



**Instituto Politécnico Nacional  
Escuela Superior de Cómputo**



## **Tarea 2: Transferencia de archivos utilizando datagramas confiables**

**Olay Silis Jose Eduardo  
4CV12**

**Prof. Carlos Pineda Guerrero**

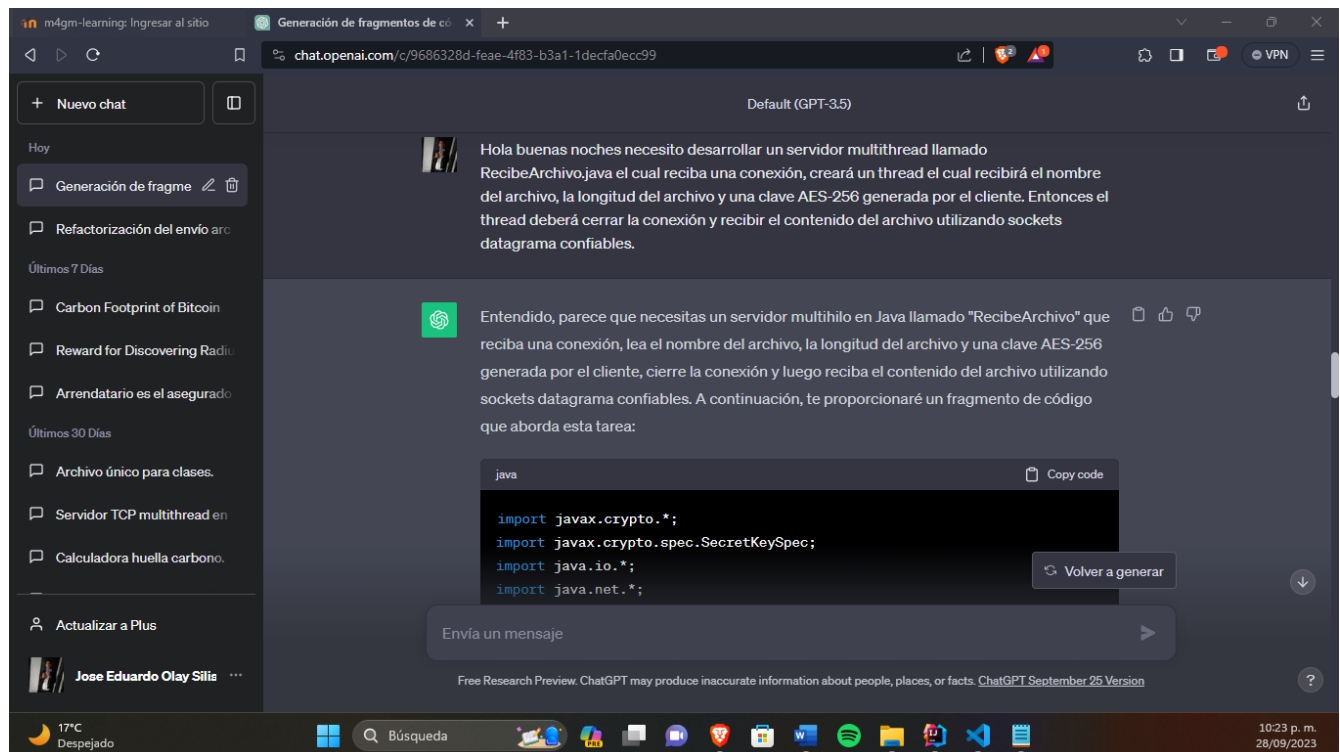
# Introducción

En la presente práctica se implementarán un par de programas diseñados para la transferencia segura y confiable de archivos a través de una red. Estos programas utilizan cifrado AES y verificación CRC32 para garantizar la privacidad y la integridad de los datos transferidos. "EnviaArchivo" cifra y envía archivos desde el cliente, mientras que "RecibeArchivo" recibe, descifra y almacena los archivos en el servidor, asegurando una transmisión segura y fiable de datos.

## Desarrollo

### Fase 1

En primera instancia se realizó un servidor multithread el cual lo llamamos **RecibeArchivo.java** dicho programa se realizó con ayuda de la IA ChatGPT de la siguiente manera.



m4gm-learning: Ingresar al sitio x Generación de fragmentos de código x +

chat.openai.com/c/9686328d-feae-4f83-b3a1-1decfa0ecc99

+ Nuevo chat

Hoy

- Generación de fragmentos de código
- Refactorización del envío de archivos

Últimos 7 Días

- Carbon Footprint of Bitcoin
- Reward for Discovering Radio
- Arrendatario es el asegurado

Últimos 30 Días

- Archivo único para clases.
- Servidor TCP multithread en
- Calculadora huella carbono.

Actualizar a Plus

Jose Eduardo Olay Siliis

```
import java.nio.ByteBuffer;
import java.util.*;
import java.util.zip.CRC32;
import java.util.zip.Checksum;

public class RecibeArchivo {

    public static void main(String[] args) {

        try {
            int puerto = 12345;
            DatagramSocket socket = new DatagramSocket(puerto);

            System.out.println("El servidor está en línea. Esperando conexión");

            while (true) {
                DatagramPacket solicitud = new DatagramPacket(new byte[1024], 1024);
                socket.receive(solicitud);

                // Crea un nuevo hilo para manejar la solicitud del cliente
                Thread clienteHandlerThread = new Thread(new ClienteHandler(socket, solicitud));
                clienteHandlerThread.start();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

17°C Despejado 10:23 p. m. 28/09/2023

m4gm-learning: Ingresar al sitio x Generación de fragmentos de código x +

chat.openai.com/c/9686328d-feae-4f83-b3a1-1decfa0ecc99

+ Nuevo chat

Hoy

- Generación de fragmentos de código
- Refactorización del envío de archivos

Últimos 7 Días

- Carbon Footprint of Bitcoin
- Reward for Discovering Radio
- Arrendatario es el asegurado

Últimos 30 Días

- Archivo único para clases.
- Servidor TCP multithread en
- Calculadora huella carbono.

Actualizar a Plus

Jose Eduardo Olay Siliis

