

Practica 5

IMPLEMENTACIÓN DE LA APLICACIÓN WGET



MATERIA

Aplicaciones para comunicaciones de red

PROFESOR

Moreno Cervantes Axel Ernesto

GRUPO

6CM2

ALUMNOS

- Guevara Badillo Areli Alejandra
- Ramírez Martínez Alejandro

FECHA

martes, 10 de junio de 2025

INTRODUCCIÓN

En el contexto de las comunicaciones web, la descarga eficiente de recursos desde servidores HTTP es esencial para múltiples aplicaciones modernas, como respaldos de sitios web, análisis de contenido o automatización de procesos. Para optimizar este tipo de tareas, se recurre al uso de hilos concurrentes que permiten procesar múltiples solicitudes en paralelo. No obstante, la creación excesiva de hilos puede afectar el rendimiento del sistema, por lo que se recurre a técnicas como el uso de albercas de hilos (thread pools) para controlar su número y reutilizarlos eficientemente.

Esta práctica se enfoca en la implementación de una aplicación en consola que simula el comportamiento de WGET, permitiendo la descarga recursiva de recursos web mediante el protocolo HTTP y utilizando albercas de hilos para administrar múltiples descargas concurrentes. A través de este desarrollo, se comprenderán mejor los principios de la programación concurrente y el manejo de recursos compartidos en entornos de red.

OBJETIVOS

- Implementar una aplicación en consola capaz de descargar recursos desde servidores HTTP utilizando el método GET.
- Utilizar una alberca de hilos (thread pool) para gestionar múltiples descargas de forma concurrente.
- Permitir la descarga recursiva de recursos asociados a hipervínculos dentro de páginas HTML.
- Controlar el nivel de anidamiento en la descarga para evitar ciclos infinitos o sobrecarga innecesaria.
- Gestionar de forma eficiente una cola compartida de URLs pendientes y un registro de URLs ya procesadas.
- Evitar la descarga duplicada o simultánea de los mismos recursos durante la ejecución del programa.
- Analizar el impacto del número de hilos sobre el rendimiento general de la aplicación.

DESARROLLO

En esta práctica se deberá desarrollar una aplicación en modo consola capaz de realizar la descarga o clonación de sitios web mediante el protocolo HTTP, utilizando peticiones del tipo GET. La aplicación permitirá descargar uno o varios recursos especificados por el usuario y, adicionalmente, ofrecerá la opción de realizar una descarga recursiva, en la cual se obtendrán también los recursos asociados a los hipervínculos presentes en las páginas HTML descargadas. Para evitar un proceso de descarga infinito, se podrá establecer un nivel máximo de anidamiento que limite la profundidad con la que se seguirán los enlaces encontrados dentro de los documentos descargados.

- **WGET.java:** Este programa en Java implementa el núcleo de una aplicación tipo WGET para la descarga de sitios web mediante el protocolo HTTP. A través de una interfaz en consola, el usuario puede especificar una URL de inicio, el número de hilos a utilizar y la profundidad máxima de descarga recursiva. Internamente, utiliza un pool de hilos fijo para procesar múltiples descargas de manera concurrente, evitando la creación excesiva de hilos y mejorando el rendimiento general. Cada recurso a descargar se encola junto con su nivel de profundidad, y las tareas de descarga se distribuyen entre los hilos disponibles. Cada tarea se encarga de establecer la conexión HTTP, guardar el contenido en el sistema de archivos, y en caso de tratarse de una página HTML, analizar los hipervínculos para continuar la descarga recursiva, siempre respetando el límite de profundidad. El programa gestiona de forma segura una cola compartida de URLs pendientes y un conjunto sincronizado de URLs ya descargadas, evitando duplicados y conflictos. Además, implementa reintentos y manejo de errores para conexiones fallidas, y presenta un resumen final con estadísticas de la descarga.

WGET.JAVA

La función principal de este programa es descargar recursivamente una página web y todos sus recursos relacionados (como otras páginas, imágenes, scripts, etc.), guardándolos localmente, utilizando múltiples hilos, y manteniendo la estructura de enlaces y carpetas original.

