



Clase 3

Webscraping y API's

Aproximación a las políticas públicas desde los datos | UC | 09 de junio, 2023

👤 José D. Conejeros | ✉️ jdconejeros@uc.cl

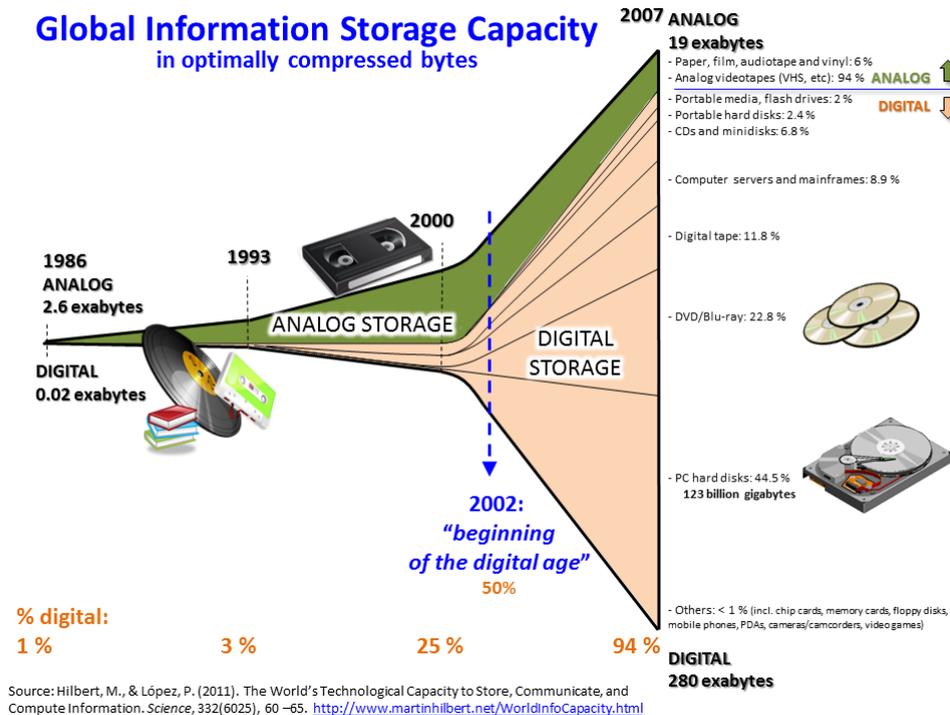
Guía

1. **Extracción de datos vía webscraping**
2. Extracción de datos vía uso de APIS's

La información en internet

La información en Internet es abundante y está al alcance de la mano, pero hay desafíos en:

- Cómo recopilar la información → Webscraping/Api's
- Cómo analizarla la indormación → `tidydata` (lo veran a lo largo del diplomado)



¿Qué es el webscraping?

Es el proceso de extracción de datos estructurados, semi estructurados y no estructurados (HTML) de forma automática desde un sitio web.

- Datos estructurados: datos que tienen un modelo definido o provienen de un campo determinado en un registro. Por ejemplo: el clásico marco de datos con filas y columnas.
- Datos semi estructurados: datos que no tienen formatos fijos, pero contienen atributos o etiquetas. Por ejemplo: correos electrónicos, **páginas HTML**
- Datos no estructurados: datos que no tienen un modelo predefinido o no están organizados de alguna manera. Por ejemplo: videos, fotografías, audio, texto.

Buscamos transformar los datos no estructurados a una estructura conocida del tipo bidimensional: filas y columnas, **aunque no siempre esto es lo más conveniente**. ¿De qué depende?

Algunos ejemplos de uso de webscraping

- Clasificar productos o servicios para crear motores de recomendación, también nos sirve para obtener datos de texto, como en Wikipedia o para hacer sistemas basados en el Procesamiento del Lenguaje Natural.
- Generar datos a partir de etiquetas de imágenes, de sitios web como Google, Flickr, etc. para entrenar modelos de clasificación de imágenes.
- Consolidar los datos de las redes sociales: Facebook y Twitter, para realizar análisis de sentimientos, opiniones o *topic modeling*.
- Extraer comentarios de usuarios y sitios de comercio electrónico como Alibaba, Amazon o Walmart, etc.
- ¡Todas las que puedan imaginar!

