Fundamentos de Análisis de Datos

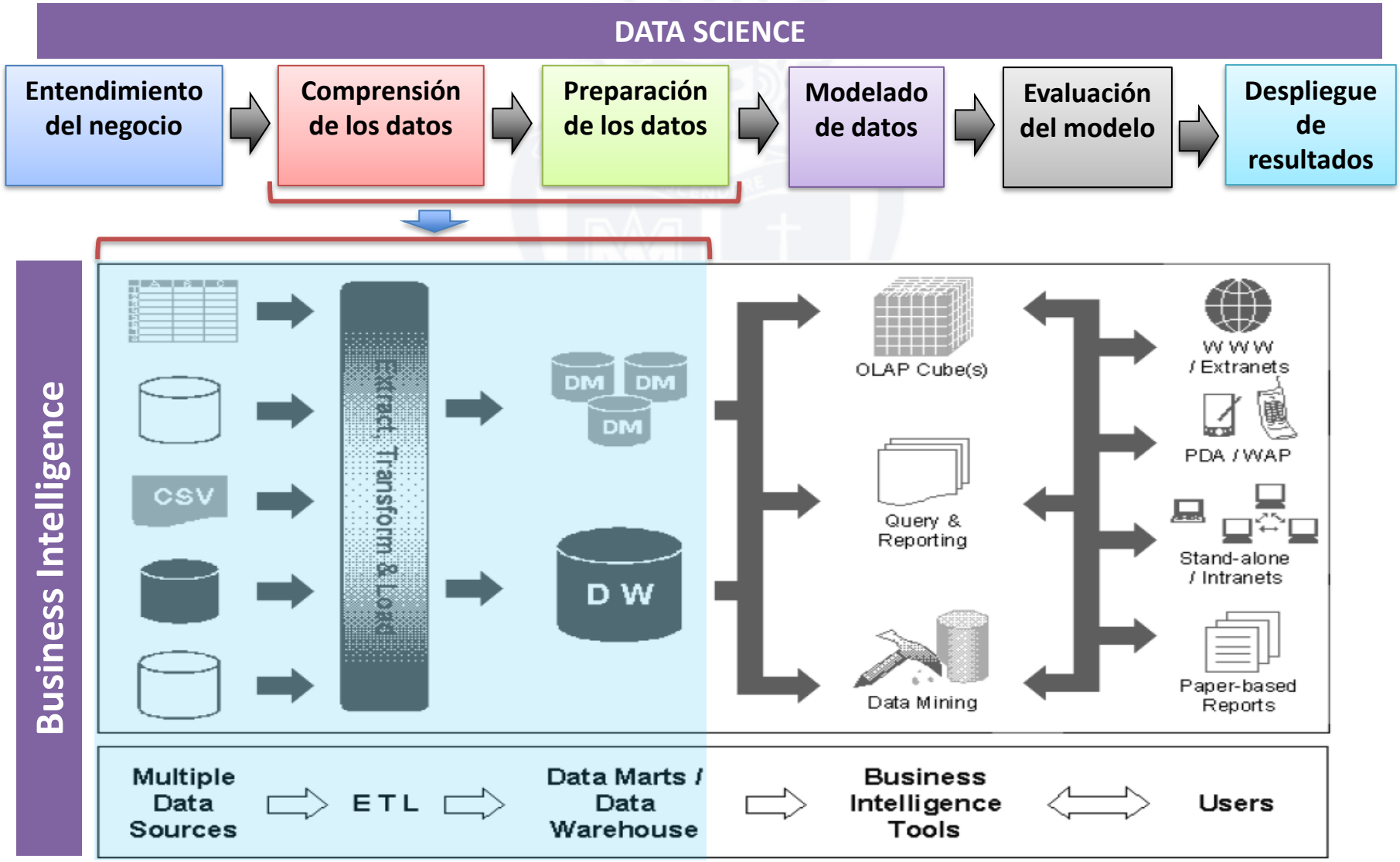
Unidad 3

Extracción de datos

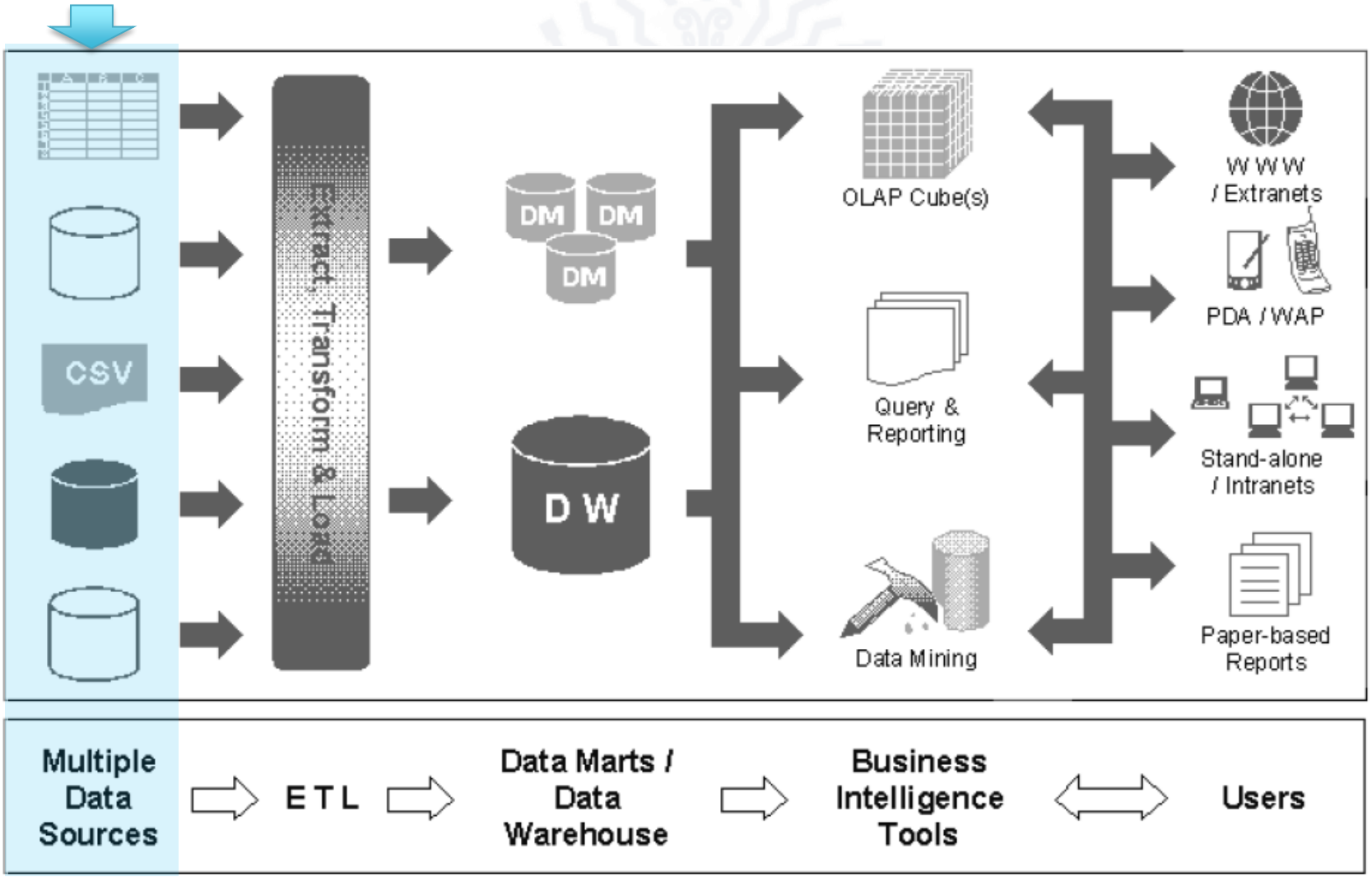
Agenda

- El proceso ETL
- Librerías para extracción y manipulación de datos
- Librerías para el manejo de datos
- Lectura, consumo y visualización de datos
- Técnicas de extracción de datos
  - Extracción desde archivos estructurados

