R**, tenemos que usar funciones de la librería **{jsonlite}**.

```
{
  "tag": "select",
  "name": "ctl00$plnMain$ddlClasses",
  "id": "plnMain_ddlClasses",
  "class": "sg-combobox",
  "children": [
    {
      "tag": "option",
      "selected": "selected",
      "value": "ALL",
      "html": "(All Classes)"
    },
    {
      "tag": "option",
      "value": "1319202|1",
      "html": "0020A - 36  STUDY HALL (INSTRUCT)"
    },
    {
      "tag": "option",
      "value": "1319203|1",
      "html": "0020B - 33  STUDY HALL (INSTRUCT)"
    },
    {
      "tag": "option",
      "value": "1232138|1",
      "html": "0113A - 1  APENGLAN A"
    },
    {
      "tag": "option",
      "value": "1232137|1",
      "html": "0113B - 1  APENGLAN B"
    },
    {
      "tag": "option",
      "value": "1229605|1",
      "html": "0284A - 2  PRE CALC PREAP A"
    },
    {
      "tag": "option",
      "value": "1229608|1",
      "html": "0284B - 2  PRE CALC PREAP B"
    },
    {
      "tag": "option",
      "value": "1229088|1",

```

APIs

API (*Application Programming Interface* o *Interfaz de Programación de Aplicaciones*).

Las APIs son **componentes** del servidor que **hacen que sea fácil que nuestro código interactúe con un servicio y obtenga datos de este.**



APIs





En una contexto de R:

- Las APIs **son como páginas web, pero para máquinas.**
- Pueden ser utilizadas para **importar grandes cantidades de datos** de manera automática.
- Nos permiten hacer **búsquedas para pedazos específicos de los datos**, dentro de un universo de datos muy grande.



– Para R, existen dos formas de usar las APIs.

Con la librería `{httr}`

La librería `httr` nos permite hacer consultas a las APIs de una manera tradicional (muy similar a como lo haría un programador).

A través de API Clients

Son librerías de R específicas para acceder a las APIs de diferentes servicios. Ejemplos: `{spotifyr}`, `{rtweet}`, `{tuber}`, o `{pageviews}`.

Cientes APIs para R

La forma fácil

- Siempre usar un cliente, cuando sea posible.
- Son interfaces nativas para acceder a las APIs con código y funciones de R. (Son como cualquier otra librería).
- Ocultan completamente la API, y nos permite acceder solo a los resultados estructurados.
- Nos permiten leer los datos a través de objetos de R, en vez de tener que lidiar con archivos JSON.

Etiqueta al usar APIs



– Muchas APIs nos exigen un registro previo, esto con el fin de **controlar el acceso y el uso moderado.**

– Para usar una API, tras el registro, se nos dan tokens de acceso para controlar el acceso y el consumo de los servicios. **Estas llaves muchas veces van incluidas en nuestro código.**

Etiqueta al usar APIs



- Sobrecargar una API con solicitudes puede **reducir la calidad del servicio** tanto para ti como para otros usuarios.
- Muchas APIs regulan el consumo de datos: **permitiendo cierto numero de consultas durante cierto tiempo.**

¿Y si no hay cliente de APIs?



Cuando no hay un paquete intermediario que nos permita comunicarnos con un API, hay que **conectarnos de manera directa con este servicio.**

Para esto, hay que utilizar las librerías *{httr}* y *{jsonlite}*

Llamadas http

GET and POST request in theory

HTTP requests

- Conversation between your machine and the server
- First: what you want to happen
- 'methods' - different requests for different tasks

GET and POST

- GET: 'get me something'
- POST: 'have something of mine'