Formas de extraer información: DOM

- **Análisis DOM (Documento Objeto Modelo):** interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos en HTML y XML.
- El DOM permite el acceso dinámico a través de la programación para ingresar, añadir y cambiar dinámicamente el contenido estructurado en documentos a través de lenguajes como *JavaScript*.
- Los objetos DOM modelan tanto la ventana del navegador como el historial, el documento o la página web, y todos los elementos que la propia página puede tener: párrafos, divisiones, tablas, formularios y sus campos, etc.
- A través del DOM se puede acceder, mediante *Javascript*, a cualquiera de estos elementos, es decir, a sus correspondientes objetos para alterar sus propiedades o invocar sus métodos.

Formas de extraer información: DOM

The image displays a web browser window on the left and the Chrome DevTools interface on the right. The browser shows a page titled "Detector de riesgos y desastres" with a subtitle "Un seguimiento diario y sistemático de los riesgos y desastres que surgen en Chile, construido a partir de más de 340 medios de noticias locales que cubren todas las regiones del país." Below this, there is a section "Alertas de riesgos y desastres en prensa local" with a data visualization. The visualization consists of three large semi-circular gauges and four smaller circular gauges. The data points are: "Noticias revisadas" (6.885), "Alertas detectadas /30 días" (170), "Regiones con alertas" (15), "refieren a Ecología" (4), "mencionan Peligros" (65), "mencionan Desastres" (26), and "mencionan Eventos" (80). At the bottom, there are filters for "Escoge regiones" and "Escoge un área" (Ecología, Peligros, Desastres, Eventos). The DevTools interface on the right shows the "Elements" pane with the DOM tree expanded to a specific section. The "Styles" pane on the far right shows the CSS styles for the selected element. The console pane at the bottom shows "What's New" and "Issues" sections.

Alertas de riesgos y desastres en prensa local

Categoría	Valor
Noticias revisadas	6.885
Alertas detectadas /30 días	170
Regiones con alertas	15
refieren a Ecología	4
mencionan Peligros	65
mencionan Desastres	26
mencionan Eventos	80

Monitor Social

Formas de extraer información: DOM

Podemos extraer datos del código fuente del sitio web, con un analizador html y expresiones regulares:

```
# Paquete de R que permite la extracción  
install.packages("rvest")  
library(rvest)
```

Puede ver la documentación de la librería dado [click aquí](#)

- Ojo con las condiciones de servicio: generalmente en la parte inferior del sitio web
- Verificar que los datos que se extraen sean de acceso público
- Cuidado con recopilar información personal
- Revisar los derechos de autor

Ojo a la hora de hacer extracciones

1. Respete los deseos del sitio de alojamiento:

Compruebe si existe una API o si los datos están disponibles para descargar

Tenga en cuenta de dónde provienen los datos y dé el crédito respectivo: respete los derechos de autor si desea publicar los datos

Algunos sitios web no permiten extracciones de datos en el archivo `robots.txt` (lo veremos)

2. Limite sus extracciones:

Espere uno o dos segundos después de cada extracción: los servidores web identifican las extracciones de datos

Extraiga solo lo que necesita, y solo una vez (por ejemplo, almacene el archivo html y luego analícelo)

Grandes extracciones puede demandar mucho al servidor. Esto puede acarrear problemas y finalmente las personas a cargo de la plataforma decidan no disponer de los datos de interés.

robots.txt

Es un archivo presente en la mayoría de los sitios web. Contiene el llamado estándar de exclusión de robot que son una serie de instrucciones especialmente dirigidas a los programas que buscan indexar el contenido de estas páginas (por ejemplo el bot de Google que “guarda” las nuevas páginas que se crean en Internet).

- Este método se utiliza para evitar que ciertos bots que analizan sitios de Internet añadan información “innecesaria” a los resultados de las búsquedas.
- Un archivo [robots.txt](#) en una página web funcionará como una petición para que ciertos bots ignoren archivos o directorios específicos en su búsqueda.
- Se aconseja siempre revisar este archivo antes de iniciar el proceso de extracción.

```
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
```

[robots.txt](#)

robots.txt

```
User-agent: *  
Allow: /  
Disallow: /*/alt-*.html  
Disallow: /*/aux-*.html  
Crawl-Delay: 2
```

robots.txt

User-agent: * indica a quién se aplican estas instrucciones. A veces aparece el nombre de algún robot en particular. En este caso, el asterisco explicita que estas instrucciones aplican a todos.

