



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



-----**APLICACIONES EN COMUNICACIONES EN RED**-----

PRÁCTICA 5:

Wget

Alumno:

Meza Vargas Brandon David

Grupo:

3CM16

Profesor:

Moreno Cervantes Axel Ernesto

Índice

Introducción.....	4
Desarrollo	5
FileDownloader	5
getFiles.....	6
getFileName	6
downloadFiles	7
isDir	7
Main.....	8
Constants	9
Pruebas de funcionamiento	9
Conclusiones.....	14
Bibliografía.....	15

Índice de ilustraciones

Ilustración 1. Primer constructor. de FileDownloader	5
Ilustración 2. Segundo constructor de FileDownloader	6
Ilustración 3. Método getfiles.	6
Ilustración 4. Método getFileName	6
Ilustración 5. Método downloadFiles.....	7
Ilustración 6. Método isDir	8
Ilustración 7. Clase Main	9
Ilustración 8. Clase Constants	9
Ilustración 9. Corriendo el programa	10
Ilustración 10. Descargando archivo.	10
Ilustración 11. Archivo descargado.	10
Ilustración 12. Archivo visualizado	11
Ilustración 13. Descargando archivos de una carpeta.	13
Ilustración 14. Descarga de carpeta.	13
Ilustración 15. Contenido de carpeta.	14

Introducción

Wget es una herramienta creada por GNU, se puede usar para recuperar contenido y archivos de varios servidores web. El nombre es una combinación World Wide Web y la palabra get. Admite descargas a través de FTP, SFTP, HTTP y HTTPS.

En la práctica actual se realizará una copia de este pequeño programa usando el lenguaje de programación Java e implementando flujos de entrada y salida para hacer la descarga de archivos, además de archivos se descargarán carpetas completas haciendo uso de la recursión y creando un hilo por cada archivo que haya para tener una descarga rápida, esto usando un pool de hilos.

Desarrollo

A continuación se muestra el desarrollo de la práctica explicando el código que lo conforma.

FileDownloader

Esta es la parte principal del programa de la práctica, en esta parte tenemos dos constructores de la clase, el primero recibe como parámetros un link, un path y la operación, esta operación indicará si se trata de una nueva descarga de un archivo o para descargar un html que tendrá el índice de los archivos contenidos en caso de que sea una carpeta.

En el primer constructor lo primero que se hace es crear una nueva URL con el link y si es la operación 1 se obtienen los archivos del link, en caso contrario se descarga el archivo.

```
public FileDownloader(String link, String path, int op){
    pool = Executors.newFixedThreadPool(Constants.THREADS);
    try {
        this.link = new URL(link);
        this.path = path;
        if(op == 1)
            getFiles();
        else
            downloadFiles(new File( pathname: path+ "/" + getFileName(link)));
    }catch (Exception e){
        e.printStackTrace();
        pool.shutdown();
    }
    pool.shutdown();
}
```

Ilustración 1. Primer constructor. de FileDownloader

En el segundo constructor tenemos como parámetros un link, path, el pool de hilos y la operación, es exactamente lo mismo que el anterior, con la diferencia que aquí mandamos el pool de hilos para hacer la descarga de una manera más rápida.

```

public FileDownloader(String link, String path, final ExecutorService pool, int op){
    this.pool = pool;
    try {
        this.link = new URL(link);
        this.path = path;
        if(op == 1)
            getFiles();
        else
            downloadFiles(new File( pathname: path+"/index.html"));
    }catch (Exception e){
        e.printStackTrace();
    }
}
}

```

Ilustración 2. Segundo constructor de FileDownloader

getFiles

Este método verifica si el archivo es una carpeta o un solo archivo, en caso de que sea una carpeta se manda a llamar al método isDir(), en caso de que no se descarga el archivo.

```

public void getFiles(){
    if(link.getFile().indexOf(".") == -1)
        isDir();
    else
        downloadFiles(new File( pathname: path + getFileName(link.getFile())));
}

```

Ilustración 3. Método getfiles.