El proceso ETL



Fuentes de datos



Fuentes de Datos

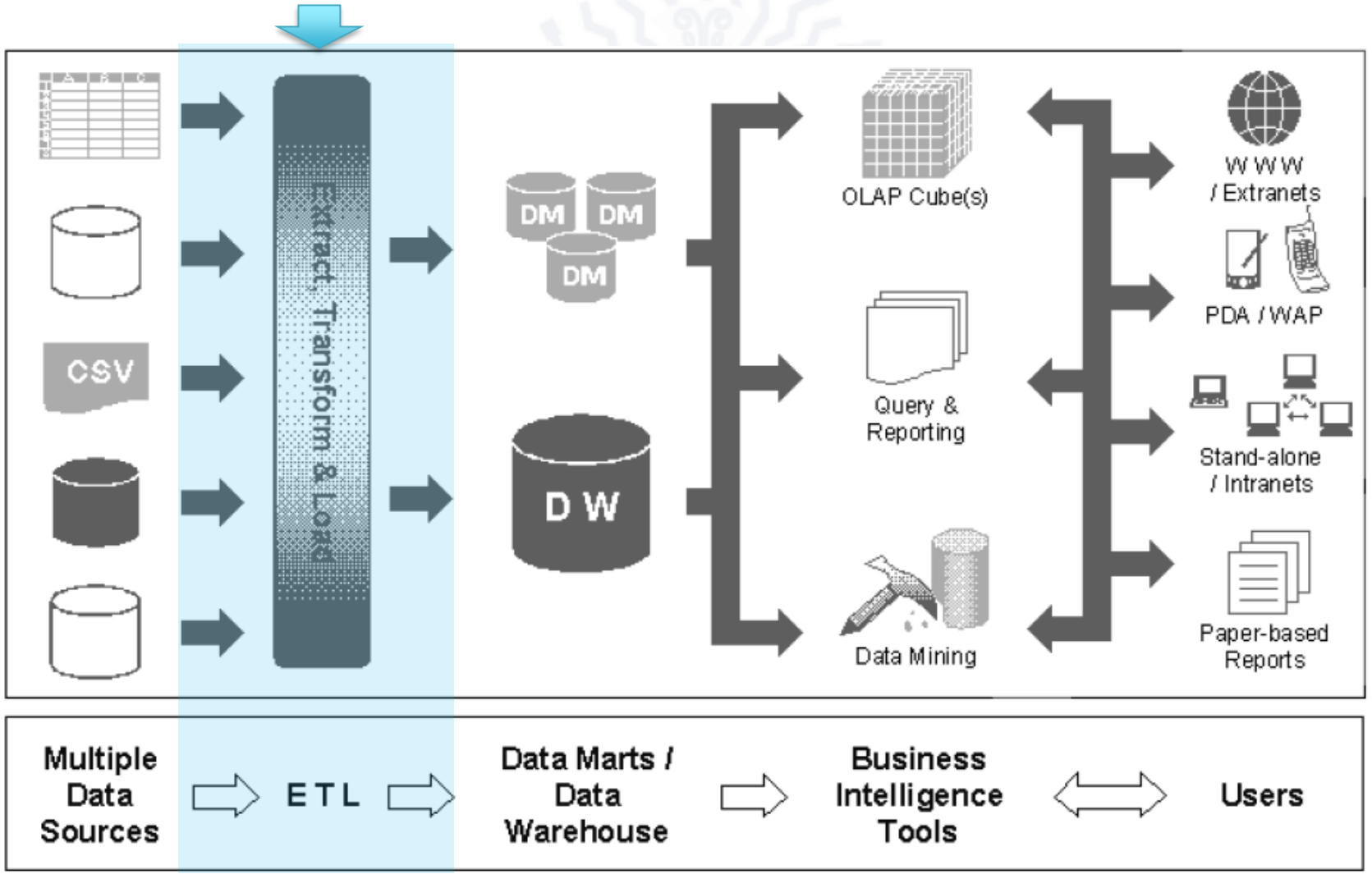
Fuentes:

- Bases de datos, csv, json, APIs, páginas web, archivos de texto, etc.

Otras consideraciones

- Volumen de datos
- Propiedad de los datos
- Confidencialidad la información
- Formato de datos, convenciones de nombramiento, diccionario de datos.

Extracción, Transformación y Carga (ETL)



Extracción, Transformación y Carga (ETL)

EXTRACCIÓN

- Estudio de los datos: Permitirá identificar:
  - Datos fuera de los límites (atípicos).
  - Violaciones a las dependencias (Ej campos derivados).
  - Datos redundantes.
  - Datos huérfanos (integridad referencial).
- Extracción de los datos:
  - En qué formatos se encuentran los datos?
  - Cuál será la frecuencia con la que se extraerán los datos?
  - En qué orden se cargarán los datos?
  - Cómo se minimizará el tiempo requerido para cargar los datos?
- Estacionamiento de los datos
  - Almacén de datos temporal
  - Evitar sobrecarga en las fuentes de datos

Extracción eficiente

Replica fiel de datos origen

Evitar impacto en las fuentes

Extracción, Transformación y Carga (ETL)

TRANSFORMACIÓN

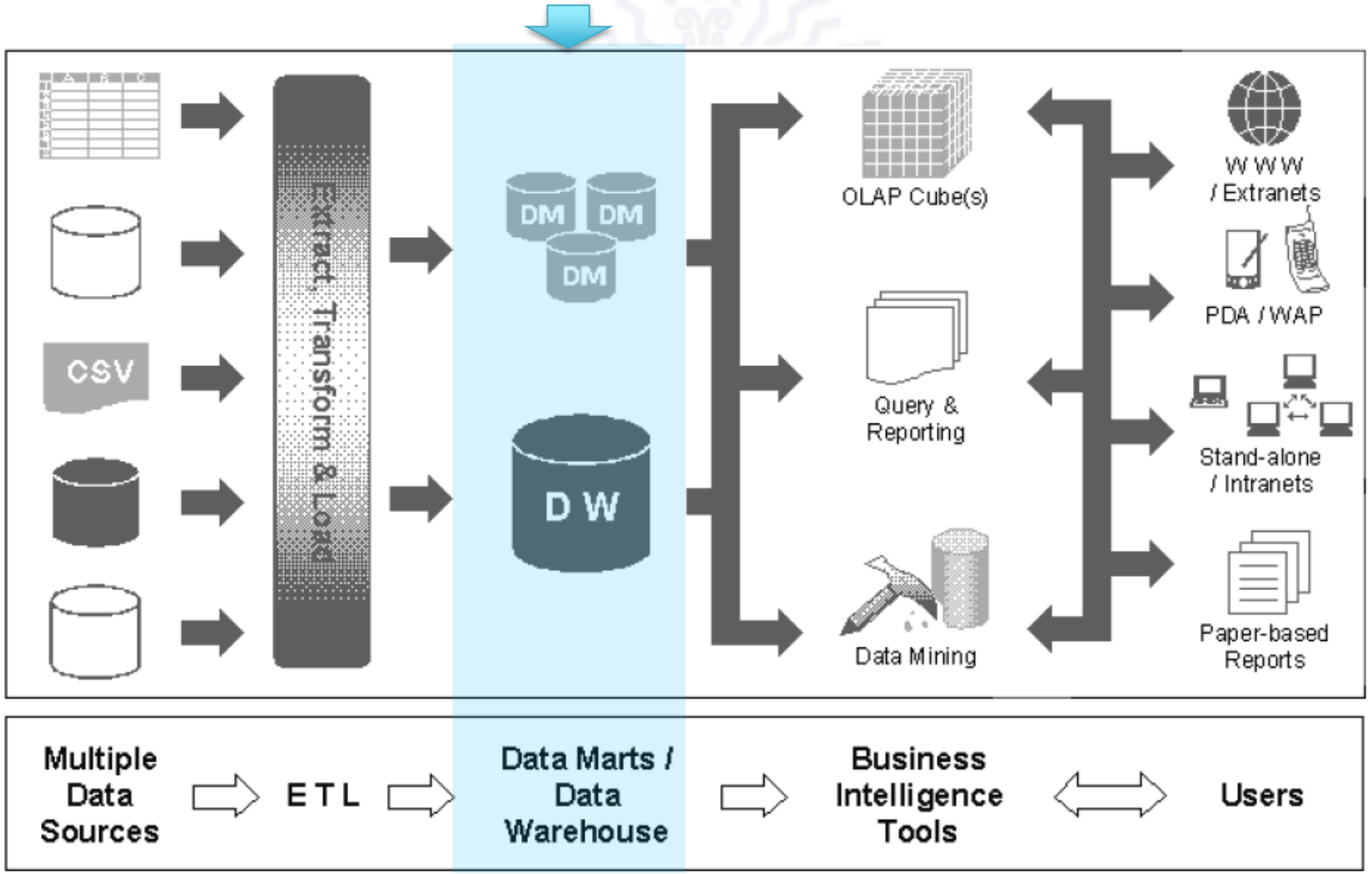
Convertir los datos brutos en un formato más adecuado para análisis