```
e.printStackTrace();
}
}

private static class ClienteHandler implements Runnable {
    private DatagramSocket socket;
    private DatagramPacket solicitud;

    public ClienteHandler(DatagramSocket socket, DatagramPacket solicitud) {
        this.socket = socket;
        this.solicitud = solicitud;
    }

    @Override
    public void run() {
        try {
            System.out.println("Nueva conexión establecida con el cliente");

            // Recibe el nombre del archivo, la longitud y la clave del archivo
            byte[] datos = solicitud.getData();
            ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(datos);
            ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
            String nombreArchivo = objectInputStream.readUTF();
            int longitud = objectInputStream.readInt();
            byte[] clave = objectInputStream.readBytes(longitud);

            // Aquí se podría implementar la lógica para guardar el archivo
            // Ejemplo: File file = new File("recibido/" + nombreArchivo);
            // FileOutputStream fos = new FileOutputStream(file);
            // fos.write(datos);
            // fos.close();

            System.out.println("Archivo recibido: " + nombreArchivo + " Longitud: " + longitud + " Clave: " + clave);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

17°C Despejado 10:23 p. m. 28/09/2023

Nuevo chat

Hoy

Generación de fragme

Refactorización del envío arc

Últimos 7 Días

Carbon Footprint of Bitcoin

Reward for Discovering Radiu

Arrendatario es el asegurado

Últimos 30 Días

Archivo único para clases.

Servidor TCP multithread en

Calculadora huella carbono.

Actualizar a Plus

Jose Eduardo Olay Silis

chat.openai.com/c/9686328d-feae-4f83-b3a1-1decfa0ecc99

Generación de fragmentos de código

```
ObjectInputStream objectInputStream = new ObjectInputStream(

String nombreArchivo = (String) objectInputStream.readObject
long longitudArchivo = objectInputStream.readLong();
String claveEnBase64 = (String) objectInputStream.readObject

// Cerrar la conexión ya que hemos recibido los detalles del
socket.close();

System.out.println("Cliente " + solicitud.getAddress() + ":"
+ longitudArchivo);

// Continuar con la lógica para recibir el contenido del arc
// ...

} catch (Exception e) {
e.printStackTrace();
}
}
}
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

17°C Despejado

Búsqueda

10:24 p. m. 28/09/2023

Nuevo chat

Hoy

Generación de fragme

Refactorización del envío arc

Últimos 7 Días

Carbon Footprint of Bitcoin

Reward for Discovering Radiu

Arrendatario es el asegurado

Últimos 30 Días

Archivo único para clases.

Servidor TCP multithread en

Calculadora huella carbono.

Actualizar a Plus

Jose Eduardo Olay Silis

chat.openai.com/c/9686328d-feae-4f83-b3a1-1decfa0ecc99

Generación de fragmentos de código

Default (GPT-3.5)

```
// Continuar con la lógica para recibir el contenido del arc
// ...