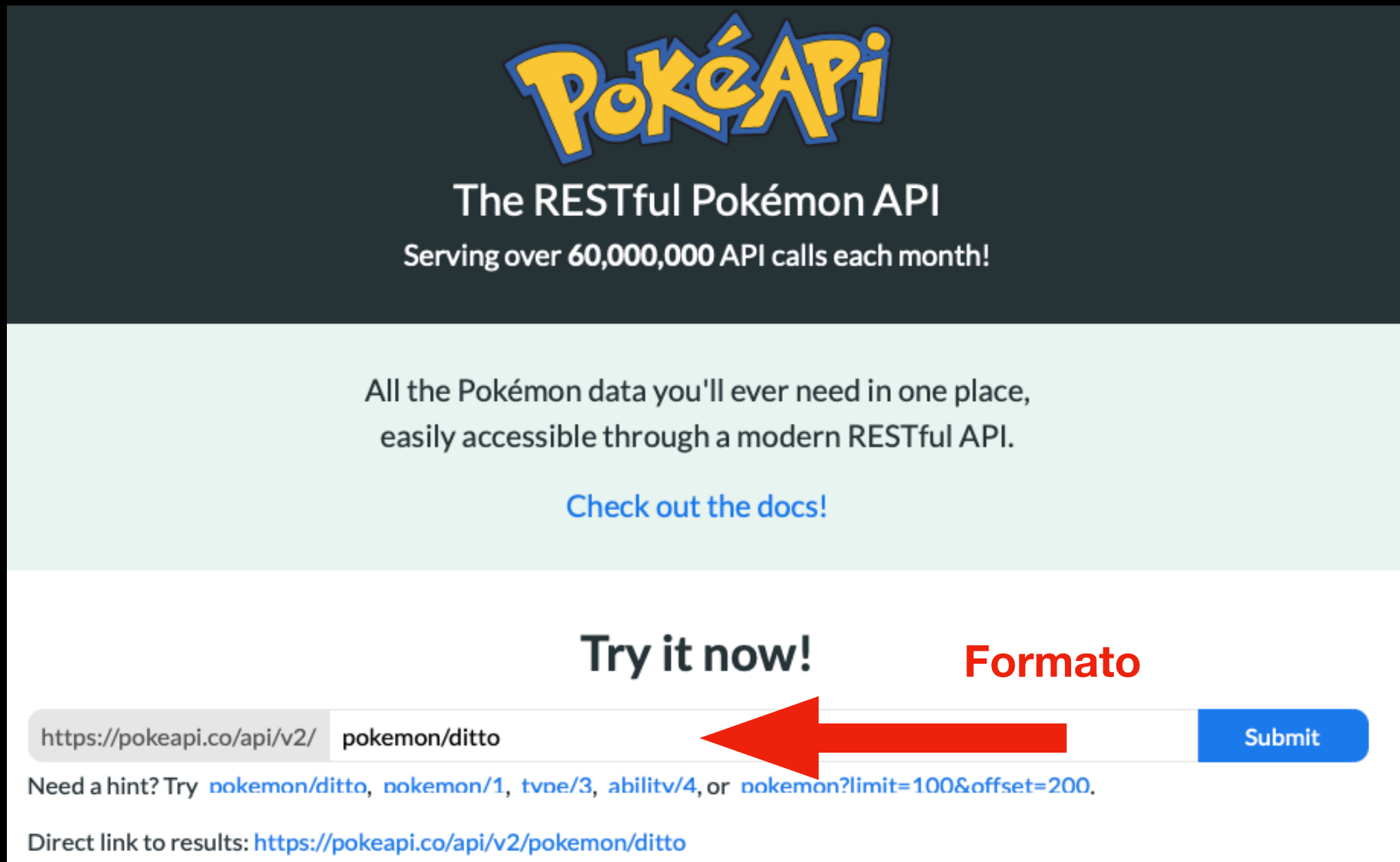
El bueno

Other tyoes

- HEAD - just like `head()`
- DELETE - 'remove this thing'
- Many others! But GET and POST are the big ones!

Haciendo una conexión directa

Paso 0. Construyo mi petición en el formato que me lo indique la documentación.



The screenshot shows the PokéAPI website. At the top, the logo "PokéAPI" is displayed in a stylized yellow font with a blue outline. Below it, the text "The RESTful Pokémon API" and "Serving over 60,000,000 API calls each month!" are shown. A light blue section contains the text "All the Pokémon data you'll ever need in one place, easily accessible through a modern RESTful API." and a link "Check out the docs!". Below this, a "Try it now!" button is visible. To the right of the button, the word "Formato" is written in red. A red arrow points from the "Formato" text to the input field of the API request bar. The input field contains the URL "https://pokeapi.co/api/v2/pokemon/ditto". To the right of the input field is a blue "Submit" button. Below the input field, a hint is provided: "Need a hint? Try [pokemon/ditto](#), [pokemon/1](#), [type/3](#), [ability/4](#), or [pokemon?limit=100&offset=200](#)." At the bottom, a direct link to the results is shown: "Direct link to results: <https://pokeapi.co/api/v2/pokemon/ditto>".

PokéAPI
The RESTful Pokémon API
Serving over 60,000,000 API calls each month!

All the Pokémon data you'll ever need in one place,
easily accessible through a modern RESTful API.

[Check out the docs!](#)

Try it now! **Formato**

Need a hint? Try [pokemon/ditto](#), [pokemon/1](#), [type/3](#), [ability/4](#), or [pokemon?limit=100&offset=200](#).