Datos homogéneos libres de inconsistencias

CALIDAD

- Limpieza de los datos
  - Ausencia de valores
  - Valores inconsistentes
  - Valores duplicados
  - Violaciones a las reglas del negocio
- Enriquecimiento de datos
  - Ordenar
  - Dividir
  - Juntar
  - Derivar
  - Buscar y Anexar
- Agrupación de datos (resúmenes)

Almacenamiento integrado de datos



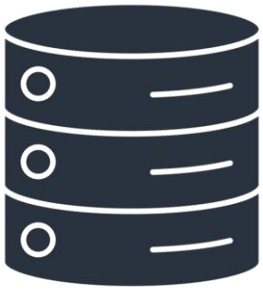
Extracción, Transformación y Carga (ETL)

CARGA

En batch o en tiempo real

Carga de datos en un repositorio central

Debe ser un proceso eficiente



Base de datos transaccional

Data Warehouse

Data Lake

Archivo, o formato propietario

Importancia del proceso ETL

- **Facilita la integración de los datos:** Consolidad datos de múltiples fuentes.
- **Mejora la calidad de los datos:** Limpia, corrige, estandariza, enriquece los datos
- **Soporte de Decisiones**: Alimentan sistemas de inteligencia de negocios y análisis.
- **Mejora del Acceso a los Datos**: Acceso más rápido y eficiente a los datos unificados y organizados.
- **Escalabilidad:** Puede escalar para manejar volúmenes de datos mayores.
- **Eficiencia Operativa**: Automatiza el proceso de mover y transformar datos, reduciendo errores.
- **Cumplimiento y Seguridad de Datos**: Ayuda al cumplimiento de normativas y políticas de seguridad en el tratamiento de datos.

Librerías para extracción y manipulación de datos

- R:
  - **xlsx:** manejo de archivos excel
  - **DBI**: para interactuar con bases de datos en R, lo que facilita la conexión y la ejecución de consultas
  - **httr**: para la extracción de datos de APIs y otros recursos en línea
  - **rvest:** para web scraping
  - **dplyr**: facilita la selección, filtrado de filas, reordenamiento y agregación de datos
  - **tidyr**: permite transformar los datos en un formato más adecuado para análisis. Reorganizar, completar, separar, unir columnas
  - **jsonlite**: Leer y escribir archivos JSON.
  - **lubridate**: Manejar fechas y tiempos.
  - **janitor**: Limpiar nombres de columnas, detectar duplicados y valores inconsistentes.
  - **validate**: Validar reglas de calidad de datos.
  - **stringr**: Operaciones sobre texto (buscar, reemplazar, extraer).

Librerías para extracción y manipulación de datos

- Python:
  - **Pandas**: Es esencial para la manipulación de los datos: limpieza, transformación y análisis
  - **Numpy**: permite manipular estructuras de datos de tipo matriz
  - **requests**: Consumir APIs y manejar peticiones HTTP.
  - **beautifulsoup4**: Web scraping de HTML/XML.
  - **Scrapy**: Framework de web scraping a gran escala.
  - **SQLAlchemy**: ORM para conectar Python con bases de datos relacionales.
  - **openpyxl**: Trabajar con archivos Excel (.xlsx).
  - **json**: Trabajar con JSON nativo en Python.
  - **Selenium**: automatizar la interacción con navegadores web. Web scraping avanzado

Lectura, consumo y visualización de datos

Lectura de datos:

Es el proceso de importar datos almacenados en archivos o bases de datos hacia un entorno de análisis, para trabajar con ellos de manera local.

Consumo de datos:

Es la obtención de datos provenientes de fuentes dinámicas, como APIs o servicios en línea, accediendo a información actualizada bajo demanda o en tiempo real.

Visualización de datos:

Es la transformación de datos en representaciones gráficas, facilitando la identificación de patrones, tendencias y la comunicación efectiva de resultados.

Lectura, consumo y visualización de datos

Concepto	Lectura de datos	Consumo de datos
Qué es	Es el acto técnico de importar datos desde un origen a tu programa o entorno de trabajo.	Es el proceso de adquirir, procesar y actualizar datos, usualmente de fuentes dinámicas o remotas.
Enfoque	Recuperar un archivo o dataset completo para comenzar a trabajar.	Obtener datos que pueden cambiar frecuentemente o que requieren consultas activas.
Ejemplos típicos	Leer archivos CSV, Excel, JSON locales. Leer tablas completas de bases de datos.	Consultar APIs con parámetros. Hacer web scraping de páginas en tiempo real. Actualizar datos periódicamente desde bases de datos o servicios.
Frecuencia	Generalmente puntual: una vez o al iniciar un proyecto.	Puede ser continua, en intervalos o en tiempo real.
Tipo de fuente	Normalmente estática.	Normalmente dinámica.
Objetivo	Traer el conjunto de datos para su posterior análisis.	Integrar datos como parte activa de un flujo de trabajo o aplicación.

Lectura, consumo y visualización de datos

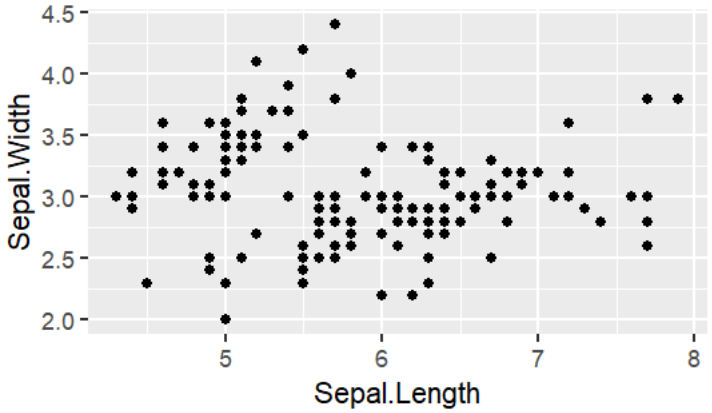
La **visualización de datos** convierte la información en representaciones gráficas para facilitar su comprensión y análisis.

Tipos de visualizaciones comunes:

- **Gráficos de barras**: comparar cantidades.
- **Histogramas**: analizar distribución de datos.
- **Diagramas de dispersión**: identificar correlaciones entre variables.
- **Mapas de calor (heatmaps)**: detectar concentraciones de valores.
- **Gráficos de líneas**: mostrar evolución a lo largo del tiempo.

Herramientas principales (R):

- **ggplot2**: gráficos estáticos de alta calidad.
- **plotly**: gráficos interactivos.
- **leaflet**: mapas interactivos.



Técnicas de extracción de datos

- Extracción de datos desde archivos estructurados
- Extracción de datos desde bases de datos
- Extracción de datos desde archivos semi-estructurados
- Extracción de datos mediante web scraping
- Extracción de datos usando APIs
- Extracción de datos desde archivos SAS

Extracción desde archivos estructurados

- Se pueden extraer datos desde archivos estructurados (ejemplo: csv, excel)
- En R se lo puede hacer mediante funciones que trae el paquete **base**, y mediante otras librerías como **xlsx**.
- Es una de las formas más comunes de compartir y adquirir datos.

Extracción desde archivos estructurados

Ejemplo de extracción de datos desde archivos CSV

```
4  # Configurar el directorio de trabajo
5  setwd("C:/Users/aeenc/RStudio")
6
7  # Leer y cargar el contenido del archivo csv
8  datos_csv <- read.csv("titanic.csv", sep = ",", quote = "\"", dec = ".")
9
10 # Ver el contenido del data frame
11 print(datos_csv)
12 str(datos_csv)
```

Extracción desde archivos estructurados

Ejemplo de extracción de datos desde archivos EXCEL

```
4  # Configurar el directorio de trabajo
5  setwd("C:/Users/aeenc/RStudio")
6
7  # Instalar y cargar librería "xlsx"
8
9  install.packages("xlsx")
10 library(xlsx)
11
12 # Leer y cargar el contenido de la primera hoja del archivo xlsx
13 datos_excel <- read.xlsx("titanic.xlsx", sheetIndex = 1)
14
15 # Ver el contenido del data frame
16 print(datos_excel)
17 str(datos_excel)
18
```

dplyr

dplyr es una librería de R que facilita la **manipulación de datos**. Permite **seleccionar, filtrar, ordenar, agrupar y resumir** datos de manera rápida y legible.