Declaración de Constantes

- USER_AGENT: Identifica el crawler como un navegador Mozilla
- LINK_PATTERN/SRC_PATTERN: Expresiones regulares para encontrar enlaces y recursos en HTML

```
public class WGET {  
    private static final String USER_AGENT = "Mozilla/5.0";  
    private static final Pattern LINK_PATTERN = Pattern.compile("href=\"(.*?)\"", Pattern.CASE_INSENSITIVE);  
    private static final Pattern SRC_PATTERN = Pattern.compile("src=\"(.*?)\"", Pattern.CASE_INSENSITIVE);
```

Estructuras de Datos Compartidas

Contienen:

- El pool de hilos para ejecutar tareas concurrentes.
- Un conjunto sincronizado para URLs ya descargadas.
- Una cola concurrente de URLs pendientes de procesar.
- Profundidad máxima permitida.
- URL base para restringir descargas a un mismo dominio.
- Carpeta donde se guardarán los archivos.
- Contadores atómicos de éxito y error para estadísticas.

```
private static ExecutorService threadPool;  
private static Queue<String> urlQueue = new ConcurrentLinkedQueue<>();  
private static Set<String> downloadedUrls = ConcurrentHashMap.newKeySet();  
private static Set<String> failedUrls = ConcurrentHashMap.newKeySet();  
private static AtomicInteger successCount = new AtomicInteger(0);  
private static AtomicInteger totalAttempts = new AtomicInteger(0);
```

Método main

- Obtiene configuración del usuario (URL, hilos, profundidad)
- Prepara directorios de salida
- Inicia el thread pool
- Procesa URLs hasta que la cola esté vacía
- Muestra resumen final

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.println("=== Aplicación WGET ===");  
    System.out.print("Ingrese la URL a descargar: ");  
    baseUrl = scanner.nextLine().trim();  
  
    try {  
        URL urlObj = new URL(baseUrl);  
        siteFolderName = urlObj.getHost();  
        if (siteFolderName.startsWith("www.")) {  
            siteFolderName = siteFolderName.substring(4);  
        }  
    } catch (MalformedURLException e) {  
        System.err.println("URL inválida: " + e.getMessage());  
        return;  
    }  
}
```

```

System.out.print("Ingrese el número de hilos: ");
int threadCount = scanner.nextInt();

System.out.print("Ingrese la profundidad máxima: ");
maxDepth = scanner.nextInt();

System.out.println("\n=== Configuración ===");
System.out.println("URL: " + baseUrl);
System.out.println("Número de hilos: " + threadCount);
System.out.println("Profundidad máxima: " + maxDepth);
System.out.println("Directorio de salida: " + outputDir + "/" + siteFolderName);
System.out.println("=====\n");

try {
    Files.createDirectories(Paths.get(outputDir, siteFolderName));

} catch (IOException e) {
    System.err.println("Error al crear el directorio de salida: " + e.getMessage());
    return;
}

ThreadPoolExecutor threadPool = Executors.newFixedThreadPool(threadCount);

urlQueue.add(baseUrl + "|0");

while (!urlQueue.isEmpty() || getActiveCount() > 0) {
    String urlWithDepth = urlQueue.poll();
    if (urlWithDepth != null) {
        threadPool.execute(() -> processUrl(urlWithDepth));
    } else {
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

threadPool.shutdown();
try {
    threadPool.awaitTermination(1, TimeUnit.MINUTES);
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
}

System.out.println("\n=== Resumen Final ===");
System.out.println("Total de archivos intentados: " + totalAttempts.get());
System.out.println("Descargas exitosas: " + successCount.get());
}

```

Procesamiento de URLs (processUrl)

- Descarga el contenido usando HttpURLConnection
- Guarda en sistema de archivos con estructura de directorios similar al sitio
- Para HTML: extrae enlaces y los añade a la cola (si no excede profundidad máxima)
- Reescribe enlaces para que apunten a rutas locales

```

private static void processUrl(String urlWithDepth) {
    String[] parts = urlWithDepth.split("\\|");
    String url = parts[0];
    int depth = Integer.parseInt(parts[1]);

    if (downloadedUrls.contains(url)) {
        return;
    }

    downloadedUrls.add(url);
    totalAttempts.incrementAndGet();

    try {
        System.out.println("Descargando: " + url);