getFileName

Este método solo retorna el nombre del archivo a descargar, esto lo hacemos cortando el nombre que viene de la url, para esto cortamos por cada / que se encuentre en la URL obteniendo así un array de strings, sabiendo que el nombre del archivo será el que viene hasta la última posición del arreglo, será esta la que regresaremos.

```

public String getFileName(String file){
    String [] x = file.split( regex: "/" );
    return x[x.length-1];
}

```

Ilustración 4. Método getFileName

downloadFiles

Este método se encarga de descargar un archivo especificado por el usuario o los archivos que viene dentro de la carpeta, lo que se hace es crear un flujo de entrada y de salida del archivo que el usuario descargará, posteriormente solo iremos leyendo cada byte del archivo recibido por el flujo de entrada y lo iremos escribiendo hasta que no haya más bytes por leer.

```
public void downloadFiles(File file){
    try {
        System.out.println(String.format(Constants.DOWNLOAD_FILE_MESSAGE, file.getName()));
        DataOutputStream dos = new DataOutputStream(new FileOutputStream(file));
        DataInputStream dis = new DataInputStream(link.openStream());

        byte[] b = new byte[Constants.BUFFER_SIZE];
        int l = dis.read(b);
        /** Downloading data*/
        while(l != -1){
            dos.write(b, off: 0, l);
            l = dis.read(b);
        }
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

Ilustración 5. Método downloadFiles

isDir

Este método es muy importante en el programa, primeramente creamos un archivo a partir de la ruta y el nombre del archivo obtenido de la url ingresada por el usuario, posteriormente creamos una carpeta ya que este método nos indica que el archivo será una carpeta.

Posteriormente crearemos un flujo de entrada con el método openStream de la URL, este método crea una conexión TCP al servidor que la URL resuelve, de esta manera se manda una petición GET al servidor, que en este caso es nuestro navegador y devuelve el archivo, posteriormente leemos los bytes de ese flujo hasta que ya no haya datos, posteriormente vamos recorriendo cada archivo y ejecutando un hilo gracias a nuestro pool para que se encargue de ese archivo, se mandará entonces el link del archivo actual, el path, el pool de hilos y la operación 0, esta indicará que se descargará el archivo.

```

public void isDir(){
    File file = new File( pathname: path+getFileName(Link.getFile()));

    if(file.mkdir()){
        path = file.getAbsolutePath()+"\\";
        try {
            DataInputStream dis = new DataInputStream(link.openStream());
            byte [] b = new byte[Constants.BUFFER_SIZE];
            int l = dis.read(b);
            String s = "";
            /** Downloading data*/
            while(l != -1){
                s += new String(b, offset: 0, l);
                l = dis.read(b);
            }
            int i = s.indexOf("PARENTDIR");
            if(i == -1){
                System.out.println(String.format(Constants.FILE_ERROR, link.getFile()));
                pool.execute(new FileDownloader(link.toString(), path, pool, op: 0));
                return;
            }
            i = s.indexOf( str: "href", i);
            pool.execute(new FileDownloader(link.toString(), path, pool, op: 0));
            while((i = s.indexOf( str: "href", fromIndex: i+1)) != -1){
                String nextFile = s.substring(i+6, s.indexOf( str: "\"", fromIndex: i+6));
                pool.execute(new FileDownloader( link: link.toString()+nextFile, path, pool, op: 1));
            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
}

```

Ilustración 6. Método isDir

Main

En esta clase solo se crea el scanner para que el usuario ingrese texto por el teclado y se manda a llamar a la clase FileDownloader para comenzar con el proceso de descarga de archivos.


```

public class Main {

    public static void main(String[] args) {
        String resp = "si";
        while(resp.equalsIgnoreCase( anotherString: "si")){
            Scanner scanner = new Scanner(System.in);
            System.out.println(Constants.ENTER_LINK_MESSAGE);
            String link = scanner.nextLine();

            FileDownloader wGet = new FileDownloader(link, path: "", op: 1);
            System.err.println(Constants.DOWNLOAD_ANOTHER_MESSAGE);
            resp = scanner.nextLine();
        }
    }
}

```

Ilustración 7. Clase Main

Constants

En esta clase solo se encuentran las constantes usadas a lo largo del programa principal.