Se puede usar con el operador **%>%** (pipe) para conectar pasos de manipulación de **izquierda a derecha**

Funciones básicas de dplyr

Función	¿Qué hace?
filter()	Filtra filas según condiciones
select()	Selecciona columnas específicas
arrange()	Ordena filas
mutate()	Crea o transforma columnas
summarise()	Resume datos (agrega, promedia, etc.)
group_by()	Agrupa datos para operaciones por grupo

Extracción desde bases de datos

- Conlleva la adquisición de datos desde bases de datos relacionales o no relacionales.
- Requiere establecer una conexión al servicio de base de datos, y el envío de una consulta mediante lenguaje de base de datos (Ej: SQL)
- En R se puede usar la librería **DBI**.
- **DBI** permite trabajar con varios tipos de bases de datos. Puede requerir instalar paquetes adicionales de acuerdo al motor específico.

Extracción desde bases de datos

Ejemplo de extracción de datos desde MySQL

```r
# Instalación y carga de librerias
install.packages("DBI")
install.packages("RMySQL")

library("DBI")
library("RMySQL")

# Conección a una base de datos MySQL
con = dbConnect(RMySQL::MySQL(),
                host = "localhost",
                dbname = "enemdu",
                user = "user01",
                password = "1q2w3e4r",
                port = 3306)

# Construacción de la consulta SQL
comando_sql = paste0("select * ",
                     "from encuestas ",
                     "where cod_provincia = 11 ",
                     "  and genero = \"H\"; ")

# Ejecución de la consulta SQL
datos_mysql <- dbGetQuery(con,comando_sql)

#Cierre de la conexión MySQL
dbDisconnect(con)

View(datos_mysql)
```

Extracción desde bases de datos

Ejemplo de extracción de datos desde Oracle

```
# Instalación y carga de librerias
install.packages("odbc")

library("odbc")

# Búsqueda de un DSN asociado al DBMS a conectar
odbc::odbcListDrivers()

# Conección a Oracle via ODBC
con <- dbConnect(odbc::odbc(),
                 Driver = "Oracle in XE",
                 Server = "localhost",
                 UID = "hr",
                 PWD = "oracle",
                 Port = 1521,
                 SVC = "XE")

# Construacción de la consulta SQL
comando_sql = "select * from employees;"

# Ejecución de la consulta SQL
datos_oracle = dbGetQuery(con,comando_sql)

#Cierre de la conexión MySQL
dbDisconnect(con)

View(datos_oracle)
```

Extracción desde archivos semi-estructurados

- Se pueden también extraer datos desde archivos semiestructurados (Ejemplo: JSON)
- En R se lo puede hacer mediante librerías como **jsonlite**.
- Es otra de las formas más comunes de compartir y adquirir datos.
- JSON se usa también en el contexto de extracción de datos a partir de APIs.

Extracción desde archivos semi-estructurados

Ejemplo 1 de extracción de datos desde archivos JSON

```r
# Instalación y carga de librerias
install.packages("jsonlite")

library(jsonlite)

# Leer y cargar contenido desde archivo json
libros <- fromJSON("LibrosPearson.json")

# Inspeccionar formato y contenido de dataset resultante
class(libros)
View(libros)
str(libros)
libros$area
libros$autor
```

Extracción desde archivos semi-estructurados

Ejemplo 1 de extracción de datos desde archivos JSON

```
[
    {
        "_id": "9786073233316",
        "titulo": "CALCULO UNA VARIABLE",
        "edicion": 13,
        "area": "Matemáticas",
        "año": 2015,
        "autor": "THOMAS"
    },{
        "_id": "9786073235761",
        "titulo": "GEOMETRIA ANALITICA
        "edicion": 4,
        "area": "Matemáticas",
        "año": 2015,
        "autor": "AGUILAR"
    },{
        "_id": "9786073235808",
        "titulo": "GEOMETRIA, TRIGONOM
        "edicion": 4,
        "area": "Matemáticas",
        "año": 2015,
        "autor": "AGUILAR"
    },{
        "_id": "9786073235853",
        "titulo": "CALCULO DIFERENCIAL
        "edicion": 4,
        "area": "Matemáticas",
        "año": 2015, "autor": "AGUILAR
```

LibrosPearson.json

Data frame LIBROS

	_id	titulo	edicion	area	año	autor
1	9786073233316	CALCULO UNA VARIABLE	13	Matemáticas	2015	THOMAS
2	9786073235761	GEOMETRIA ANALITICA	4	Matemáticas	2015	AGUILAR
3	9786073235808	GEOMETRIA, TRIGONOMETRIA Y GEOMETRIA ANALITICA	4	Matemáticas	2015	AGUILAR
4	9786073235853	CALCULO DIFERENCIAL E INTEGRAL	4	Matemáticas	2015	AGUILAR
5	9786073237451	ALGEBRA LINEAL Y SUS APLICACIONES	5	Matemáticas	2016	C. LAY
6	9786073237642	MATEMATICAS 1	1	Matemáticas	2016	c("ESTRADA", "JIMENEZ")
7	9786073237789	TALLER DE LECTURA Y REDACCION 1	3	Humanidades	2016	DE TERESA
8	9786073238021	COMO PROGRAMAR EN JAVA	10	Computación	2016	DEITEL
9	9786073238229	FISICA CONCEPTUAL	12	Ciencias	2016	HEWITT
10	9786073238380	BIOLOGIA	1	Ciencias	2016	GAMA
11	9786073239073	INFORMATICA 2	1	Computación	2016	PÉREZ
12	9786073239233	QUIMICA 2	1	Ciencias	2016	GUTIERREZ
13	9786073240994	MECANICA DE MATERIALES	9	Ingeniería	2017	HIBBELER
14	9786073241984	ORIENTACION EDUCATIVA IV	2	Humanidades	2018	ONOFRE
15	9786073242011	EL PLACER DE LA ESCRITURA MANUAL DE APROPIACION D...	5	Humanidades	2018	CORREA

Extracción desde archivos semi-estructurados

Ejemplo 2 de extracción de datos desde archivos JSON

```
4
5  library(jsonlite)
6
7  # Leer y cargar contenido desde archivo json
8  personas <- fromJSON("personas.json",flatten = TRUE)
9
10 # Inspeccionar formato y contenido de dataset resultante
11 class(personas)
12 View(personas)
13 str(personas)
```

Extracción desde archivos semi-estructurados

Ejemplo 2 de extracción de datos desde archivos JSON

```json
[{
  "cedula": "1111111111",
  "nombre": "juan",
  "telefono": [
    "0978786767",
    "0728938293"
  ],
  "sueldo": 1200
},{
  "cedula": "2222222222",
  "nombre": "maria",
  "telefono": [
    "027428483",
    "0983764552"
  ],
  "sueldo": 1230.34,
  "titulo": {
      "denominacion": "Arquitecto",
      "anio": 1990
  }
},{
  "cedula": "3333333333",
  "nombre": "pedro",
  "telefono": "0498263541",
  "titulo": {
      "denominacion": "Ingeniero en informática",
      "anio": 2000
  },
```

Personas.json

Data frame PERSONAS

	cedula	nombre	telefono	sueldo	email	titulo.denominacion	titulo.anio
1	1111111111	juan	c("0978786767", "0728938293")	1200.00	*NA*	*NA*	*NA*
2	2222222222	maria	c("027428483 ", "0983764552")	1230.34	*NA*	Arquitecto	1990
3	3333333333	pedro	0498263541	*NA*	pedro@mail.com	Ingeniero en informática	2000

Extracción mediante web scraping

- Es posible extraer datos desde una página web mediante la extracción de elementos de la página, con base en la lectura de nodos dentro del código HTML.
- Se usa cuando no podemos obtener los datos mediante una descarga en formatos convencionales (CSV, JSON, etc.), o mediante APIs.
- En R para este proceso se pueden usar funciones que vienen con la librería "**rvest**"
- Web scraping debe realizarse considerando las leyes de privacidad y derechos de autor, así como los términos de servicio del sitio web.
- Debe realizarse de manera responsable evitando impacto en las fuentes

Extracción mediante web scraping

Principales funciones rvest

Función	Descripción breve
read_html()	Lee el HTML de una página web o archivo local
html_elements()	Extrae múltiples nodos HTML que coinciden con un **selector**
html_element()	Extrae el primer nodo que coincide con un **selector**
html_text() / html_text2()	Extrae el texto interno del nodo HTML
html_attr()	Extrae el valor de un atributo HTML (ej. href, src, title)
html_table()	Convierte tablas HTML en data.frames

Extracción mediante web scraping

¿Qué es un selector CSS?

