Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación


# IIC2115 - Programación como Herramienta para la Ingeniería

## Extracción y visualización de datos

**Profesor:** Hans Löbel

# Este capítulo final del curso apunta a dos problemas muy comunes en la práctica

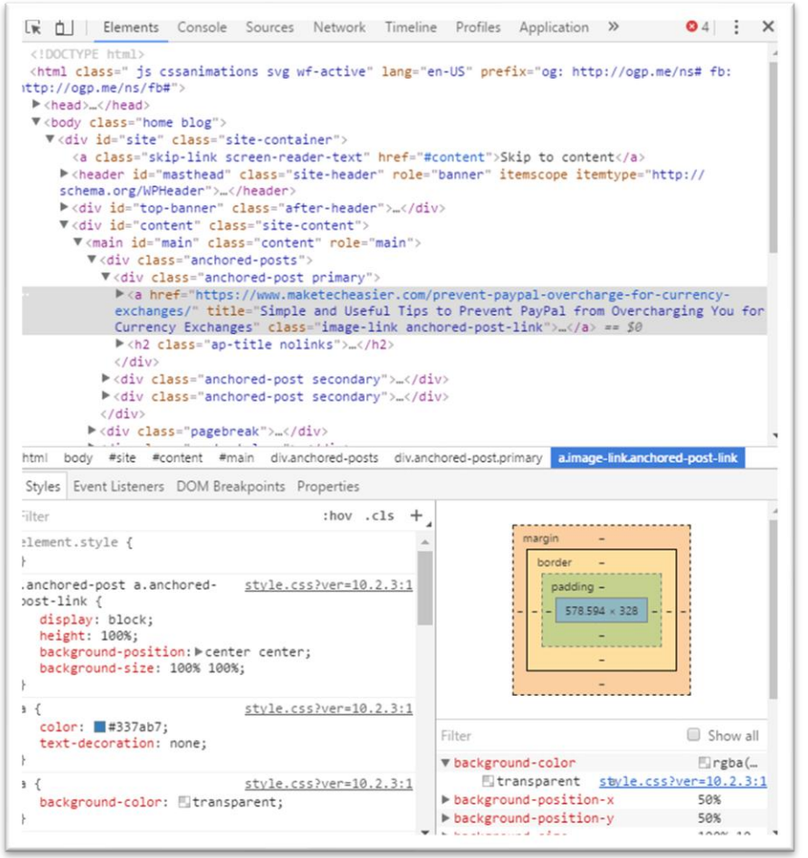
- Muchas veces, por no decir todas, los datos no están disponibles en archivos o fuentes fácilmente accesibles.
- Luego, una vez que los tenemos disponibles y preparados, su visualización efectiva y atractiva demanda mucho tiempo de preparación.
- Para enfrentar ambas situaciones, en este capítulo cubriremos:
  - Descarga de información desde la Web, utilizando diversas técnicas.
  - Visualización de datos a través de dashboards rápidamente desplegables en Python.

# Web Scraping es el esquema más común para extraer información desde la web de manera automática

- Definición: proceso de extracción de datos de páginas web de manera automatizada.
- Objetivo: recopilar información que no está disponible de manera directamente para una máquina.
- Aplicaciones: análisis de mercado, recopilación de datos para proyectos de machine learning, monitoreo de precios, entre otros.

# Pasos Simples para Web Scraping

- Seleccionar el sitio web: identificar la página que tiene los datos necesarios.
- Buscar los datos: inspeccionar el código HTML para localizar los elementos que contienen los datos deseados.
- Extraer los datos: usar un lenguaje de programación como Python para extraer los datos.
- Guardar los datos: almacenar la información en un formato útil, como CSV o JSON.
- Procesar y consultar los datos: utilizar herramientas como Pandas para limpiar, transformar y analizar la información extraída.



# Web Scraping en Python

- BeautifulSoup (BS4): Para analizar HTML y extraer información específica.

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd

url = 'https://ejemplo.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Extraer datos
titles = [title.get_text() for title in soup.find_all('h2')]
# Organizar y guadrar con pandas
df = pd.DataFrame(titles, columns=['Title'])
df.to_csv('output.csv', index=False)
```

- Pandas: Para organizar, limpiar, analizar y también extraer datos de tablas HTML.

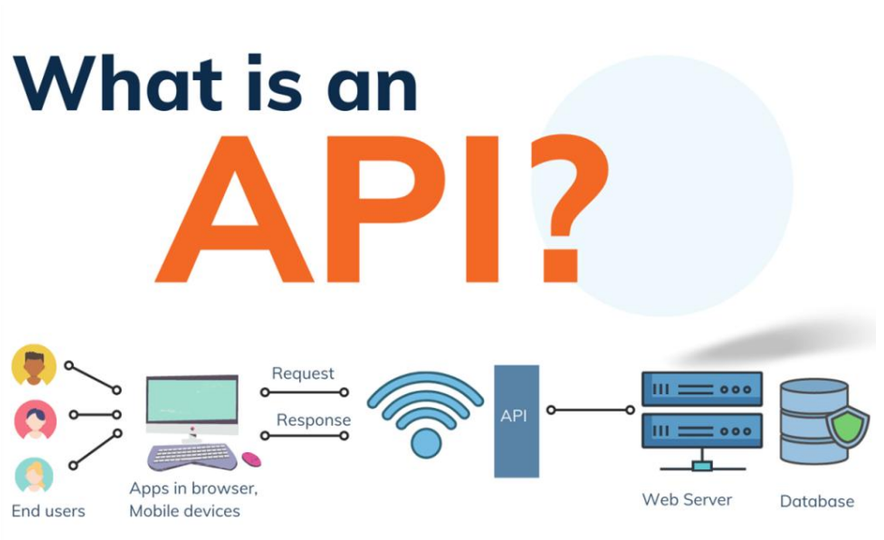
```python
# Extraer datos con pandas
import pandas as pd