```

public class Constants {

    public static final int BUFFER_SIZE = 65535;
    public static final String DOWNLOAD_MESSAGE = "Se ha descargado el %0.4f del archivo";
    public static final String DOWNLOAD_FILE_MESSAGE = "Descargando el archivo %s";
    public static final String ENTER_LINK_MESSAGE = "Ingrese el link del que desea descargar su contenido";
    public static final String FILE_ERROR = "Ha ocurrido un error, el archivo %s no se pudo reconocer";
    public static final String DOWNLOAD_ANOTHER_MESSAGE = "\n¿Desea ingresar otro link? Si/No";
    public static final int THREADS = 10;
}

```

Ilustración 8. Clase Constants

Pruebas de funcionamiento

A continuación se mostrará el funcionamiento de la práctica a través de capturas del programa funcionando.

Al correr el programa, esta pregunta por un link al usuario para descargar archivos de el.

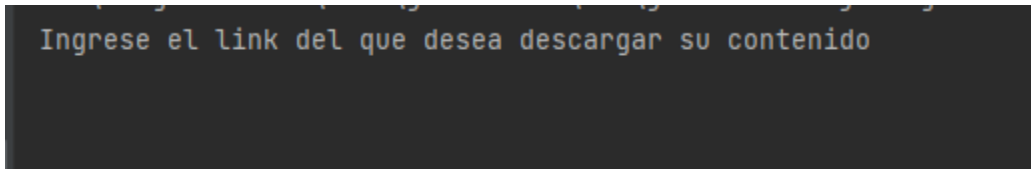


Ilustración 9. Corriendo el programa

Una vez que el usuario ingrese un link con un archivo, el programa lo descargará y preguntará al usuario si desea ingresar otro link como se ve a continuación.

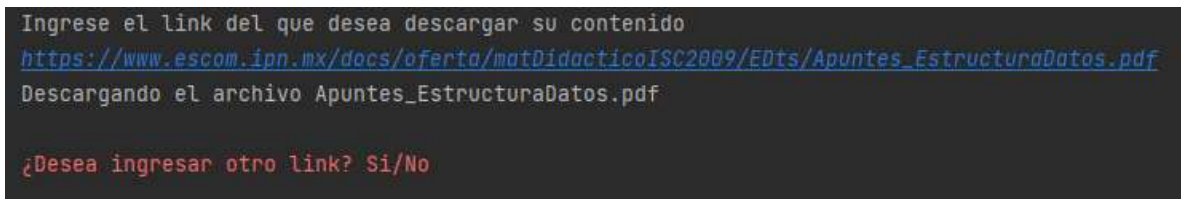


Ilustración 10. Descargando archivo.

El archivo lo podremos ver en la raíz del proyecto como se ve en la siguiente captura.

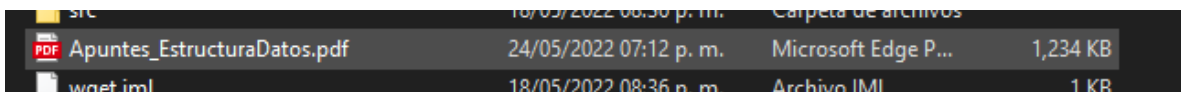


Ilustración 11. Archivo descargado.

Si lo abrimos, vemos que se puede visualizar sin ningún problema.

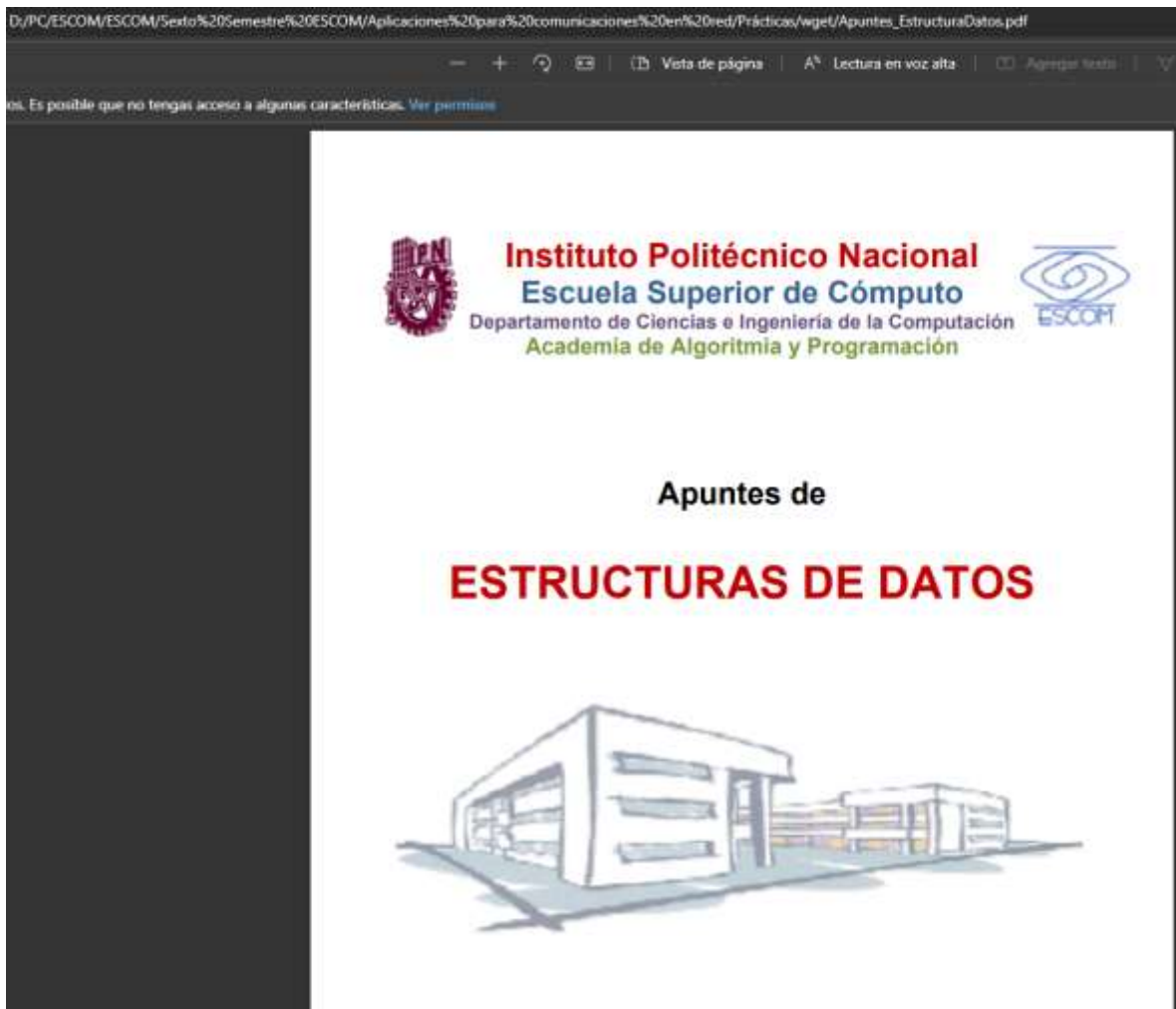


Ilustración 12. Archivo visualizado

Ahora probaremos con un archivo que s una carpeta que contiene archivos dentro de ella así como más carpetas.