- Un **selector CSS** es una regla que permite identificar elementos dentro del código HTML de una página web, basándose en atributos como **etiqueta (tipo)**, **clase**, **ID**, o **posiciones estructurales**.
- En el contexto de **rvest en R**, un selector CSS es una **cadena de texto** que le indica al programa **qué elementos del dominio queremos extraer**. Funciona como una **guía precisa** para señalar contenido como títulos, listas de precios, enlaces, imágenes, entre otros.
- Es equivalente a decirle a R:
  - "Encuentra todos los elementos <h1>",
  - "Extrae todas las imágenes con clase producto", o
  - "Tráeme los enlaces dentro de una tabla".
- Un selector CSS en rvest actúa como una **fórmula estructurada para localizar elementos HTML**, y se utiliza en funciones como html_nodes() o html_elements().

Extracción mediante web scraping

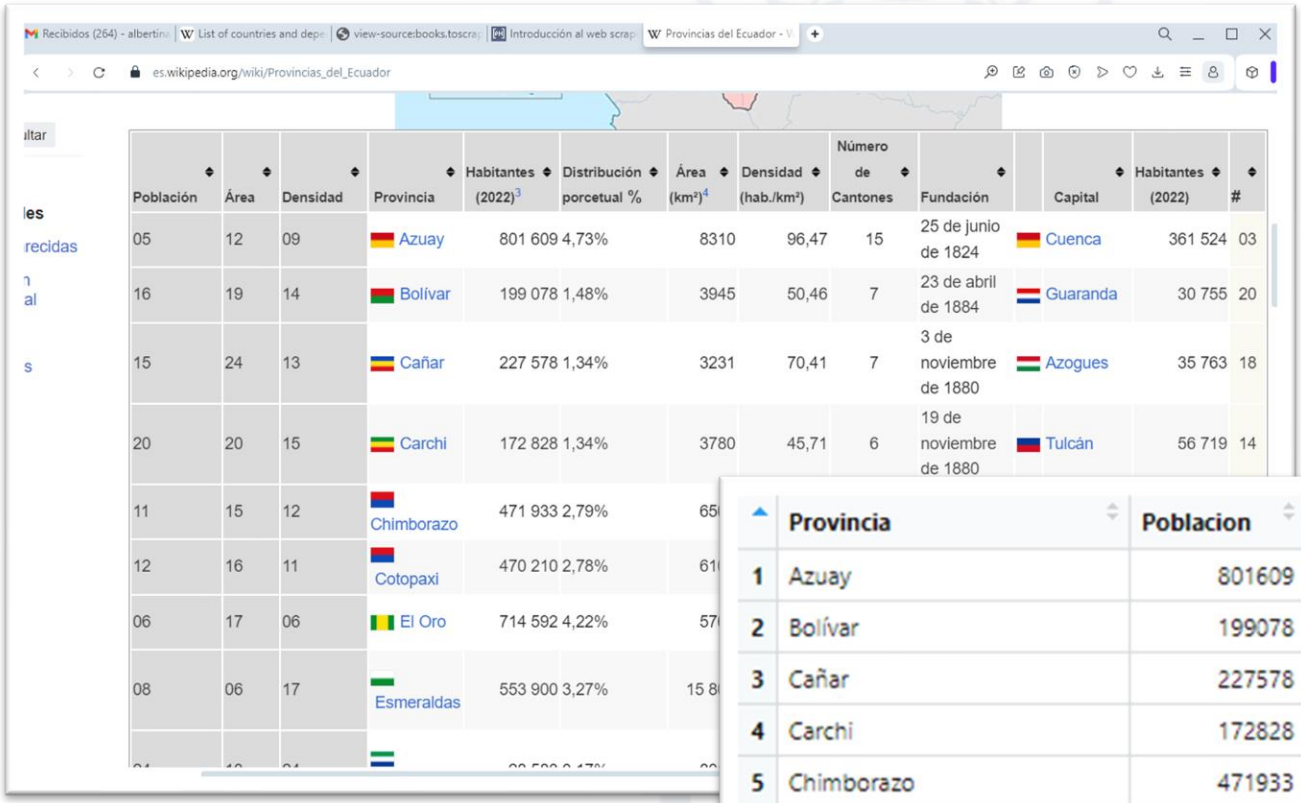
Ejemplos de selectores CSS

Selector CSS	Qué selecciona
`h1`	Todos los encabezados <h1>
`.precio`	Todos los elementos con clase precio
`#producto123`	El elemento con ID producto123
`div > p`	Todos los párrafos <p> que son hijos directos de un
`ul li a`	Todos los enlaces dentro de listas
`img[src$=".jpg"]`	Todas las imágenes cuyo atributo src termina en .jpg
`a[href*="producto"]`	Enlaces con href que contienen la palabra "producto"
`div[class^="item"]`	Todos los div cuya clase comienza con item

Ejemplo Web Scraping con R

https://es.wikipedia.org/wiki/Provincias_del_Ecuador

EJEMPLO 1



Población	Área	Densidad	Provincia	Habitantes (2022)[3]	Distribución porcetual %	Área (km²)[4]	Densidad (hab./km²)	Número de Cantones	Fundación		Capital	Habitantes (2022)	#
05	12	09	Azuay	801 609	4,73%	8310	96,47	15	25 de junio de 1824		Cuenca	361 524	03
16	19	14	Bolívar	199 078	1,48%	3945	50,46	7	23 de abril de 1884		Guaranda	30 755	20
15	24	13	Cañar	227 578	1,34%	3231	70,41	7	3 de noviembre de 1880		Azogues	35 763	18
20	20	15	Carchi	172 828	1,34%	3780	45,71	6	19 de noviembre de 1880		Tulcán	56 719	14
11	15	12	Chimborazo	471 933	2,79%	65							
12	16	11	Cotopaxi	470 210	2,78%	61							
06	17	06	El Oro	714 592	4,22%	57							
08	06	17	Esmeraldas	553 900	3,27%	15 8							

Data frame PROVINCIAS

	Provincia	Poblacion	Area	Densidad	Cantones
1	Azuay	801609	8310	96.47	15
2	Bolívar	199078	3945	50.46	7
3	Cañar	227578	3231	70.41	7
4	Carchi	172828	3780	45.71	6
5	Chimborazo	471933	6500	72.61	10
6	Cotopaxi	470210	6108	76.98	7
7	El Oro	714592	5767	123.92	14
8	Esmeraldas	553900	15809	35.04	7
9	Galápagos	28583	8010	3.57	3

Extracción mediante web scraping

EJEMPLO 1

```r
# EJEMPLO WEB SCRAPING CON R
# (Usando librería rvest)

# Obtener información de las provincias del ecuador a partir de
# información disponible en Wikipedia

# Cargar las bibliotecas necesarias
install.packages("rvest")
install.packages("xml2")

library(rvest)
library(xml2)
library(dplyr)

# Definir la URL
url <- "https://es.wikipedia.org/wiki/Provincias_de_Ecuador"

# Leer el contenido de la página
pagina = read_html(url)

# Extraer tablas de la página
tablas = html_table(pagina)

# Asumiendo que la tabla de interés es la primera
# que contiene las Provincias de Ecuador
tabla_provincias = tablas[[1]]
```