Allow: / indica cuáles son las rutas dentro del sitio desde las que es posible extraer datos. En este caso la barra "/" indica que es posible hacerlo de todas.

Disallow: /*/alt-*.html y **Disallow: /*/aux-*.html** explicitan aquellas rutas dentro del sitio web desde las que no se da autorización para hacer una extracción de forma automatizada. En este caso, son todas aquellas páginas cuya url incluye alt- o aux-. Es decir, acá se está poniendo una restricción a la autorización general que nos dieron en la línea anterior.

Crawl-Delay: 2 señala la cantidad de segundos que es necesario esperar entre cada petición (request) de datos al sitio. Esto es importante tenerlo en cuenta cuando extraemos datos de varias páginas dentro de un sitio web al mismo tiempo, ya que esos segundos de espera permiten que el funcionamiento del sitio no se vea afectado.

Flujo de trabajo en un Webscraping

- a. Investigar y comprender la estructura del sitio web.
- b. Elegir una estrategia de extracción: objetivos y outcomes de interés.
- c. Cree una maqueta del código: extraiga, evalúe y ajuste.
- d. Generalice su maqueta para automatizar la extracción: uso de funciones, bucles for y depuración.
- e. Aplique una exploración y limpieza de su extracción: debe considerar que esto toma bastante tiempo.

Antes de hacer una extracción

Página web=archivo de texto etiquetado

Lenguaje de marcado de hipertexto (HTML): estándar oculto detrás de cada sitio web HTML es texto con estructura marcada, definida por etiquetas:

```
<html>
<head>
  <title>Mi website</title>
</head>
<body>
  <h1 id='first'>Título de mi web</h1>
  <p>Texto & <b>Texto en negrita.</b>
  <img src='myimg.png' width='100' height
  <h2 class="encabezado-lista">Título lis
  <ol>
    <li>Un primer elemento</li>
    <li>Un segundo elemento</li>
    <li>Un tercer elemento</li>
  </ol>
  <h2 class="encabezado-parrafo">Otro títul
</body>
```

Título de mi web

Texto & **Texto en negrita.**



Título lista

1. Un primer elemento
2. Un segundo elemento
3. Un tercer elemento

Otro título

Antes de hacer una extracción

Lo que hacemos al realizar webscraping es importar el código html de un sitio web a nuestro computador (usando R, por ejemplo) y extraer aquellas partes específicas que nos interesan.

- El contenido de un archivo html se encuentra etiquetado, podemos utilizar esas mismas etiquetas para especificar cuáles son las partes de la página que buscamos extraer.
- Las etiquetas sirven para distinguir elementos, clases y, en algunos casos, el id de sus distintos componentes.
- Las etiquetas pueden tener atributos con nombre que se parecen a `name1='value1'` `name2='value2'`. Dos de los atributos más importantes son `id` y `class`, que se utilizan junto con el archivo CSS (hojas de estilo en cascada) lo que permite controlar la apariencia visual de la página.
- Entender la diferencia entre un “elemento” del documento html y una “clase” nos va ayuda a precisar mejor cuál es el contenido exacto que queremos de una página.

Antes de hacer una extracción

Hay más de 100 elementos HTML. Algunas etiquetas comunes:

Elementos del documento: `<head>`, `<body>`, `<footer>`, ...

Componentes del documento: `<title>`, `<h1>`, `<div>`, ...

Estilo de texto: ``, `<i>`, ``, ...

hipervínculos: `<a>`

El webscraping es posible porque la mayoría de las páginas que contienen datos tienen una estructura consistente.

Puede revisar más sobre etiquetado html en: [MDN web docs](#)

Una herramienta útil para identificar selectores: [selectorGadget](#)

Webscraping de páginas dinámicas

Las extracciones funcionan bastante bien con páginas web estáticas donde el sitio web y sus componentes no interactúan con el usuario. Sin embargo, hay páginas web que son dinámicas pues:

- Cambian a medida que el usuario interactúa con él
- Tienen elementos reactivos (por ejemplo, menús desplegables)

En este escenario será necesario controlar el navegador para realizar extracciones en este tipo de páginas web. Algunos paquetes útiles para esto: [RSelenium](#) o [chromote](#).

Esto excede los objetivos del curso, pero pueden encontrar guías de uso:

[Web RSelenium](#)

[Guía general](#)

[Video de instalación](#)

[Video guía de uso](#)

En simple, se iniciará una sesión del navegador a través de R y toda la comunicación se enrutará a partir de dicha sesión.