```
¿Desea ingresar otro link? Si/No
si
Ingrese el link del que desea descargar su contenido
http://148.204.58.221/axel/aplicaciones/sockets/java/canales/
Descargando el archivo index.html
Descargando el archivo EchoClient.java
Descargando el archivo EchoClient2.java
Descargando el archivo EchoServer.java
Descargando el archivo EcoC.java
Descargando el archivo EcoS.java
Descargando el archivo MultiC.java
Descargando el archivo MultiS.java
Descargando el archivo MulticastChannel.docx
Descargando el archivo MulticastClient.java
Descargando el archivo MulticastServer.java
Descargando el archivo Receiver.class
Descargando el archivo Receiver.java
Descargando el archivo Sender.class
Descargando el archivo Sender.java
Descargando el archivo index.html
Descargando el archivo CNB.java
Descargando el archivo Receiver_0.java
Descargando el archivo SNB.java
Descargando el archivo Sender_0.java
Descargando el archivo SimpleDaytimeServer.java
Descargando el archivo TestEchoClient.java
Descargando el archivo TestEchoServer.java
Descargando el archivo TimeClient.java
Descargando el archivo TimeServer.java
Descargando el archivo UCNB.java
Descargando el archivo UCNBC.java
Descargando el archivo UDPC.java
```