} catch (Exception e) {
e.printStackTrace();
}
}
}
}
```

Volver a generar

Envía un mensaje

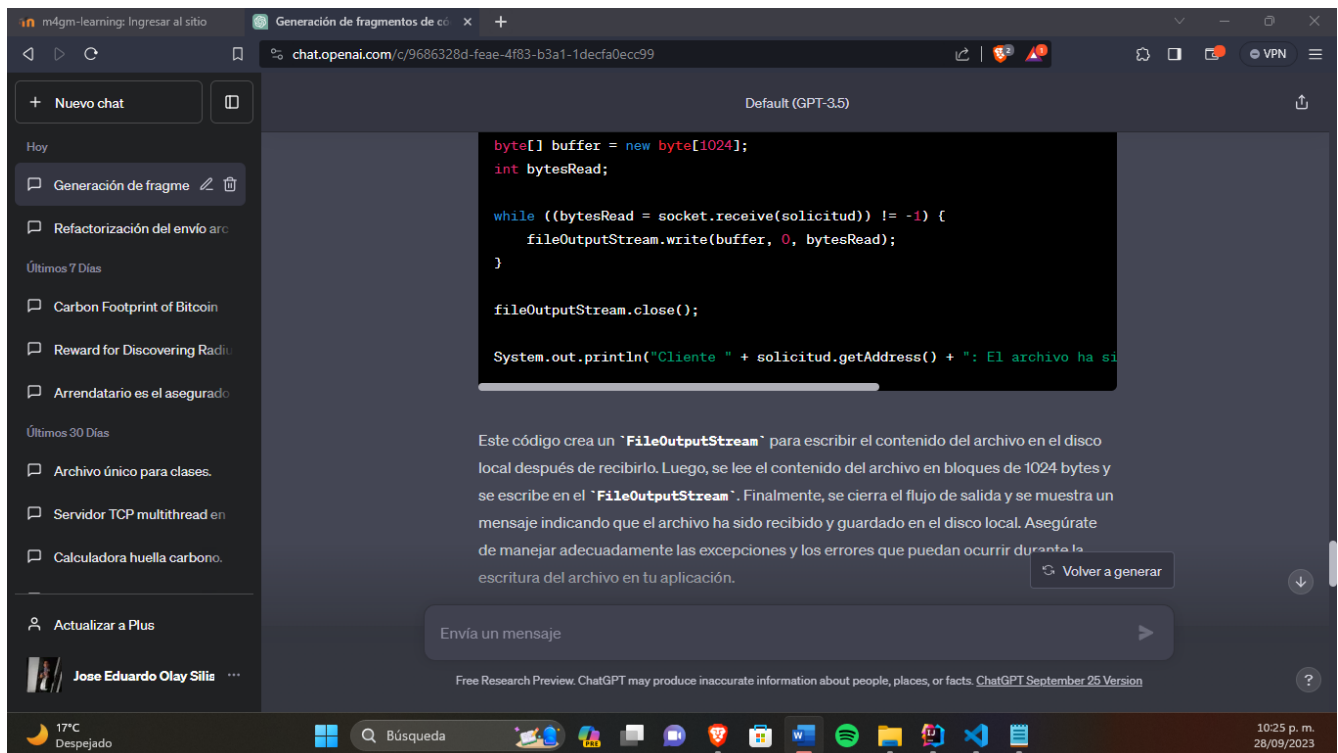
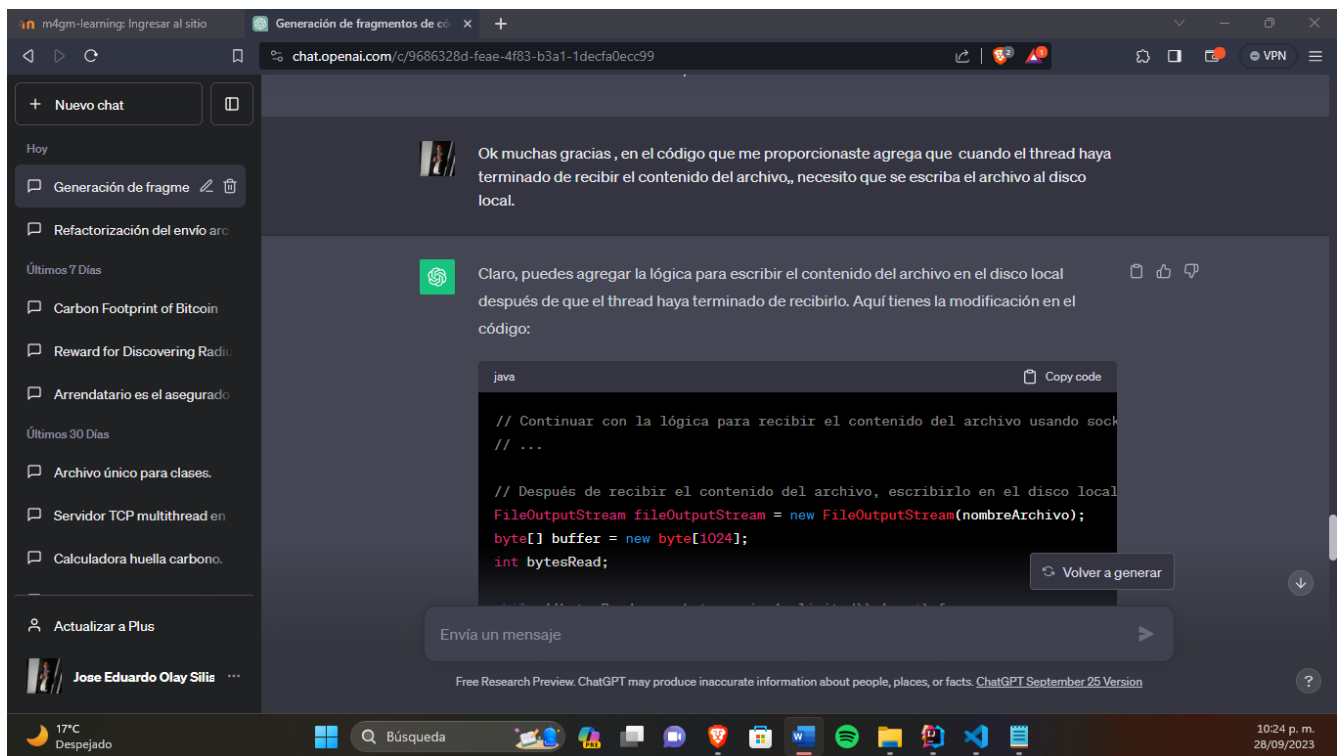
Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

17°C Despejado

Búsqueda

Este fragmento de código crea un servidor multihilo que maneja la solicitud del cliente, recibe el nombre del archivo, la longitud y la clave, y luego cierra la conexión para que puedas continuar con la lógica para recibir el contenido del archivo utilizando sockets datagrama confiables. Deberás completar la parte de recibir el contenido del archivo de acuerdo a tus necesidades específicas.

10:24 p. m. 28/09/2023



Tras la conversación realizada con el chat, el código en un principio no funcionó por lo que tuve que meterle mano por mi cuenta. A continuación, iré explicando poco a poco como es que en el desarrollo de este programa y lo que proporciono el chat pude ir implemento la integridad, llegada de paquetes, descarte de duplicados, orden y encriptado.

- **Confiabilidad (Verificación de CRC32):**

La verificación de CRC32 se realiza en el servidor después de recibir un paquete y antes de procesarlo. Si se detecta un error en la suma de comprobación, se solicita un reenvío.

```
// Dentro del método run() de la clase ClienteHandler
// ...

while (fileOutputStream.getChannel().size() < longitudArchivo) {
    socket.receive(solicitud);

    byte[] decryptedBytes = descifrador.doFinal(solicitud.getData(), 0,
solicitud.getLength());

    // Verificar la suma de comprobación CRC32
    ByteBuffer packetBuffer = ByteBuffer.wrap(decryptedBytes);
    int packetId = packetBuffer.getInt();
    int bytesRead = packetBuffer.getInt();
    byte[] content = new byte[bytesRead];
    packetBuffer.get(content);
    byte[] receivedCrcBytes = new byte[8];
    packetBuffer.get(receivedCrcBytes);
    long receivedCrcValue = ByteBuffer.wrap(receivedCrcBytes).getLong();

    Checksum crc32 = new CRC32();
    crc32.update(content);
    long calculatedCrcValue = crc32.getValue();

    if (calculatedCrcValue != receivedCrcValue) {
        System.err.println("Error en la verificación CRC32 para el paquete " +
packetId + ". Se solicita reenvío...");

        requestPacketResend(socket, solicitud.getAddress(), solicitud.getPort(),
packetId);
        continue;
    }

    // Continuar con el procesamiento del paquete
    // ...
}

// ...
```

- **Integridad (Cálculo de CRC32 en el cliente):**

En el cliente, se calcula el valor CRC32 para cada paquete antes de enviarlo al servidor.

```
// En el cliente antes de enviar un paquete
Checksum crc32 = new CRC32();
crc32.update(content);
```

```
long crcValue = crc32.getValue();

// Agregar crcValue al paquete antes de enviarlo
// ...
```

- **Descarte de duplicados:**

En el servidor, se utiliza un conjunto (receivedPacketIds) para realizar un seguimiento de los identificadores de paquetes recibidos y evitar el procesamiento de paquetes duplicados.

```
// Dentro del método run() de la clase ClienteHandler
// ...