Extracción mediante web scraping

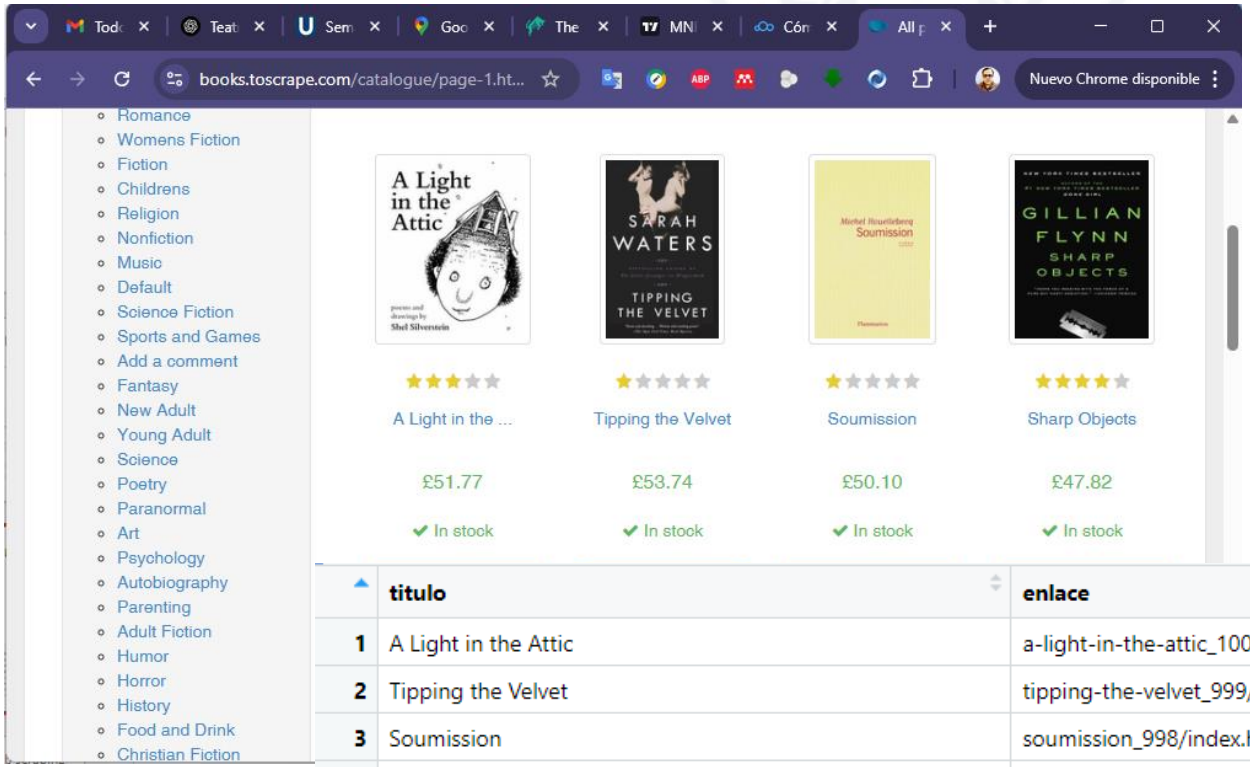
EJEMPLO 1

```r
# Inspeccionar la estructura del data frame
names(tabla_provincias)
str(tabla_provincias)

# Extraer únicamente información de nombre de la provincia,
# población, superficie, densidad poblacional, y número de cantones
provincias = tabla_provincias %>%
  select(Provincia, Poblacion = 5, Area = 7, Densidad = 8, Cantones = 9)

str(provincias)

# Eliminar caracteres distintos a digitos en columnas numéricas enteras,
# y reemplazar la coma decimal por punto en columnas numéricas decimales,
# y convertir a valores enteros o decimales.
provincias = provincias %>%
  mutate(Poblacion = as.integer(gsub("[^0-9]","",Poblacion)),
         Area = as.integer(gsub("[^0-9]","",Area)),
         Densidad = as.numeric(gsub(",",".",Densidad)))

str(provincias)
print(provincias)
```

Ejemplo Web Scraping con R

https://books.toscrape.com/catalogue/page-1.html

EJEMPLO 2



Data frame LIBROS

	titulo	enlace	precio
1	A Light in the Attic	a-light-in-the-attic_1000/index.html	51.77
2	Tipping the Velvet	tipping-the-velvet_999/index.html	53.74
3	Soumission	soumission_998/index.html	50.10
4	Sharp Objects	sharp-objects_997/index.html	47.82
5	Sapiens: A Brief History of Humankind	sapiens-a-brief-history-of-humankind_996/index.html	54.23
6	The Requiem Red	the-requiem-red_995/index.html	22.65
7	The Dirty Little Secrets of Getting Your Dream Job	the-dirty-little-secrets-of-getting-your-dream-job_994/inde...	33.34
8	The Coming Woman: A Novel Based on the Life of the Infam...	the-coming-woman-a-novel-based-on-the-life-of-the-infa...	17.93

Extracción mediante web scraping

EJEMPLO 2

```r
# Obtener lista de titulos y enlaces de libros publicados
# en el sitio books.toscrape.com

# Cargar librerias necesarias
library(rvest)
library(xml2)

# Leer el contenido de la página web
url <- "https://books.toscrape.com/catalogue/page-1.html"
pagina <- read_html(url)

# Extraer información de los contenedores que contienen la información de los
# libros utilizando selectores CSS
nodos = pagina %>%
  html_elements(".product_pod")

# Extraer titulos, enlaces, precio, e imágeness de los libros
titulos = nodos %>%
  html_element("h3 a") %>%
  html_attr("title")

enlaces = nodos %>%
  html_element("h3 a") %>%
  html_attr("href")
```

Extracción mediante web scraping

EJEMPLO 2

```
precios = nodos %>%
  html_element(".price_color") %>%
  html_text2()

imagenes = nodos %>%
  html_element(".image_container a img") %>%
  html_attr("src")

imagenes = gsub("\\.\\.","",imagenes)

# Combinar vectores en un solo data frame
libros = data.frame(titulo = titulos,
                    enlace = paste0("http://books.toscrape.com/catalogue/",enlaces),
                    precio = precios,
                    imagen = paste0("http://books.toscrape.com",imagenes))

# Corregir el formato del precio de los libros
libros = libros %>%
  mutate(precio = as.numeric(gsub("£","",precio)))
str(libros)

# Inspeccionar el data frame final resultante
print(libros)

```

Extracción mediante web scraping

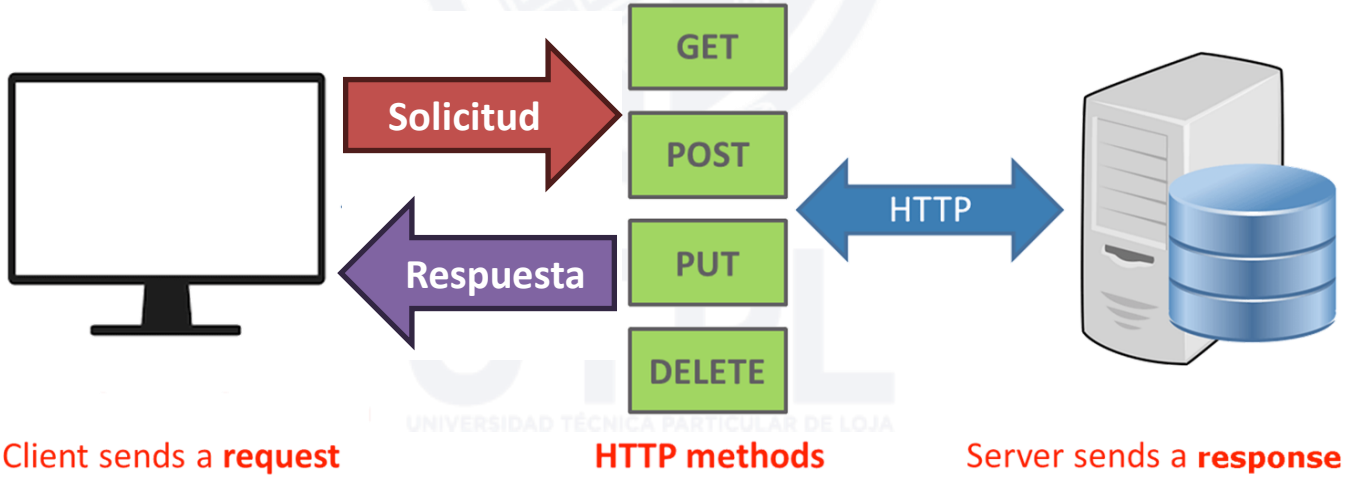
- Recursos recomendados
  - Web scraping in R (https://www.r-bloggers.com/2023/01/web-scraping-in-r-2/)
- Revisar más acerca de:
  - CSS selectors
  - XPath

Ejercicio Web Scraping

- Ir al sitio de Alpha Editorial (https://www.alphaeditorialcloud.com/)
- Realizar una búsqueda de libros bajo cualquier criterio.
- Del resultado de la búsqueda extraer:
  - Titulo del libro
  - Fecha
  - Autor
  - Precio
  - URL de la imagen portada del libro