url = 'https://ejemplo.com'
tables = pd.read_html(url)  # Extrae todas las tablas de la página
df = tables[0]  # Selecciona la primera tabla extraída
```

# Consideraciones Importantes

- *Rate Limiting*: no sobrecargar los servidores, usar pausas entre *requests*.
- Manejo de Errores: usar manejo de excepciones para evitar que el script falle.
- Tiempo de Ejecución por Request: usar `time.sleep()` para retrasar solicitudes y evitar bloqueos.
- Rotación de IPs y *User Agents*: técnica para cambiar la IP y el agente de usuario en cada solicitud y evitar bloqueos por parte del sitio.

La descarga de información a través de API ofrece una alternativa más práctica y ordenada, pero no siempre está disponible o es pagada

- Definición: conjunto de reglas que permite que dos sistemas se comuniquen entre sí de forma estructurada.
- Objetivo: facilitar el acceso a datos o funcionalidades de un sistema sin necesidad de interactuar directamente con su interfaz interna.
- Aplicaciones: integración de servicios, automatización de procesos, acceso a datos en tiempo real (como clima, mapas, redes sociales), entre otros.



# Ventajas de usar APIs

- Facilitan la integración entre aplicaciones.
- Permiten acceso a datos actualizados en tiempo real.
- Promueven la reutilización de servicios.
- Mejoran la escalabilidad de sistemas.

## Componentes básicos de una request a una API

- URL: Dirección del endpoint al que se hace la solicitud (ej: https://api.openweathermap.org/data/2.5/weather).
- API Key: Clave que identifica al usuario y otorga acceso autorizado al servicio.
- Headers: Información adicional que acompaña la solicitud (ej: tipo de contenido, autorización).
- Parámetros: Datos enviados en la URL o el cuerpo para especificar la consulta (ej: ciudad, unidades, idioma).
- Respuesta (JSON): La mayoría de las APIs modernas devuelven datos en formato JSON, fácil de leer y procesar.

```
1 API_KEY = os.getenv("API_KEY")
2 ciudad = "Santiago"
3 url = f"http://api.openweathermap.org/data/2.5/weather?q={ciudad}&appid={API_KEY}"
4
5 respuesta = requests.get(url)
6 datos = respuesta.json()
```

# ¿Cómo se construye la URL de una request a una API?

- Dominio base: la dirección del servidor de la API
    - Ej: https://api.openweathermap.org
- Endpoint: ruta específica para el recurso deseado
    - Ej: /data/2.5/weather
- Parámetros de consulta: valores clave para personalizar la respuesta
    - Ej: ?q=Santiago&units=metric&appid=TU_API_KEY