Set<Integer> receivedPacketIds = new HashSet<>();

while (fileOutputStream.getChannel().size() < longitudArchivo) {
    socket.receive(solicitud);

    // Verificar si el paquete ya ha sido recibido
    int packetId = /* obtener el identificador del paquete */;
    if (receivedPacketIds.contains(packetId)) {
        continue; // Paquete duplicado, ignorar
    }

    // Marcar el paquete como recibido
    receivedPacketIds.add(packetId);

    // Continuar con el procesamiento del paquete
    // ...
}

// ...
```

- **Orden (Variable expectedPacketId):**

Se utiliza una variable (expectedPacketId) para rastrear el identificador de paquete esperado. Si se recibe un paquete con un identificador fuera de orden, se puede tomar una acción apropiada, como el descarte o el reenvío.

```
// Dentro del método run() de la clase ClienteHandler
// ...

int expectedPacketId = 1;

while (fileOutputStream.getChannel().size() < longitudArchivo) {
    socket.receive(solicitud);
```

```

// Verificar si el identificador de paquete coincide con el esperado
int packetId = /* obtener el identificador del paquete */;
if (packetId != expectedPacketId) {
    // Paquete fuera de orden, tomar una acción apropiada
    // ...
    continue;
}

// Continuar con el procesamiento del paquete
// ...

expectedPacketId++; // Actualizar el identificador de paquete esperado
}

// ...

```

Una vez realizado esto tocó realizar el cliente llamado **EnviaArchivo.java** el cual de igual manera se realizó con ayuda del Chat. A continuación, se muestran las instrucciones realizadas.

Generación de fragmentos de código

chat.openai.com/c/9686328d-feae-4f83-b3a1-1decfa0ecc99

Default (GPT-3.5)

Hoy

- Generación de fragmentos de código
- Refactorización del envío de archivo

Últimos 7 Días

- Carbon Footprint of Bitcoin
- Reward for Discovering Radi...
- Arrendatario es el asegurado

Últimos 30 Días

- Archivo único para clases.
- Servidor TCP multithread en
- Calculadora huella carbono.

Actualizar a Plus

Jose Eduardo Olaj Silia

Ahora necesito por favor que me ayudes a codificar un cliente el cual tendrá por nombre EnviaArchivo.java. Este recibirá como parámetro (desde la línea de comandos) el nombre de archivo a enviar, dicho archivo se leerá del disco local. Este cliente deberá generar una clave AES-256 de manera aleatoria y se conectará al servidor utilizando sockets seguros y enviará el nombre del archivo, la longitud del archivo y la clave AES-256 generada. Entonces cerrará la conexión.

Entendido, aquí tienes un código de ejemplo para un cliente llamado "EnviaArchivo.java" que cumple con tus requisitos:

```

java
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.net.*;
import java.nio.*;
import java.util.*;

```

Copy code

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 26 Version

17°C Despejado

Búsqueda

10:45 p. m. 28/09/2023



m4gm-learning: Ingresar al sitio Generación de fragmentos de código

chat.openai.com/c/9686328d-feae-4f83-b3a1-1decfa0ecc99

+ Nuevo chat

Hoy

- Generación de fragmento
- Refactorización del envío arc

Últimos 7 Días

- Carbon Footprint of Bitcoin
- Reward for Discovering Radiu
- Arrendatario es el asegurado

Últimos 30 Días

- Archivo único para clases.
- Servidor TCP multithread en
- Calculadora huella carbono.

Actualizar a Plus

Jose Eduardo Olay Silis

```
public class EnviaArchivo {  
  
    public static void main(String[] args) {  
        if (args.length != 1) {  
            System.err.println("Usage: java EnviaArchivo <file_path>");  
            return;  
        }  
  
        String filePath = args[0];  
        File file = new File(filePath);  
  
        if (!file.exists()) {  
            System.err.println("File does not exist.");  
            return;  
        }  
  
        try {  
            EnviaArchivo sender = new EnviaArchivo();  
            sender.send(file);  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

17°C Despejado 10:45 p. m. 28/09/2023

m4gm-learning: Ingresar al sitio Generación de fragmentos de código

chat.openai.com/c/9686328d-feae-4f83-b3a1-1decfa0ecc99

Default (GPT-3.5)

+ Nuevo chat

Hoy

- Generación de fragmento
- Refactorización del envío arc

Últimos 7 Días

- Carbon Footprint of Bitcoin
- Reward for Discovering Radiu
- Arrendatario es el asegurado

Últimos 30 Días

- Archivo único para clases.
- Servidor TCP multithread en
- Calculadora huella carbono.