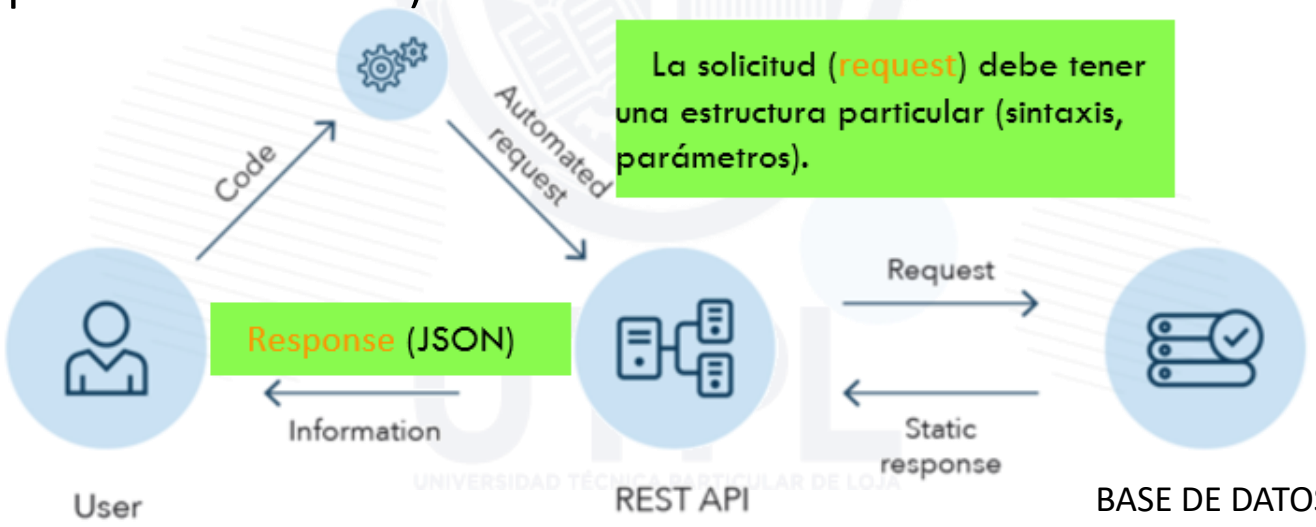
Extracción mediante APIs

- **API** (Application Programming Interface) es la interfaz que permite la comunicación (intercambio de datos) entre dos aplicaciones de software independientes a través de un conjunto de reglas,
- Una API se puede implementar mediante un **servicio web** que otorga acceso a datos y métodos específicos a los que otras aplicaciones pueden acceder, y en ocasiones editar, a través de protocolos **HTTP** estándar.



Extracción mediante APIs

- **REST** (REpresentational State Transfer) es el estilo arquitectónico más popular de API para servicios web que ofrecen los proveedores de datos.
- **REST** se refiere a un conjunto de pautas diseñadas para simplificar la comunicación cliente/servidor. Las API REST hacen que el acceso a los datos sea mucho más sencillo y lógico.
- Pueden ser privadas (empresa), abiertas sólo para partners, o públicas (cualquier desarrollador).



Extracción mediante APIs

¿Qué es una API REST o API RESTful?

https://youtu.be/gUeyJ9xDq-Y?si=PcZ9WIfFsPF9APJu

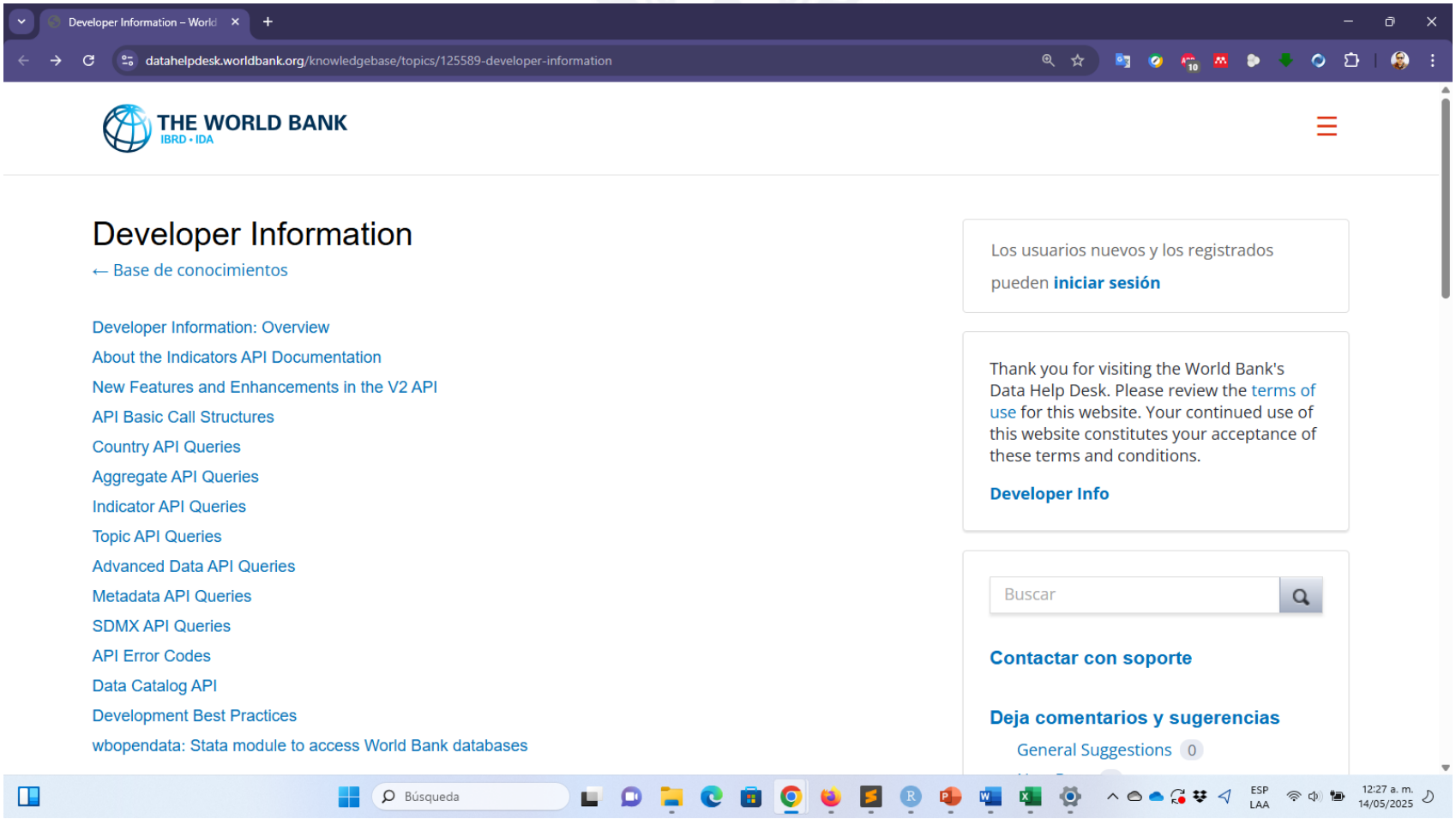
REST => R (read)
RESTful => CRUD (Create, Read, Update, Delete)

Extracción mediante APIs

- En R se utiliza la librería "**httr**" que facilita la comunicación con servicios API RESTful.
- También se requiere usar la librería "**jsonlite**" que permite trabajar con archivos JSON, que es el formato comúnmente usado para recepción o envío de datos a través de APIs.
- Se requiere revisar la documentación de la API para conocer su tipo, restricciones, parámetros, estructura del REQUEST, etc.
- Ejemplo:
  - API: Banco Mundial
  - EndPoint: http://api.worldbank.org/V2
  - Documentación: https://datahelpdesk.worldbank.org/knowledgebase/topics/125589