```
Descargando el archivo UDPEchoClient.class
Descargando el archivo UDPEchoClient.java
Descargando el archivo UDPEchoServer.class
Descargando el archivo UDPEchoServer.java
Descargando el archivo UDPS.java
Descargando el archivo USNB.java
Descargando el archivo USNBC.java
Descargando el archivo canales.zip
Descargando el archivo duke1.png
Descargando el archivo index.html
Descargando el archivo ChangeRequest.class
Descargando el archivo ChangeRequest.java
Descargando el archivo EchoClient.java
Descargando el archivo EchoServer.java
Descargando el archivo EchoWorker.class
Descargando el archivo EchoWorker.java
Descargando el archivo NioClient.class
Descargando el archivo NioClient.java
Descargando el archivo NioServer.class
Descargando el archivo NioServer.java
Descargando el archivo Rox%20Java%20NIO%20Tutorial.mht
Descargando el archivo RspHandler.class
Descargando el archivo RspHandler.java
Descargando el archivo ServerDataEvent.class
Descargando el archivo ServerDataEvent.java
Descargando el archivo UDPEchoClient.java
Descargando el archivo UDPEchoServer.java
Descargando el archivo index.html
Descargando el archivo ClienteNB.java
Descargando el archivo ServidorNB.java
Descargando el archivo input.txt