Actualizar a Plus

Jose Eduardo Olay Silis

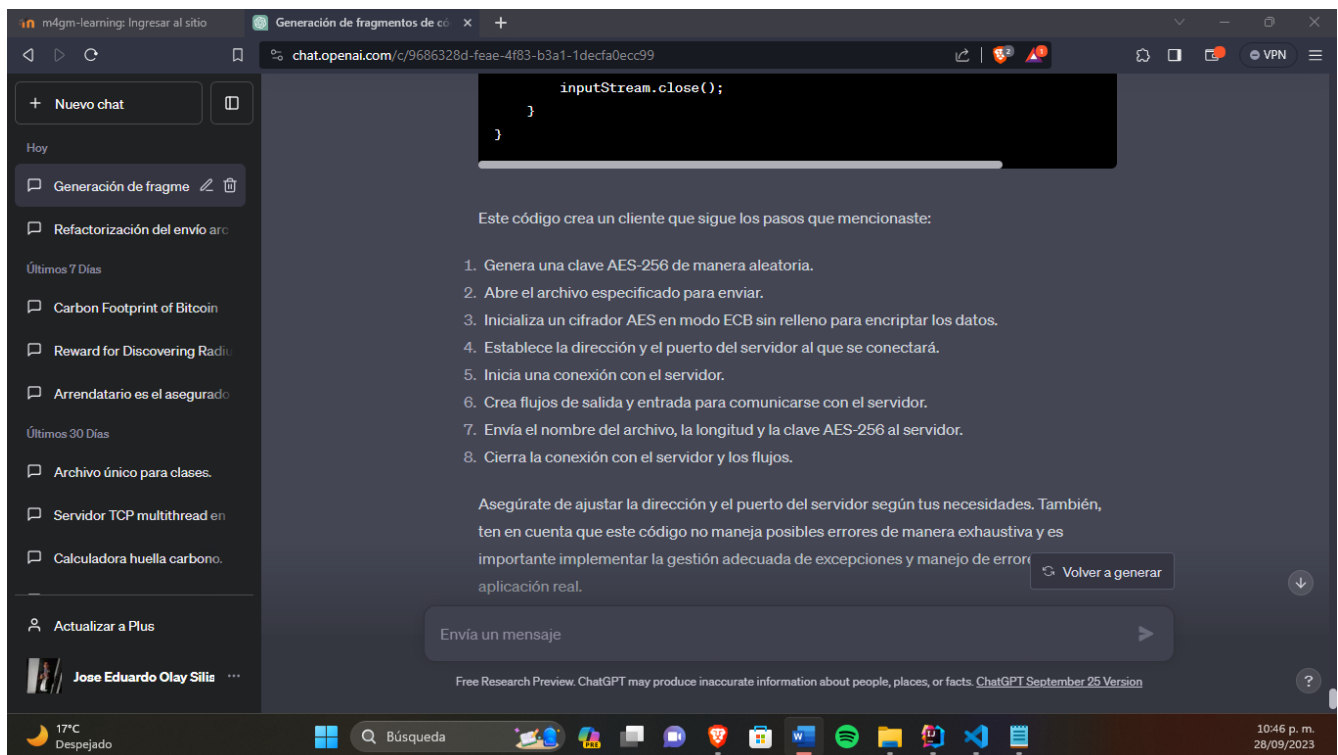
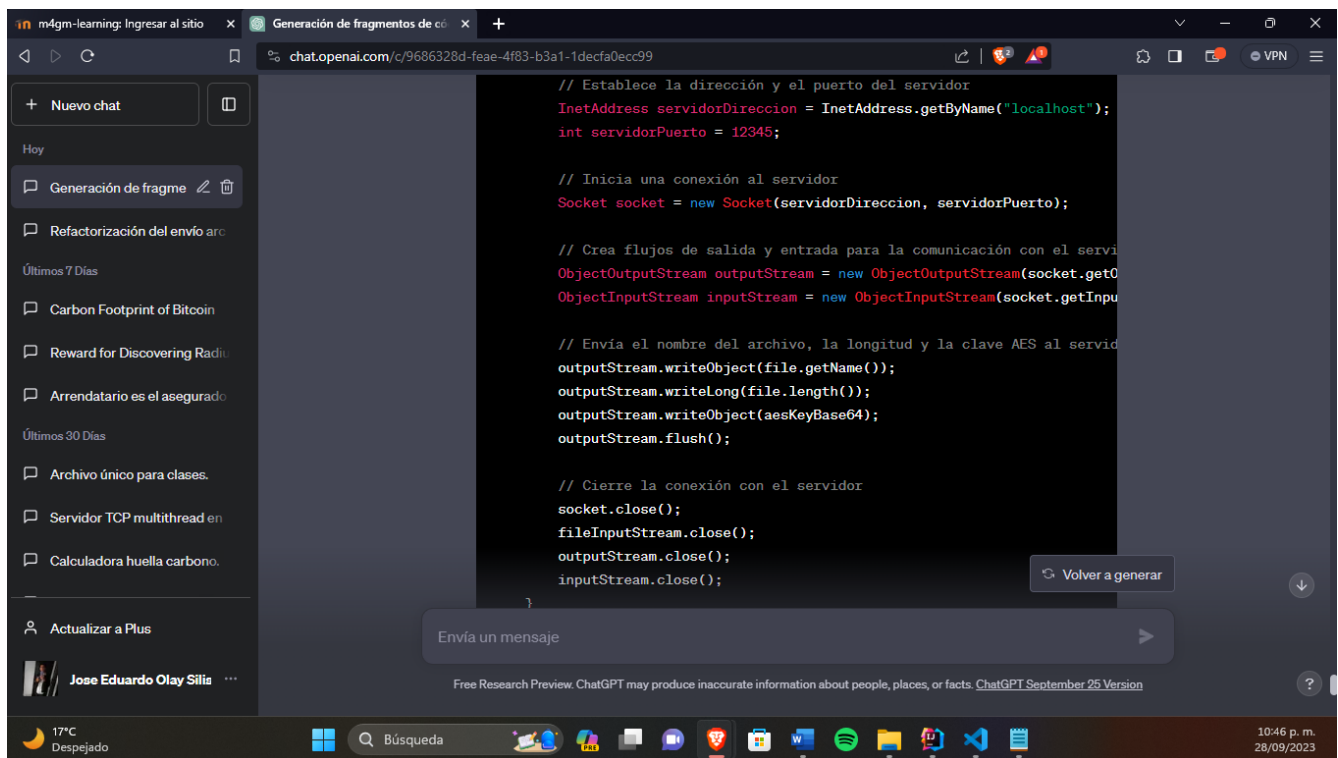
```
public void send(File file) throws Exception {  
    // Genera de manera aleatoria una clave AES-256  
    KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");  
    keyGenerator.init(256);  
    SecretKey aesKey = keyGenerator.generateKey();  
  
    // Convierte la clave en bytes  
    byte[] aesKeyBytes = aesKey.getEncoded();  
  
    // Codifica la clave en Base64  
    String aesKeyBase64 = Base64.getEncoder().encodeToString(aesKeyBytes);  
  
    // Abre el archivo especificado  
    FileInputStream fileInputStream = new FileInputStream(file);  
  
    // Inicializa el cifrador AES en modo ECB sin relleno  
    Cipher cipher = Cipher.getInstance("AES/ECB/NoPadding");  
    cipher.init(Cipher.ENCRYPT_MODE, aesKey);  
}
```

Volver a generar

Envía un mensaje

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version

17°C Despejado 10:45 p. m. 28/09/2023



Dicho programa como tal al momento de compilarlo tal cual se generó con errores debido a que faltaban cosas por agregar. Dichas implementaciones se hicieron de mano propia.

## Fase 2

Una vez teniendo los archivos `EnviaArchivo.java` y `RecibeArchivo.java` se comenzó a realizar la máquina virtual la cual simbolizará el servidor.

Comenzamos creando la máquina virtual con Ubuntu 20 en Azure con los nombres y características específicas que se mencionaron en las instrucciones de la práctica.