        URL urlObj = new URL(url);
        HttpURLConnection connection = (HttpURLConnection) urlObj.openConnection();
        connection.setRequestMethod("GET");
        connection.setRequestProperty("User-Agent", USER_AGENT);

        int responseCode = connection.getResponseCode();
    }
}

```

```

        if (responseCode == HttpURLConnection.HTTP_OK) {
            String filePath = getLocalFilePath(url);
            File outputFile = new File(filePath);

            outputFile.getParentFile().mkdirs();

            try (InputStream inputStream = connection.getInputStream();
                FileOutputStream outputStream = new FileOutputStream(outputFile)) {

                byte[] buffer = new byte[4096];
                int bytesRead;

                while ((bytesRead = inputStream.read(buffer)) != -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }

                System.out.println("Guardado como: " + filePath);
                successCount.incrementAndGet();

                if (depth < maxDepth && (url.endsWith(".html") || url.endsWith(".htm") ||
                    connection.getContentType().contains("text/html"))) {

                    String htmlContent = new String(Files.readAllBytes(outputFile.toPath()));
                    processLinks(htmlContent, url, depth);
                    ensureHtmlExtension(outputFile);

                    String htmlContent = new String(Files.readAllBytes(outputFile.toPath()));
                    String modifiedContent = rewriteLinks(htmlContent, url, filePath);
                    Files.write(outputFile.toPath(), modifiedContent.getBytes());
                } else {
                    System.err.println("Error al descargar " + url + " - Código: " + responseCode);
                    failedUrls.add(url + " (Código: " + responseCode + ")");
                }
            } catch (Exception e) {
                System.err.println("Error al procesar " + url + ": " + e.getMessage());
                failedUrls.add(url + " (Error: " + e.getMessage() + ")");
            }
        }
    }
}

```

Método ensureHtmlExtension

Garantizar que los archivos que contienen HTML tengan siempre una extensión adecuada (.html o .htm), incluso cuando la URL original no incluía una extensión explícita.

```

private static void ensureHtmlExtension(File file) throws IOException {
    String path = file.getAbsolutePath();
    if (!path.endsWith(".html") && !path.endsWith(".htm")) {
        File newFile = new File(path + ".html");
        if (file.renameTo(newFile)) {
            System.out.println("Renombrado a: " + newFile.getPath());
        } else {
            System.err.println("No se pudo renombrar el archivo: " + path);
        }
    }
}

```

Manejo de Enlaces

- Busca todos los href y src en el HTML
- Filtra enlaces no relevantes (javascript:, mailto:, #)
- Convierte enlaces relativos a absolutos
- Añade URLs válidas a la cola con profundidad incrementada

```

private static void processLinks(String htmlContent, String baseUrl, int currentDepth) {
    Matcher hrefMatcher = LINK_PATTERN.matcher(htmlContent);
    while (hrefMatcher.find()) {
        String link = hrefMatcher.group(1);
        processFoundLink(link, baseUrl, currentDepth);
    }

    Matcher srcMatcher = SRC_PATTERN.matcher(htmlContent);
    while (srcMatcher.find()) {
        String link = srcMatcher.group(1);
        processFoundLink(link, baseUrl, currentDepth);
    }
}

private static void processFoundLink(String link, String baseUrl, int currentDepth) {
    if (link.startsWith("javascript:") || link.startsWith("mailto:") || link.startsWith("#")) {
        return;
    }

    try {
        URL absoluteUrl = new URL(new URL(baseUrl), link);
        String normalizedUrl = absoluteUrl.toString().split("#")[0];

        if (!downloadedUrls.contains(normalizedUrl)) {
            urlQueue.add(normalizedUrl + "|" + (currentDepth + 1));
        }
    } catch (MalformedURLException e) {
        System.err.println("Enlace inválido: " + link + " en " + baseUrl);
    }
}

```

Sistema de Archivos Local

- Crea estructura de directorios que refleja la del sitio web
- Renombra archivos sin extensión añadiendo .html
- Convierte / en index.html
- Reescribe enlaces en HTML para apuntar a rutas locales