Direct link to results: <https://pokeapi.co/api/v2/pokemon/ditto>

Haciendo una conexión directa



Paso 1. Ya que vi como se tenía que construir la pregunta (llamada), la construyo.

parte_constante + parte_variable

```
# Llamada a la API  
call1 <- paste0("https://pokeapi.co/api/v2/",  
               "pokemon/pikachu/")
```

Haciendo una conexión directa



Paso 2. Dada la dirección de la API (la llamada), hago una GET request, con la función *httr::GET()*, lo cual nos devuelve una respuesta de la información solicitada.

```
# Realizamos la llamada (Obtenemos la respuesta)
llamada <- GET(call1)
llamada # Status 200: Todo ok.
class(llamada) # Tipo Response
```

Objeto response



```
> llamada
Response [https://pokeapi.co/api/v2/pokemon/pikachu/]
Date: 2020-10-01 18:35
Status: 200
Content-Type: application/json; charset=utf-8
Size: 191 kB
```

Haciendo una conexión directa



Paso 3. De mi objeto response, aplico la función *httr::content()* para acceder al contenido.

```
# De la llamada, obtenemos la respuesta en formato JSON
# Formato JSON: https://en.wikipedia.org/wiki/JSON
get_data <- content(llamada, "text")
get_data
class(get_data)
```

Datos tipo JSON



```
\": \"machine\", \"url\": \"https://pokeapi.co/api/v2/move-learn-method/4/\", \"version_group\": {\"name\": \"crystal\", \"url\": \"https://pokeapi.co/api/v2/version-group/4/\"}}, {\"level_learned_at\": 0, \"move_learn_method\": {\"name\": \"tutor\", \"url\": \"https://pokeapi.co/api/v2/move-learn-method/3/\"}, \"version_group\": {\"name\": \"emerald\", \"url\": \"https://pokeapi.co/api/v2/version-group/6/\"}}, {\"level_learned_at\": 0, \"move_learn_method\": {\"name\": \"machine\", \"url\": \"https://pokeapi.co/api/v2/move-learn-method/4/\"}, \"version_group\": {\"name\": \"diamond-pearl\", \"url\": \"https://pokeapi.co/api/v2/version-group/8/\"}}, {\"level_learned_at\": 0, \"move_learn_method\": {\"name\": \"machine\", \"url\": \"https://pokeapi.co/api/v2/move-learn-method/4/\"}, \"version_group\": {\"name\": \"platinum\", \"url\": \"https://pokeapi.co/api/v2/version-group/9/\"}}, {\"level_learned_at\": 0, \"move_learn_method\": {\"name\": \"machine\", \"url\": \"https://pokeapi.co/api/v2/move-learn-method/4/\"}, \"version_group\": {\"name\": \"heartgold-soulsilver\", \"url\": \"https://pokeapi.co/api/v2/version-group/10/\"}}] }
```

Haciendo una conexión directa



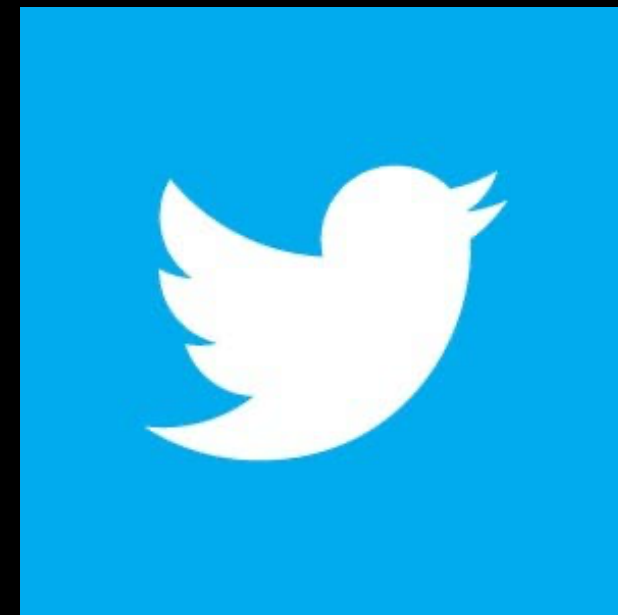
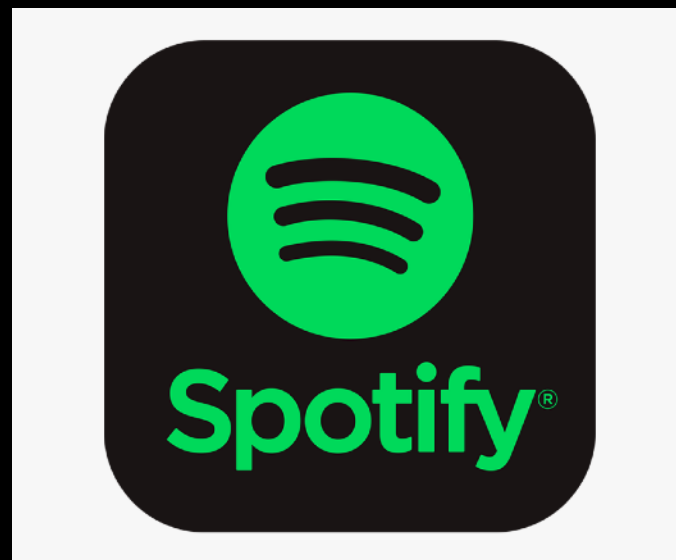
Paso 4. Una vez que tengo mi objeto respuesta arrojó el resultado en formato *JSON*, lo convierto en un objeto tipo *list* con la función *jsonlite::fromJSON()*

```
# Del JSON, convertimos este texto en un objeto lista
get_data_JSON <- fromJSON(get_data, flatten = TRUE)
class(get_data_JSON)
```