Microsoft Azure

Buscar recursos, servicios y documentos (G+)

Inicio > Máquinas virtuales >

## Crear una máquina virtual

Al cambiar opciones básicas se pueden restablecer las selecciones realizadas. Revise todas las opciones antes de crear la máquina virtual.

**Detalles del proyecto**

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción \*

Grupo de recursos \*   
[Crear nuevo](#)

**Detalles de instancia**

Nombre de máquina virtual \*

Región \*

Opciones de disponibilidad

Zona de disponibilidad \*

Ahora puede seleccionar varias zonas. Si selecciona varias zonas, se creará una VM.

[Revisar y crear](#) < Anterior Siguiente: Discos >

[Enviar comentarios](#)

Una vez creada procedemos a ingresar a la máquina creada mediante el ssh

```
System information as of Fri Sep 29 05:12:33 UTC 2023

System load: 0.91      Processes:      107
Usage of /:  5.2% of 28.89GB  Users logged in: 0
Memory usage: 27%      IPv4 address for eth0: 10.0.0.4
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

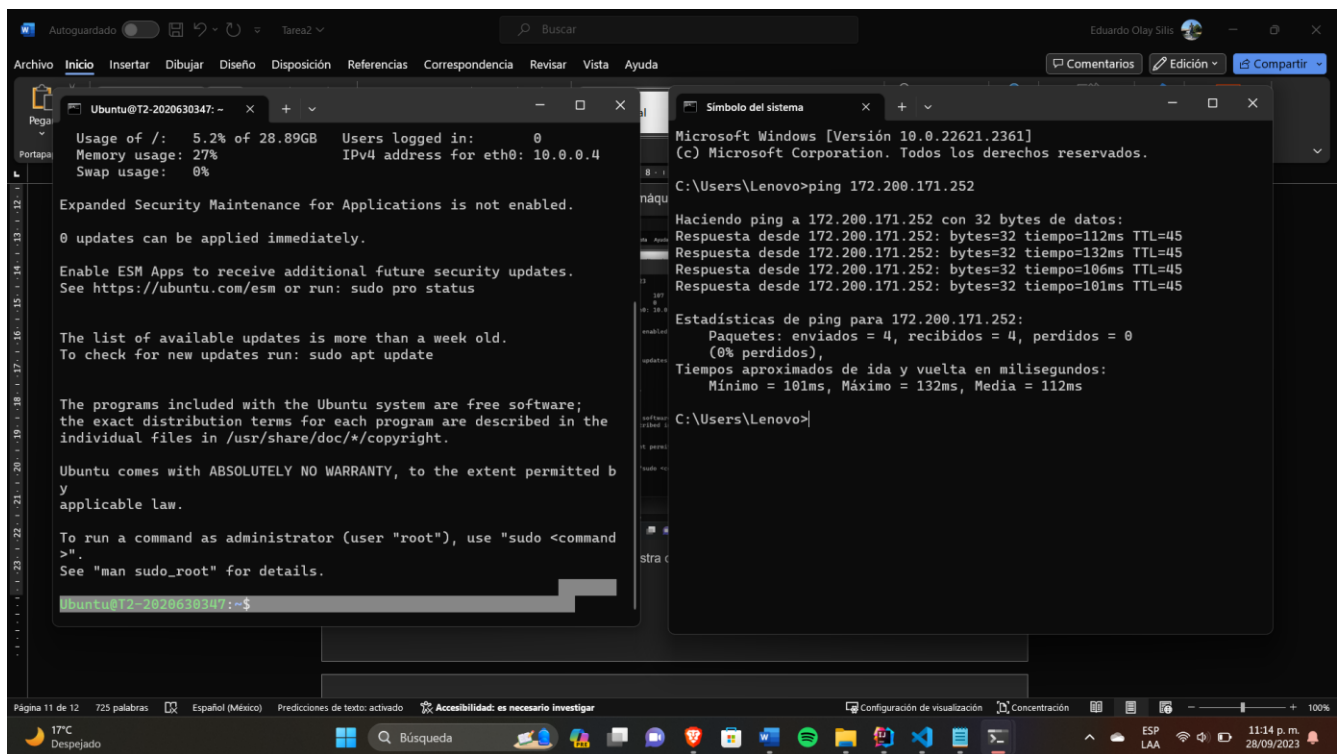
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

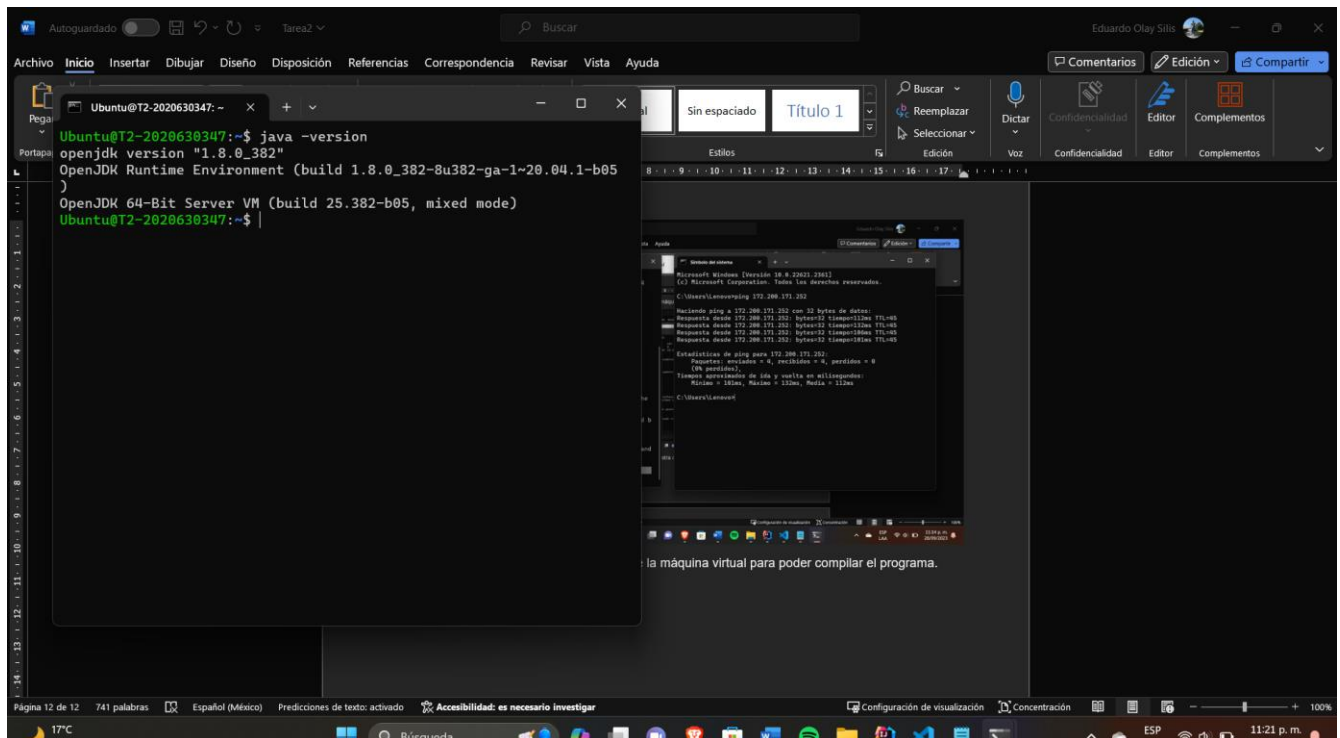
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Ubuntu@T2-2020630347:~$
```