```

private static String getLocalFilePath(String url) throws MalformedURLException {
    URL urlObj = new URL(url);
    String path = urlObj.getPath();

    path = path.split("\\?")[0].split("#")[0];

    if (path.endsWith("/")) {
        path += "index.html";
    } else {
        String[] segments = path.split("/");
        String lastSegment = segments[segments.length - 1];
        if (!lastSegment.contains(".")) {
            path += ".html";
        }
    }

    String host = urlObj.getHost();
    if (host.startsWith("www.")) {
        host = host.substring(4);
    }

    String filePath = outputDir + "/" + siteFolderName + path;
    filePath = filePath.replaceAll("/{2,}", "/");

    return filePath;
}

```

```

private static String rewriteLinks(String htmlContent, String baseUrl, String filePath) throws MalformedURLException {
    String localBasePath = new File(outputDir, siteFolderName).toURI().toString();

    Matcher hrefMatcher = LINK_PATTERN.matcher(htmlContent);
    StringBuffer sb = new StringBuffer();

    while (hrefMatcher.find()) {
        String originalLink = hrefMatcher.group(1);
        String newLink = convertToLocalLink(originalLink, baseUrl, localBasePath);
        hrefMatcher.appendReplacement(sb, "href=\"" + newLink + "\"");
    }
    hrefMatcher.appendTail(sb);

    Matcher srcMatcher = SRC_PATTERN.matcher(sb.toString());
    sb = new StringBuffer();

    while (srcMatcher.find()) {
        String originalLink = srcMatcher.group(1);
        String newLink = convertToLocalLink(originalLink, baseUrl, localBasePath);
        srcMatcher.appendReplacement(sb, "src=\"" + newLink + "\"");
    }
    srcMatcher.appendTail(sb);

    return sb.toString();
}

private static String convertToLocalLink(String originalLink, String baseUrl, String localBasePath) throws MalformedURLException {
    if (originalLink.startsWith("javascript:") || originalLink.startsWith("mailto:") || originalLink.startsWith("#")) {
        return originalLink;
    }

    URL resolvedUrl;
    if (originalLink.startsWith("http://") || originalLink.startsWith("https://")) {
        resolvedUrl = new URL(originalLink);
        if (resolvedUrl.getHost().equals(new URL(baseUrl).getHost())) {
            return handleLocalPath(resolvedUrl.getPath(), localBasePath);
        }
        return originalLink;
    }

    resolvedUrl = new URL(new URL(baseUrl), originalLink);
    return handleLocalPath(resolvedUrl.getPath(), localBasePath);
}

```

Método handleLocalPath

Es una función clave para el manejo de rutas locales en el sistema de archivos cuando se descarga un sitio web completo. Su función principal es normalizar y adaptar las rutas de los archivos descargados para que mantengan una estructura coherente en el sistema local.

```

private static String handleLocalPath(String path, String localBasePath) {
    path = path.split("\\?")[0].split("#")[0];

    if (path.endsWith("/")) {
        path += "index.html";
    }
    else if (!path.contains(".")) {
        path += "/index.html";
    }

    return localBasePath + path;
}

```

PRUEBAS Y RESULTADOS

Para verificar el correcto funcionamiento de la aplicación, se realizaron pruebas en un entorno controlado utilizando diferentes sitios web públicos y archivos HTML de prueba. Se evaluó la capacidad del programa para realizar descargas individuales y recursivas mediante peticiones GET, así como el manejo correcto de hipervínculos internos hasta el nivel de anidamiento especificado. También se probó el comportamiento del sistema ante recursos inexistentes, enlaces inválidos y restricciones de acceso, observando el manejo adecuado de errores y la generación de mensajes informativos. Estas pruebas permitieron validar la eficiencia del uso de hilos concurrentes y la integridad de los archivos descargados.

Primero se inicializa el WGET, donde lo primero que nos va a pedir es ingresar el URL de la página que nosotros deseamos copiar

