# Segunda práctica de laboratorio de Sistemas de Información para la Web

## Motivación

Un *crawler* es un programa que explora la Web de forma automática y recursiva, habitualmente con el propósito de alimentar el índice de un buscador web pero también para la creación de un *mirror* (una réplica de un sitio web dado). Con una política de visitas adecuadas el uso de *crawlers* permite que las colecciones de documentos de los buscadores web estén actualizadas. Los *crawlers* tienen un comportamiento muy diferente al de un usuario explorando un sitio web por lo que tienden a consumir más recursos del sitio que están "atacando". En consecuencia, los distintos sitios web suelen hacer uso de un fichero `robots.txt` que indica qué *crawlers* pueden visitar qué páginas y bajo qué condiciones.

## Descripción del ejercicio y del entregable

El objetivo del ejercicio es crear un script en Python que implemente un *crawler* básico para la descarga de archivos HTML. Dicho script deberá permitir al menos la configuración mediante el paso de 3 parámetros:

1. El nombre del fichero que contiene las URLs semilla.
2. El máximo de archivos a descargar.
3. El tiempo (en segundos) que debe esperar entre dos peticiones `GET`.

El archivo de URLs será un archivo de texto que tendrá una URL http o https por línea y que contendrá al menos una URL. Así, un archivo válido podría ser:

```
http://www.ingenieriainformatica.uniovi.es
```

Pero también lo sería:

```
https://en.wikipedia.org/wiki/Dromaeosauroides
https://en.wikipedia.org/wiki/Theropoda
https://en.wikipedia.org/wiki/Dinosaur
https://en.wikipedia.org/wiki/Early_Cretaceous
```

El script sólo debe procesar y almacenar archivos HTML por lo que debería determinarse el tipo de contenido examinando las cabeceras de respuesta HTTP, en concreto `content-type`.

De manera recursiva el script procesará el contenido de cada archivo HTML para buscar nuevos enlaces. A tal fin se recomienda utilizar la biblioteca Beautiful Soup que permite extraer con facilidad todos los elementos de un tipo (p.ej. El tipo `a`, enlace) de un documento HTML.

El script no realizará descargas en paralelo y deberá esperar obligatoriamente el número de segundos especificado por línea de órdenes entre una y otra petición.

Una vez se hayan descargado el máximo de documentos indicado por parámetro el script finalizará.

**¡Atención!** La exploración de las semillas puede hacerse primero en anchura o primero en profundidad. Puede implementarse cualquiera de ellas o ambas y que sea una opción configurable. En caso de implementar ambas se considerará como un *bonus* en la evaluación.

De manera opcional el script puede localizar el archivo `robots.txt` para los sitios que explore y actuar según sus indicaciones. Dicha funcionalidad también se considerará como un *bonus*. Para ello se puede utilizar la biblioteca `robotparser`.

En el campus virtual se entregará un archivo comprimido que contendrá el script así como algunos archivos de semillas indicando, además, para cada uno de ellos qué listado de archivos debería descargar el programa con una configuración dada. **No es necesario incluir los archivos descargados.**

# Pseudocódigo

```
max_downloads = 10

def crawl(url, seconds):
    if max_downloads == 0:
        return
    html = download(url)                        # Using requests
    max_downloads -= 1
    sleep(seconds)                              # Using time
    links = parse_and_find_all_links(html)      # Using BeautifulSoup
    for l in links:
        l = normalize_link(url, l)
        crawl(l, seconds)

def normalize_link(url, link):
    if link.startswith("/") or link.startswith("#"):
        return join(url, link)                  # Using urlparse
    return link

crawl("http://lne.es", 10)
```

# Referencias bibliográficas


- urllib2 — extensible library for opening URLs
- PycURL – A Python Interface To The cURL library
- Requests - Requests: HTTP for Humans™
- Beautiful Soup 4.4.0 documentation
- robotparser — Parser for robots.txt