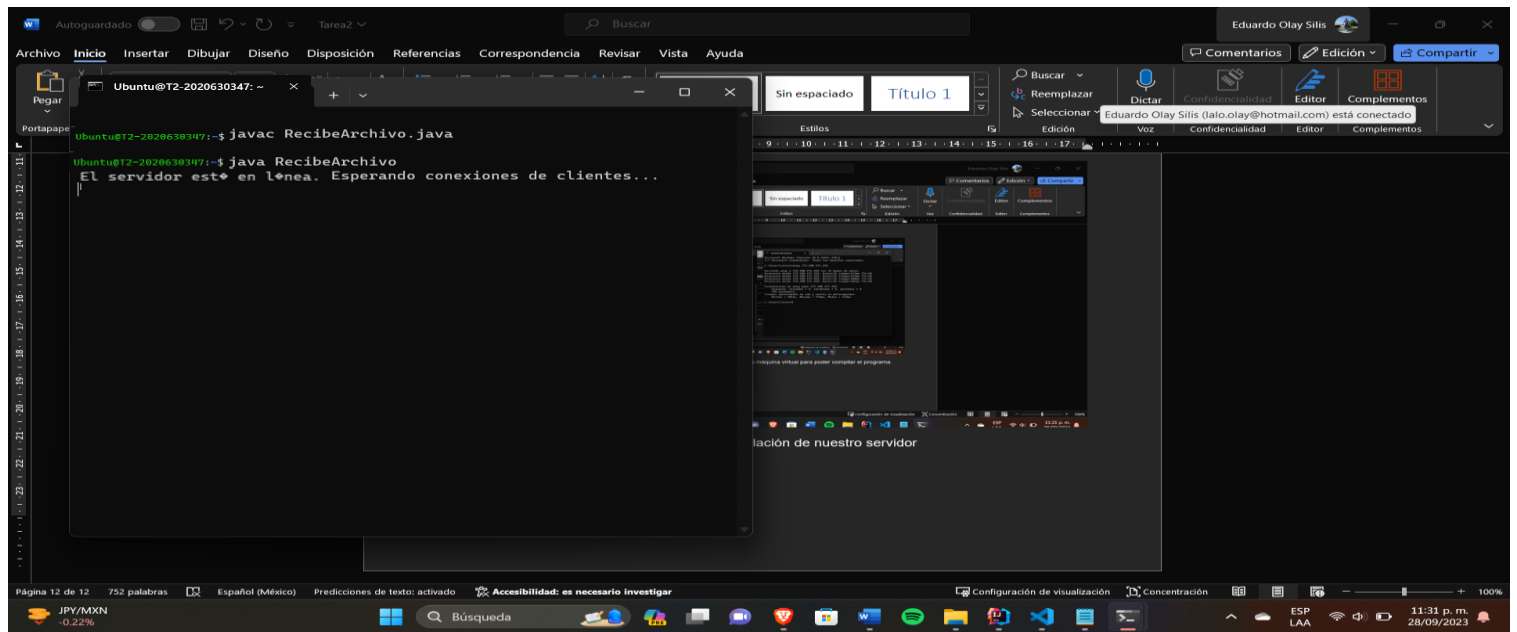
Ahora procedemos a hacerle ping desde nuestra computadora local para verificar la conexión.