¿Desea ingresar otro link? Si/No
```

Ilustración 13. Descargando archivos de una carpeta.

Podemos ver que de igual forma la descarga se encuentra en la raíz del proyecto.

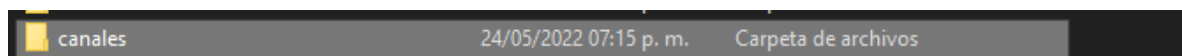
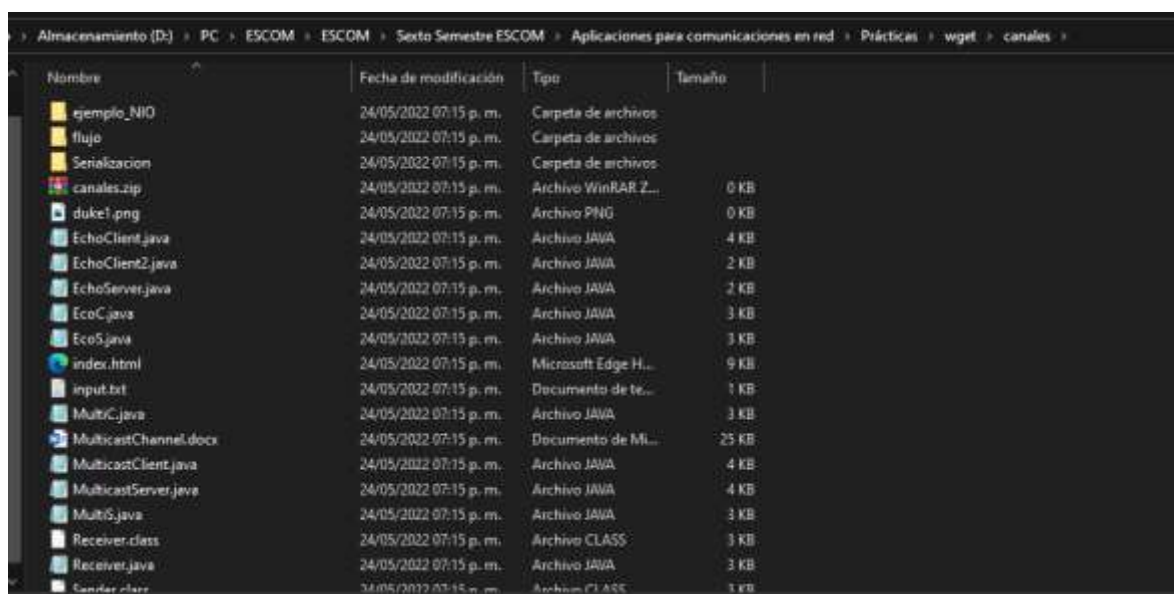


Ilustración 14. Descarga de carpeta.

Si abrimos la carpeta observamos que todos los archivos se encuentran ahí.



Nombre	Fecha de modificación	Tipo	Tamaño
ejemplo_NIO	24/05/2022 07:15 p. m.	Carpeta de archivos	
flujo	24/05/2022 07:15 p. m.	Carpeta de archivos	
Serializacion	24/05/2022 07:15 p. m.	Carpeta de archivos	
canales.zip	24/05/2022 07:15 p. m.	Archivo WinRAR Z...	0 KB
duke1.png	24/05/2022 07:15 p. m.	Archivo PNG	0 KB
EchoClient.java	24/05/2022 07:15 p. m.	Archivo JAVA	4 KB
EchoClient2.java	24/05/2022 07:15 p. m.	Archivo JAVA	2 KB
EchoServer.java	24/05/2022 07:15 p. m.	Archivo JAVA	2 KB
EcoC.java	24/05/2022 07:15 p. m.	Archivo JAVA	3 KB
EcoS.java	24/05/2022 07:15 p. m.	Archivo JAVA	3 KB
index.html	24/05/2022 07:15 p. m.	Microsoft Edge H...	9 KB
input.txt	24/05/2022 07:15 p. m.	Documento de te...	1 KB
MultiC.java	24/05/2022 07:15 p. m.	Archivo JAVA	3 KB
MulticastChannel.docx	24/05/2022 07:15 p. m.	Documento de ML...	25 KB
MulticastClient.java	24/05/2022 07:15 p. m.	Archivo JAVA	4 KB
MulticastServer.java	24/05/2022 07:15 p. m.	Archivo JAVA	4 KB
MultiS.java	24/05/2022 07:15 p. m.	Archivo JAVA	3 KB
Receiver.class	24/05/2022 07:15 p. m.	Archivo CLASS	3 KB
Receiver.java	24/05/2022 07:15 p. m.	Archivo JAVA	3 KB
Sender.class	24/05/2022 07:15 p. m.	Archivo CLASS	3 KB

Ilustración 15. Contenido de carpeta.

Conclusiones

Esta práctica estuvo divertida e interesante, pues gracias a lo que hemos aprendido hasta el momento del curso podemos hacer una copia del programa wget de una manera rápida y sencilla usando lenguaje Java en mi caso y flujos de entrada y salida asociándolos a archivos creados a partir de una URL que el usuario ingresa.

La parte complicada vino cuando se tenía que crear un hilo por cada archivo en caso de que hubiera varios en una carpeta para acelerar el proceso de descarga, sin embargo, con recursión a la misma clase y ejecutando el pool de hilos por cada archivo se solucionó esta parte y se logro que la descarga sea muy rápida pues un hilo diferente se encarga de un archivo diferente.

Una buen práctica para poner a prueba ciertos conocimientos e investigar por nuestra parte ciertos aspectos.

Bibliografía

- [1] G. Briones. (09 de marzo, 2022). Qué es y cómo usar el comando wget. [Online]. Available: [https://www.hostinger.mx/tutoriales/usar-comando-wget/#%C2%BFQue es el Comando Wget](https://www.hostinger.mx/tutoriales/usar-comando-wget/#%C2%BFQue%20es%20el%20comando%20Wget)