Guía

1. Extracción de datos vía webscraping
2. **Extracción de datos vía uso de APIS's**

Application Program Interfaces (APIs)

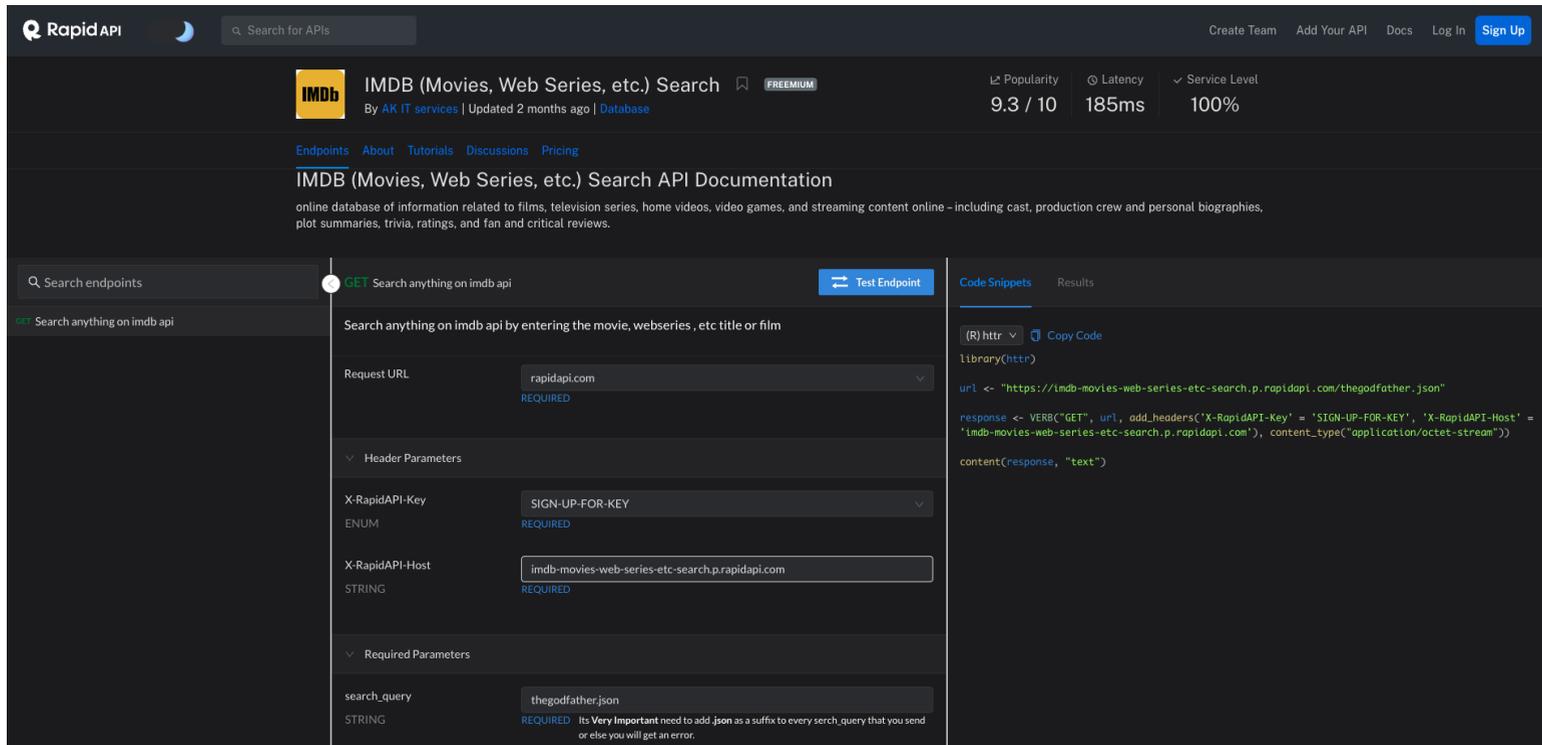
APIs (interfaz de programación de aplicaciones): es un conjunto de definiciones y protocolos utilizados para desarrollar e integrar programas informáticos de aplicación.