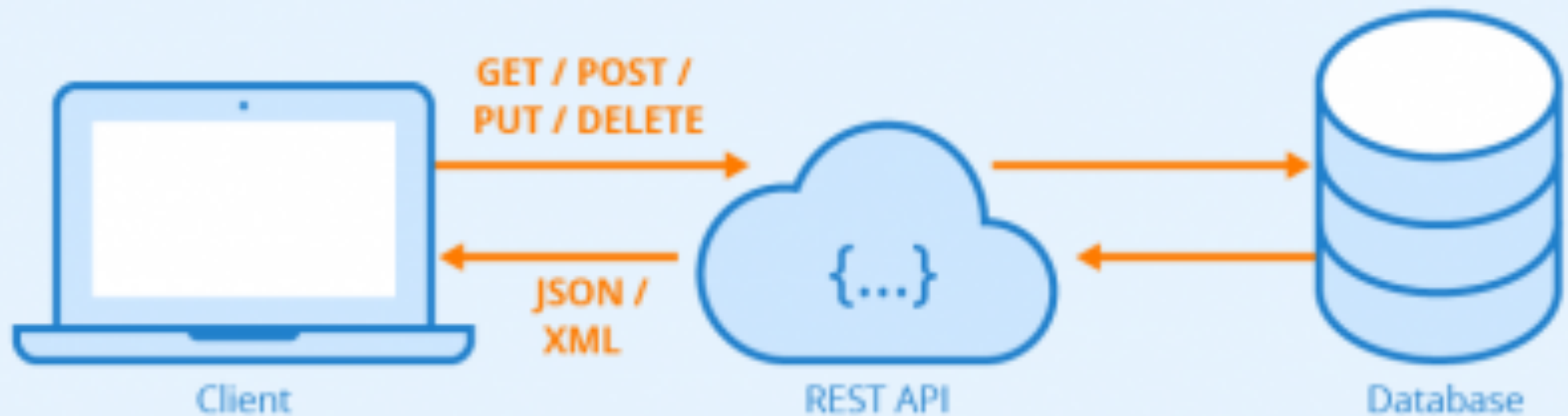
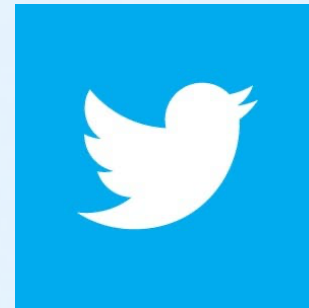
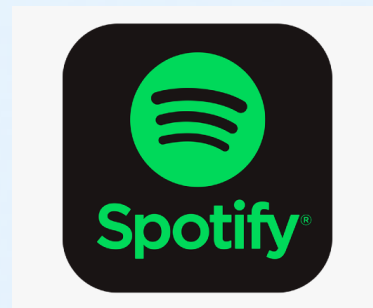
Objeto ordenado en una lista



```
• get_data_JSON List of 17
  abilities : 'data.frame': 2 obs. of 4 variables:
    ..$ is_hidden : logi [1:2] FALSE TRUE
    ..$ slot      : int  [1:2] 1 3
    ..$ ability.name: chr [1:2] "static" "lightning-r...
    ..$ ability.url : chr [1:2] "https://pokeapi.co/a...
  base_experience : int 112
  forms : 'data.frame': 1 obs. of 2 variables:
    ..$ name: chr "pikachu"
    ..$ url : chr "https://pokeapi.co/api/v2/pokemon-...
  game_indices : 'data.frame': 20 obs. of 3 variable...
    ..$ game_index : int [1:20] 84 84 84 25 25 25 25 ...
    ..$ version.name: chr [1:20] "red" "blue" "yellow...
    ..$ version.url : chr [1:20] "https://pokeapi.co/...
  height : int 4
  held_items : 'data.frame': 2 obs. of 3 variables:
    ..$ version_details: List of 2
    .. ..$ : 'data.frame': 10 obs. of 3 variables:
```



Ejemplos Prácticos



Obtención de datos de APIs

Periodismo de Datos

Abril, 2021