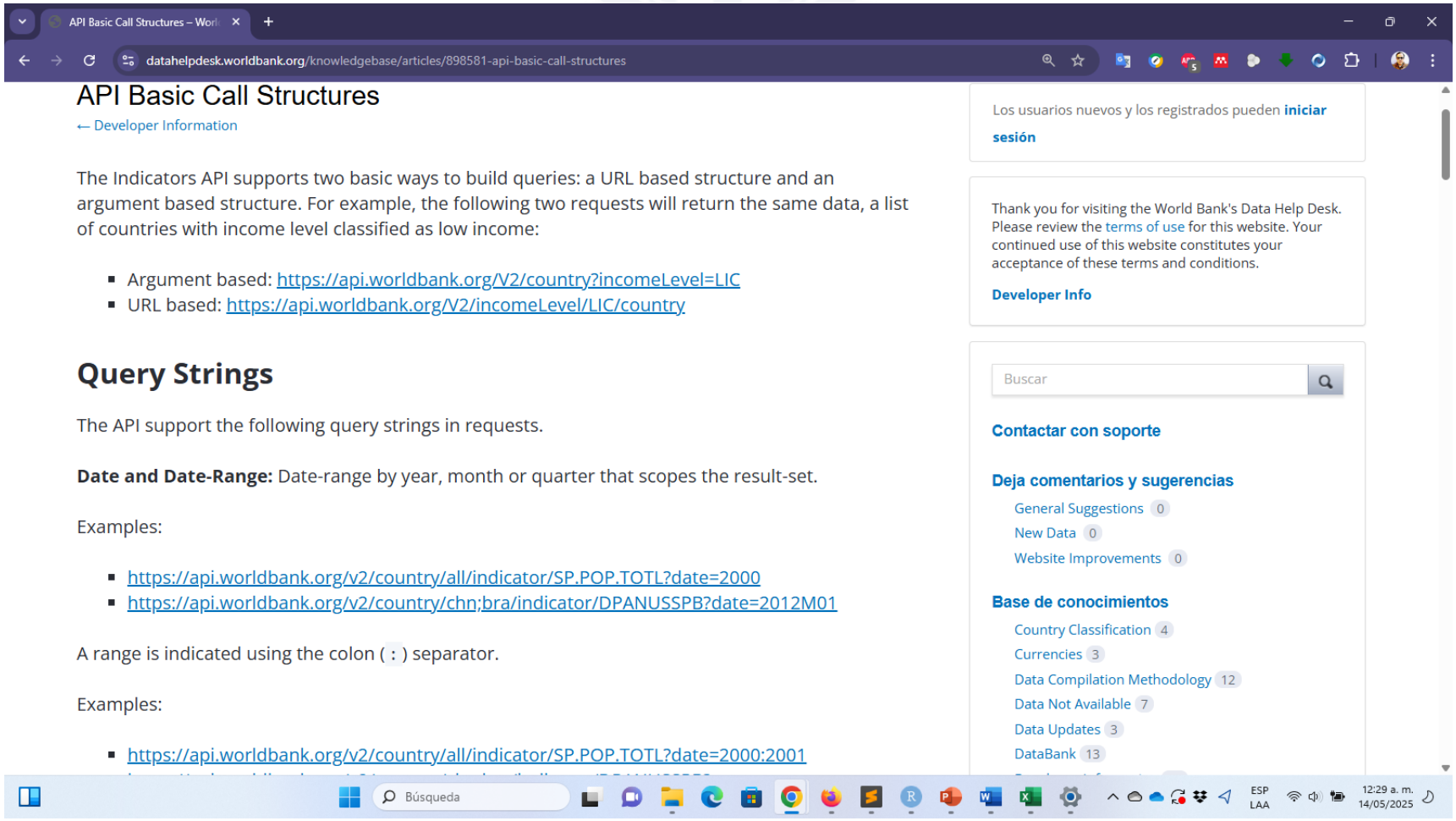
Extracción mediante APIs

Documentación



Extracción mediante APIs

Documentación



Extracción mediante APIs

- Pasos
  1. Construir la URL de la llamada a la API
  2. Realizar una solicitud GET()
  3. Verificar el estado de la respuesta:
     - Se espera siempre un status 200-OK
     - Otros status pueden ser: 400-Bad Request, 401-Unauthorized, 403-Forbidden, 404-Not Found
  4. Traducir la respuesta a un formato utilizable, como un data frame.

Extracción mediante APIs

- **GET()** genera un objeto ***response***, que contiene toda la información del *request*, que incluye:
  - La **url** a la que finalmente se envió la solicitud
  - El **status_code** HTTP
  - Una referencia interna (**handle**) usada por *httr* para gestionar la solicitud.
  - Los encabezados HTTP (**headers**) de la respuesta. Por ejemplo, tipo de contenido, permisos, etc.
  - El contenido o cuerpo de la respuesta (**content**), en un formato determinado (XML, JSON, etc.)
  - Información fecha y hora (**time**) de la solicitud
  - …

Extracción mediante APIs

EJEMPLO

- Usando la API del Banco Mundial (http://api.worldbank.org/), # extraer información países del mundo
- URL base: http://api.worldbank.org/v2/country
- Parámetros principales del REQUEST: **region** (región geográfica: LCN, EAS, ECS, …), **incomeLevel** (nivel de ingresos: LIC, LMC, UMC, HIC), **format** (json, xml), **per_page** (cantidad observaciones).
- Documentación:
  - https://datahelpdesk.worldbank.org/knowledgebase/articles/898590-country-api-queries
  - https://api.worldbank.org/v2/region?format=json
  - https://api.worldbank.org/v2/incomeLevel?format=json

Extracción mediante APIs

EJEMPLO

```r
# Usando la API del Banco Mundial (http://api.worldbank.org/),
# extraer información de todos los paises del mundo

# Instalar y cargar librerias
install.packages("httr")

library(httr)
library(jsonlite)
library(dplyr)

# Definir la URL base y parámetros para la API del World Bank
url_base <- "http://api.worldbank.org/v2/country"

region <- "CLA"
nivel_ing <- "UMC"
formato <- "json"
cant_obs <- 1000

# Construir la URL completa
url <- paste0(url_base,
              "?region=",region,
              "&incomeLevel=",nivel_ing,
              "&format=",formato,
              "&per_page=",cant_obs)
```

URL: http://api.worldbank.org/v2/country?region=CLA&incomeLevel=UMC&format=json&per_page=100

Extracción mediante APIs

EJEMPLO

```r
# Realizar la solicitud GET
respuesta <- GET(url)
class(respuesta)

# Verificar el estado de la respuesta (se espera status HTTP 200)
status_code(respuesta)

# Si la solicitud fue exitosa, extraer y procesar los datos

# Extraer el contenido del response (en formato JSON)
contenido_json <- content(respuesta,"text")

contenido_R <- fromJSON(contenido_json, flatten = TRUE)

# Se extrae el 2do elemento de la lista que contiene el data frame
# con los datos solicitados.
paises <- contenido_R[[2]]

# Mostrar los datos
print(paises)
```

Extracción usando APIs

EJEMPLO


```
[
  ▾ {
        page: 1,
        pages: 1,
        per_page: "1000",
        total: 18
    },
  ▾ [
      ▾ {
            id: "ARG",
            iso2Code: "AR",
            name: "Argentina",
          ▾ region: {
                id: "LCN",
                iso2code: "ZJ",
                value: "Latin America & Carib
            },
          ▾ adminregion: {
                id: "LAC",
                iso2code: "XJ",
                value: "Latin America & Carib
            },
          ▾ incomeLevel: {
                id: "UMC",
                iso2code: "XT",
                value: "Upper middle income"
            },
          ▾ lendingType: {
                id: "IBD",
                iso2code: "XF",
                value: "IBRD"
            },
            capitalCity: "Buenos Aires",
            longitude: "-58.4173",
            latitude: "-34.6118"
```


	id	iso2Code	name	capitalCity	longitude
1	ARG	AR	Argentina	Buenos Aires	-58.4173
2	BLZ	BZ	Belize	Belmopan	-88.7713
3	BRA	BR	Brazil	Brasilia	-47.9292
4	COL	CO	Colombia	Bogota	-74.082
5	CRI	CR	Costa Rica	San Jose	-84.0089
6	DMA	DM	Dominica	Roseau	-61.39
7	DOM	DO	Dominican Republic	Santo Domingo	-69.8908
8	ECU	EC	Ecuador	Quito	-78.5243
9	GRD	GD	Grenada	Saint George's	-61.7449
10	GTM	GT	Guatemala	Guatemala City	-90.5328
11	JAM	JM	Jamaica	Kingston	-76.792
12	LCA	LC	St. Lucia	Castries	-60.9832
13	MEX	MX	Mexico	Mexico City	-99.1276
14	PER	PE	Peru	Lima	-77.0465
15	PRY	PY	Paraguay	Asuncion	-57.6362
16	SLV	SV	El Salvador	San Salvador	-89.2073

Extracción usando APIs

- Lista de APIs públicas:

https://github.com/public-apis/public-apis

Ejercicio extracción con APIs

- Usando la API que proporciona la plataforma OpenWeather (https://api.openweathermap.org/): Extraer datos del pronóstico del tiempo de los próximos 5 días, para un lugar específico.
- URL base: https://api.openweathermap.org/data/2.5/forecast
- Documentación de la API: https://openweathermap.org/forecast5
- La API requiere una llave (API-Key), usar: **4afd29be646ee3c7027bea84dc25ce84**
- Usar el menos 4 parámetros y extraer información correspondiente a la hora, temperatura mínima, temperatura máxima, nivel de lluvia, y tipo de clima.