```
ant -f "C:\Users\erick\OneDrive\Documentos\Semestre 2025\Redes\Practica5_WGET" -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\erick\OneDrive\Documentos\Semestre 2025\Redes\Practica5_WGET\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\erick\OneDrive\Documentos\Semestre 2025\Redes\Practica5_WGET\build\build-jar.properties
compile:
run:
=== Aplicaci#n WGET ===
Ingrese la URL a descargar: https://www.ipn.mx/
```

Una vez ingresada la URL nos solicitará el número de hilos que nosotros deseamos en nuestra albarca de hilos y el nivel de profundidad en el que deseamos copiar la página

```
ant -f "C:\Users\erick\OneDrive\Documentos\Semestre 2025\Redes\Practica5_WGET" -Dnb.internal.action.name=run run
init:
Deleting: C:\Users\erick\OneDrive\Documentos\Semestre 2025\Redes\Practica5_WGET\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\erick\OneDrive\Documentos\Semestre 2025\Redes\Practica5_WGET\build\build-jar.properties
compile:
run:
=== Aplicaci#n WGET ===
Ingrese la URL a descargar: https://www.ipn.mx/
Ingrese el n#mero de hilos: 5
Ingrese la profundidad m#xima: 1
```

Después de ingresar la configuración nos mostrara un resumen de esta

```
=== Configuraci#n ===
URL: https://www.ipn.mx/
N#mero de hilos: 5
Profundidad m#xima: 1
Directorio de salida: URLs\ipn.mx
=====
```

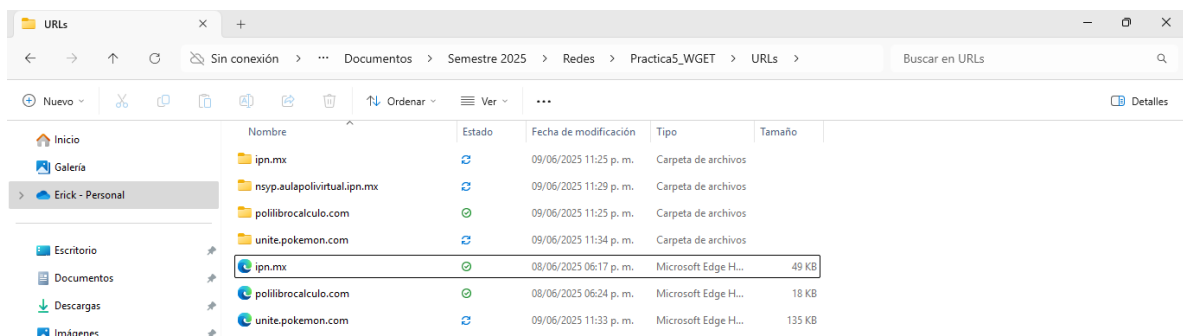
E iniciará la descarga de los archivos encontrados en el URL dado, donde nos mostrará en que directorio se va a guardar los archivos descargados, que archivos se están guardando e incluso errores por si un hipervínculo no existe (404), si hay peticiones mal formadas (400), etc.