- Es un código que indica a las aplicaciones cómo pueden comunicarse entre sí. Las API permiten que sus productos y servicios se comuniquen con otros sin necesidad de saber cómo se implementan.
- Sitios web como Facebook, Twitter, LinkedIn, entre otros, ofrecen una API pública y/o privada a la que se puede acceder mediante programación para recuperar datos en el formato deseado. No todas las páginas webs tienen API's disponibles.
- Para que las APIs públicas sean utilizadas muchos sitios solo permiten a usuarios autorizados (por ejemplo aquellos que tienen una cuenta en la plataforma).
- Siempre que sea posible debe utilizar la API, ya que, por lo general, le brindará datos más confiables.

CUIDADO: revisar los términos y condiciones de extracción, pues existen cargos para tarjetas de crédito si se superan los límites de extracción pública.

Formas de extraer información: APIs

- [httr](#): herramienta general para hacer extracciones de información desde APIs en internet.
- [rapidapi](#): web con varias APIs disponibles para la extracción de datos. Utilizan varios códigos de extracción.



The screenshot shows the RapidAPI interface for the IMDB (Movies, Web Series, etc.) Search API. The interface is dark-themed and includes a search bar at the top. The API details section shows the API name, a 'FREEMIUM' badge, and statistics: Popularity 9.3 / 10, Latency 185ms, and Service Level 100%. Below this, there are links for Endpoints, About, Tutorials, Discussions, and Pricing. The main content area displays the API documentation, which describes it as an online database of information related to films, television series, home videos, video games, and streaming content online. A search bar is present, and a 'Test Endpoint' button is visible. The configuration panel on the right shows the following details:

- Request URL:** rapidapi.com (REQUIRED)
- Header Parameters:**
 - X-RapidAPI-Key:** SIGN-UP-FOR-KEY (ENUM, REQUIRED)
 - X-RapidAPI-Host:** imdb-movies-web-series-etc-search.p.rapidapi.com (STRING, REQUIRED)
- Required Parameters:**
 - search_query:** thegodfather.json (STRING, REQUIRED). A note below states: "Its Very Important need to add .json as a suffix to every serch_query that you send or else you will get an error."

The Code Snippets section shows a sample httr request and response:

```
library(httr)

url <- "https://imdb-movies-web-series-etc-search.p.rapidapi.com/thegodfather.json"

response <- VERB("GET", url, add_headers("X-RapidAPI-Key" = "SIGN-UP-FOR-KEY", "X-RapidAPI-Host" = "imdb-movies-web-series-etc-search.p.rapidapi.com"), content_type("application/octet-stream"))

content(response, "text")
```

API IMDB (Movies, Web Series, etc.) Search

Librerías R

Existen librerías en R con los cuales se pueden trabajar extracciones a partir de API's. Algunos ejemplos:

- [spotifyr](#): para extraer funciones de audio de pistas y otra información de la API web de Spotify de forma masiva
- [WikipediR](#): para extraer información de wikipedia
- [rtimes](#): permite extraer noticias del New York Times
- [ubeR](#): extracción de datos desde UBER
- [Rlinkedin](#): extracción de datos desde LinkedIn
- [googleAnalyticsR](#): extrae información desde Google Analitycs
- [Rbnb](#): acceso a la API de Airbnb
- [ggmap](#): acceso a mapas de google
- [academictwitteR](#): recopilar tweets desde el punto final de la API v2 para el seguimiento del producto de investigación académica
- [rtweet](#): para extraer información desde la API de twitter

Un largo listado se encuentra aquí: [Lista de paquetes que conectan con API's](#).

Notas: Antes de realizar una extracción de datos es siempre bueno verificar si hay un paquete o API disponible.

Aspectos relevantes a la hora de hacer extracciones

- Crearse una cuenta para acceder a la API.
- Al usar API, lea la documentación
 - ¿Hay una opción de descarga por lotes?
 - ¿Hay algún límite de extracción?
 - ¿Desde qué punto se deben pagar los datos?
 - ¿Puedes compartir los datos?

Referencias útiles

Webscraping:

- [Minería de datos web](#)
- [Web scraping R for data science](#)
- [Data collection: Web \(screen\) scraping](#)

Riva Quiroga, "[Introducción al web scraping usando R](#)", Programming Historian en español 6 (2022), <https://doi.org/10.46430/phes0061>.

API's:

- [Using Web APIs from R](#)
- [Accessing Web APIs](#)

Curso de Brian Cooksey [An introduction to APIs](#)



Clase 3

Webscraping y API's

09 de junio, 2023

 **José D. Conejeros** |  jdconejeros@uc.cl |  JDConejeros