```
1 API_KEY = os.getenv("API_KEY")
2 ciudad = "Santiago"
3 url = f"http://api.openweathermap.org/data/2.5/weather?q={ciudad}&appid={API_KEY}"
4
5 respuesta = requests.get(url)
6 datos = respuesta.json()
```

# Diferencias entre Web Scraping y API

| | Web Scraping | API |
|---|---|---|
| **Fuente de datos** | HTML de páginas web | Respuesta estructurada (JSON/XML) |
| **Robustez** | Frágil ante cambios en el sitio | Más estable y documentada |
| **Velocidad** | Típicamente lenta | Muy rápida |
| **Eficiencia de uso** | Requiere procesar el HTML | Acceso directo a la información |

Pasemos ahora a la visualización con la siguiente pregunta: ¿qué es una página web?

- Es un documento accesible por internet que puede mostrar texto, imágenes, gráficos, formularios y más.
- Se accede a través de un navegador (como Chrome o Firefox) y está escrita comúnmente en HTML, CSS y JavaScript.
- Tipos:
  - Estática: siempre muestra lo mismo.
  - Dinámica: cambia su contenido según el usuario o datos (ej: clima en tu ciudad).

# ¿Qué es un dashboard?

- Es una página web interactiva que permite visualizar datos de forma clara y útil.
- Muestra gráficos, métricas, filtros y controles para explorar información.
- Se usa en áreas como:
  - Finanzas
  - Transporte
  - Clima
  - Salud
  - Educación

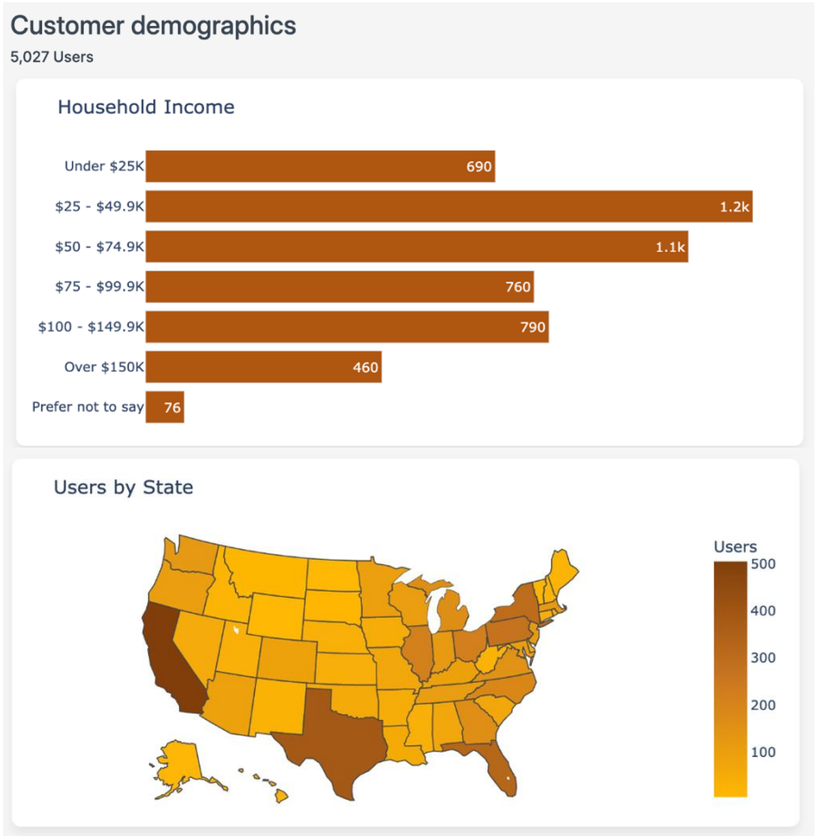
# ¿Qué es Dash?

- Dash es un framework de Python para construir aplicaciones web interactivas sin necesidad de escribir código en HTML, CSS o JavaScript.
- Ideal para crear dashboards de visualización de datos y herramientas analíticas accesibles desde el navegador.
- Desarrollado por Plotly, combina lo mejor de Flask (web backend), React (interactividad) y Plotly (gráficos).


https://plotly.com/dash/

# ¿Qué se puede hacer con Dash?

- Visualizar datos en tiempo real desde APIs o bases de datos.
- Crear gráficos interactivos (líneas, barras, mapas, etc.).
- Filtrar información con sliders, dropdowns, checkboxes y más.
- Navegar entre diferentes páginas o secciones (tabs, URLs).
- Generar interfaces personalizadas sin salir de Python.



# Componentes básicos de una app en Dash

- Layout:
  - Estructura visual de la app (títulos, textos, gráficos, menús, etc.). Se define con componentes de dash.html y dash.dcc.
- Callbacks:
  - Funciones que conectan la interacción del usuario con actualizaciones en la interfaz. Permiten hacer la app reactiva.
- Interactividad:
  - Dash permite usar elementos como dropdowns, sliders, checkboxes, tabs, y más, para que el usuario explore los datos a su ritmo.

# ¿Cómo funciona un callback en Dash?

- Un callback es una función en Python que se activa cuando el usuario interactúa con la app (por ejemplo, al seleccionar una ciudad o mover un slider).
- Cada callback tiene:
  - Input(s): lo que el usuario cambia (ej: valor de un dropdown).
  - Output(s): lo que se actualiza en la app (ej: contenido de un gráfico).
  - (Opcional) State: valores actuales que no activan el callback, pero se pueden usar en el cálculo.
- Flujo de un callback:
  - Usuario interactúa con un componente →Dash detecta el cambio →Ejecuta la función del callback →Actualiza la parte correspondiente del layout

# Para cerrar, ¿cómo son los ejercicios en este capítulo?

- El flujo comienza típicamente con la identificación de o los sitios web o APIs donde se encuentra la información requerida.
- Si se usa web scraping, se deberá explorar los sitios web y luego construir un algoritmo que extraiga la información desde el HTML descargado.
- Si se usa API, se deberá conseguir el acceso a esta (ojo con el manejo de la API key) y configurar adecuadamente los llamados.
- Finalmente, se deberá desarrollar la visualización con Dash, apuntando a la efectividad en la comunicación de la información, combinado con una representación atractiva.



plotly | Dash

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación


# IIC2115 - Programación como Herramienta para la Ingeniería

## Extracción y visualización de datos

**Profesor:** Hans Löbel