```
Guardado como: URLs\ipn.mx\dsett/index.html
Descargando: https://www.ipn.mx/agenda-de-eventos.html
Guardado como: URLs\ipn.mx/agenda-de-eventos.html
Descargando: https://www.facebook.com/ipn.mx
Guardado como: URLs\ipn.mx/vinculacion/index.html
Descargando: https://www.ipn.mx/assets/files/main/img/redes-sociales/fb.svg
Guardado como: URLs\ipn.mx/assets/files/main/img/redes-sociales/fb.svg
Descargando: https://x.com/IPN_MX
Error al descargar https://x.com/IPN_MX - C#digo: 400
Descargando: https://www.ipn.mx/assets/files/main/img/redes-sociales/x.svg
Guardado como: URLs\ipn.mx/assets/files/main/img/redes-sociales/x.svg
Descargando: https://www.instagram.com/ipn_oficial/
Error al procesar https://www.facebook.com/ipn.mx: URLs\ipn.mx\ipn.mx (Acceso denegado)
Descargando: https://www.ipn.mx/assets/files/main/img/redes-sociales/ig.svg
Error al procesar https://www.enmh.ipn.mx/servicios/catalogo-de-servicios.html: (certificate_unknown) PKIX path building failed: sun.security
Descargando: https://www.voutube.com/user/IPNoficial
```

Una vez terminada la descarga nos mostrará un resumen de las descargas

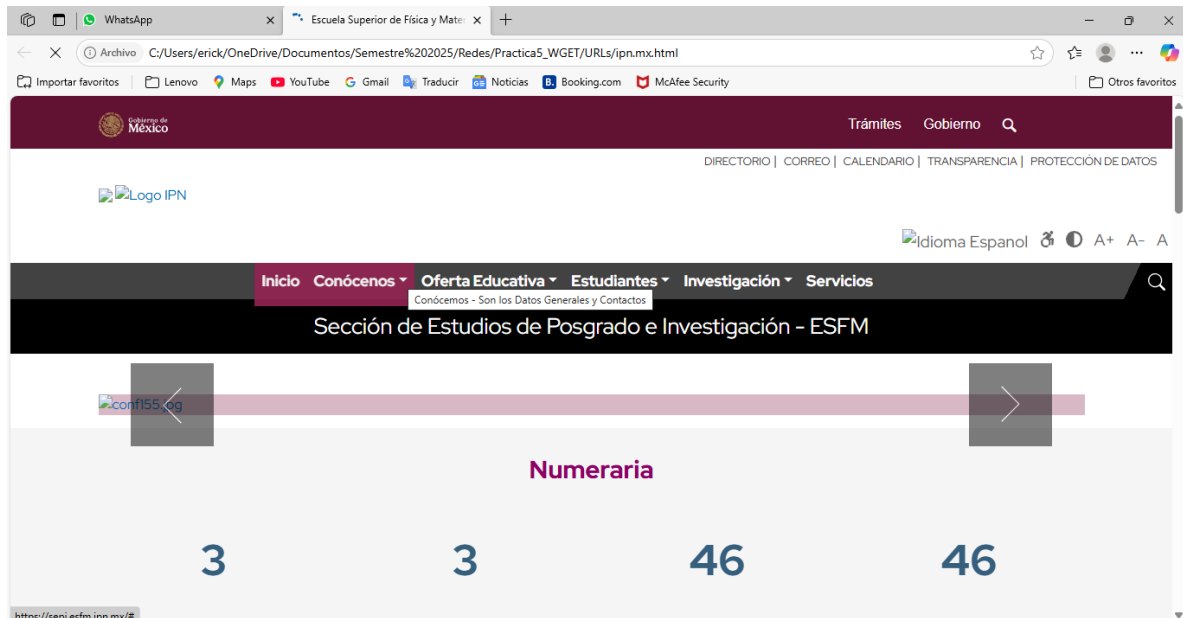
```
=== Resumen Final ===
Total de archivos intentados: 180
Descargas exitosas: 155
BUILD SUCCESSFUL (total time: 42 seconds)
```

Y podremos ver que los archivos fueron descargados si nos vamos a las carpetas

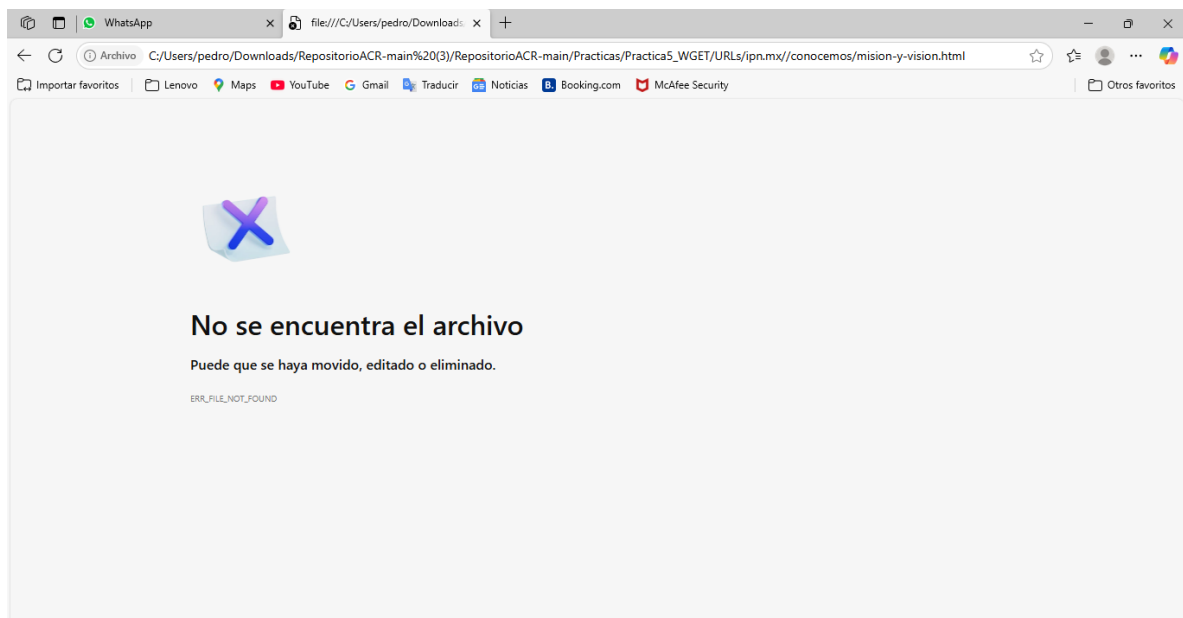


Nombre	Estado	Fecha de modificación	Tipo	Tamaño
ipn.mx	✓	09/06/2025 11:25 p. m.	Carpeta de archivos	
nssyp.aulapolivirtual.ipn.mx	✓	09/06/2025 11:29 p. m.	Carpeta de archivos	
polibrocalculo.com	✓	09/06/2025 11:25 p. m.	Carpeta de archivos	
unite.pokemon.com	✓	09/06/2025 11:34 p. m.	Carpeta de archivos	
ipn.mx	✓	08/06/2025 06:17 p. m.	Microsoft Edge H...	49 KB
polibrocalculo.com	✓	08/06/2025 06:24 p. m.	Microsoft Edge H...	18 KB
unite.pokemon.com	✓	09/06/2025 11:33 p. m.	Microsoft Edge H...	135 KB

Notamos que el link es de una ruta interna.



Si nos movemos por las rutas, éstas serán internas, en este ejemplo no se aprecia la página completamente debido al nivel de profundidad.



Los resultados de las pruebas demostraron que:

- La aplicación permite realizar descargas correctamente a partir de las URLs proporcionadas, incluyendo recursos individuales y páginas HTML completas.
- La descarga recursiva funciona de manera adecuada, respetando el nivel de anidamiento configurado por el usuario.
- El programa evita la descarga duplicada de recursos y gestiona correctamente la cola de URLs pendientes.
- Los errores 404 (recurso no encontrado) y 403 (acceso prohibido) se identifican y muestran en consola, permitiendo saber si alguna descarga no se puede completar.
- La salida en consola permite observar claramente el progreso de cada descarga, los archivos guardados, los errores detectados y el resumen final de la ejecución.

CONCLUSIONES

Desarrollar una aplicación similar a WGET fue una experiencia altamente valiosa que nos permitió aplicar de manera práctica diversos conceptos fundamentales del desarrollo en red y la programación concurrente. Uno de los aprendizajes más relevantes fue el uso eficiente de pools de hilos para gestionar múltiples descargas en paralelo, lo que destacó la importancia de controlar adecuadamente los recursos del sistema y evitar la sobrecarga provocada por la creación innecesaria de hilos.

La implementación de solicitudes HTTP tipo GET nos permitió comprender mejor el intercambio de información entre cliente y servidor, así como interpretar correctamente los encabezados y códigos de estado. Además, incorporar la funcionalidad de descarga recursiva, siguiendo enlaces internos dentro de una página, nos ayudó a entender la estructura de los sitios web y cómo automatizar su navegación y respaldo.

Durante el desarrollo, enfrentamos distintos desafíos técnicos, como prevenir la descarga repetida de las mismas URLs, gestionar adecuadamente errores de red o redireccionamientos, y organizar los archivos obtenidos de forma clara en el sistema local. Resolver estos problemas fortaleció nuestras competencias en sincronización de procesos, control de la concurrencia y análisis de contenido HTML, consolidando así nuestras habilidades en el desarrollo de herramientas prácticas para entornos reales de red.