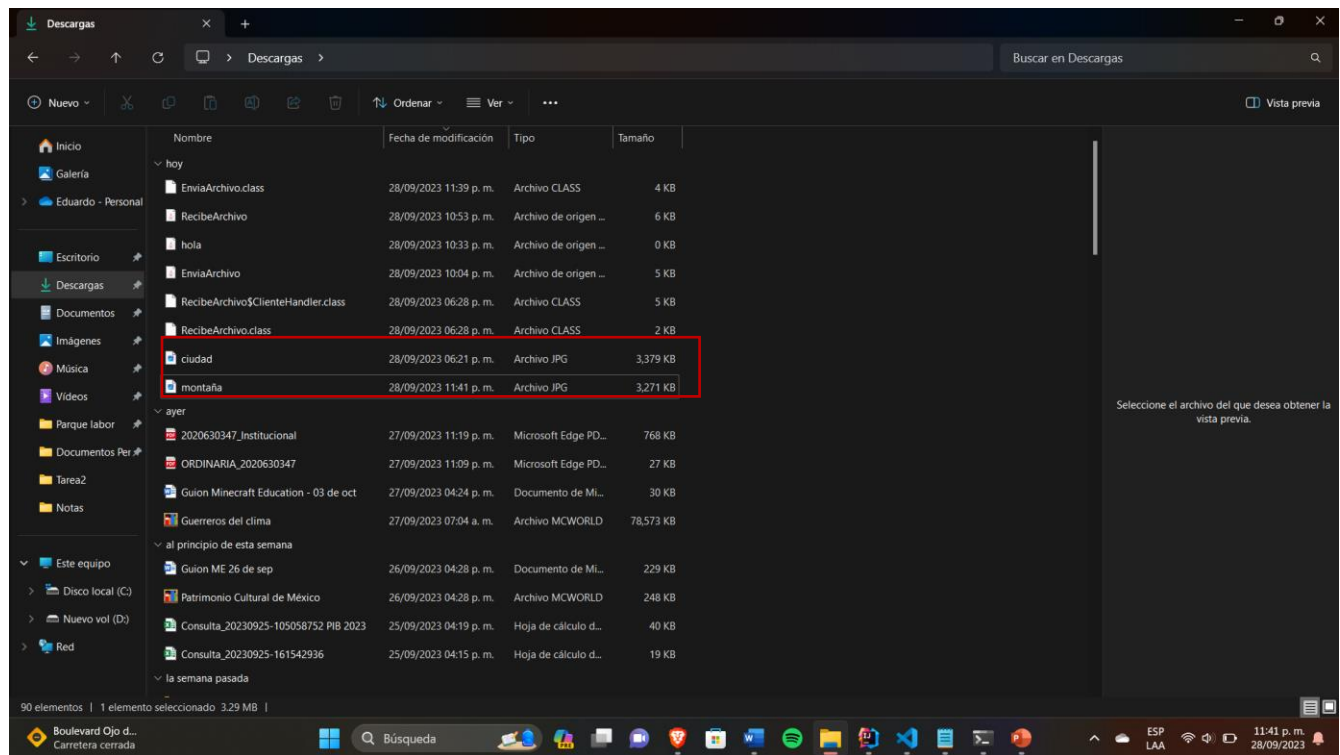
Ahora procedemos a instalar el jdk dentro de la máquina virtual para poder compilar el programa.



Una vez pasado el archivo comenzamos la compilación de nuestro servidor

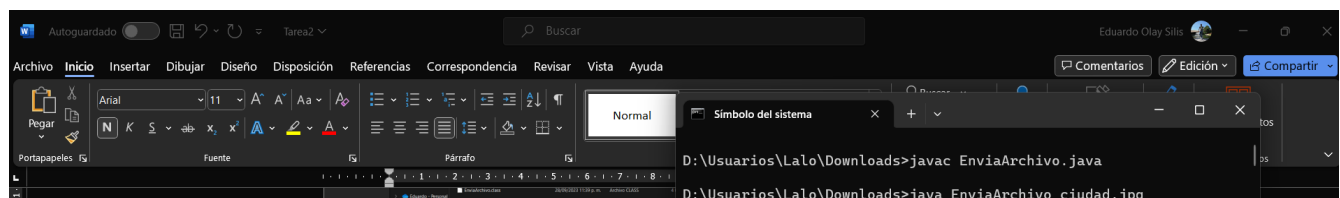


Para el envío de archivos utilizaremos 2 imágenes una de un bosque y la otra de una ciudad



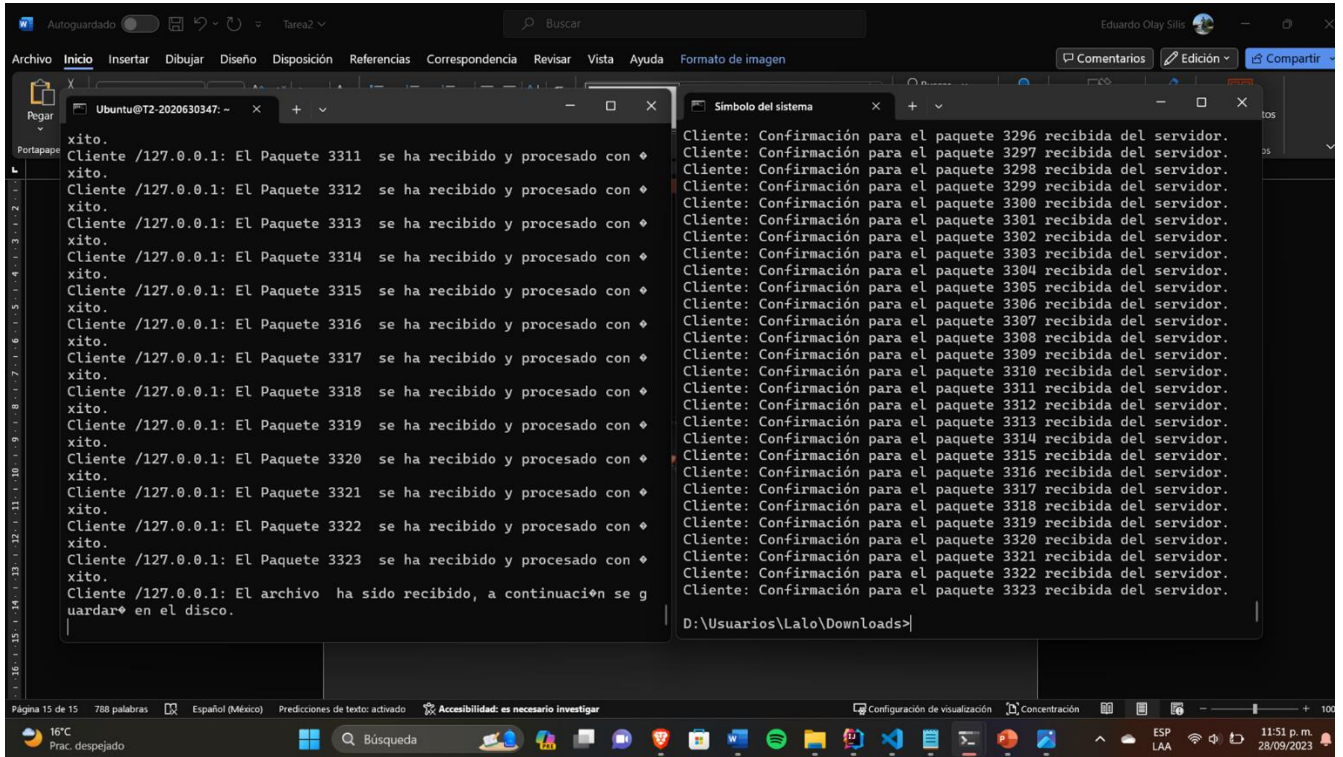
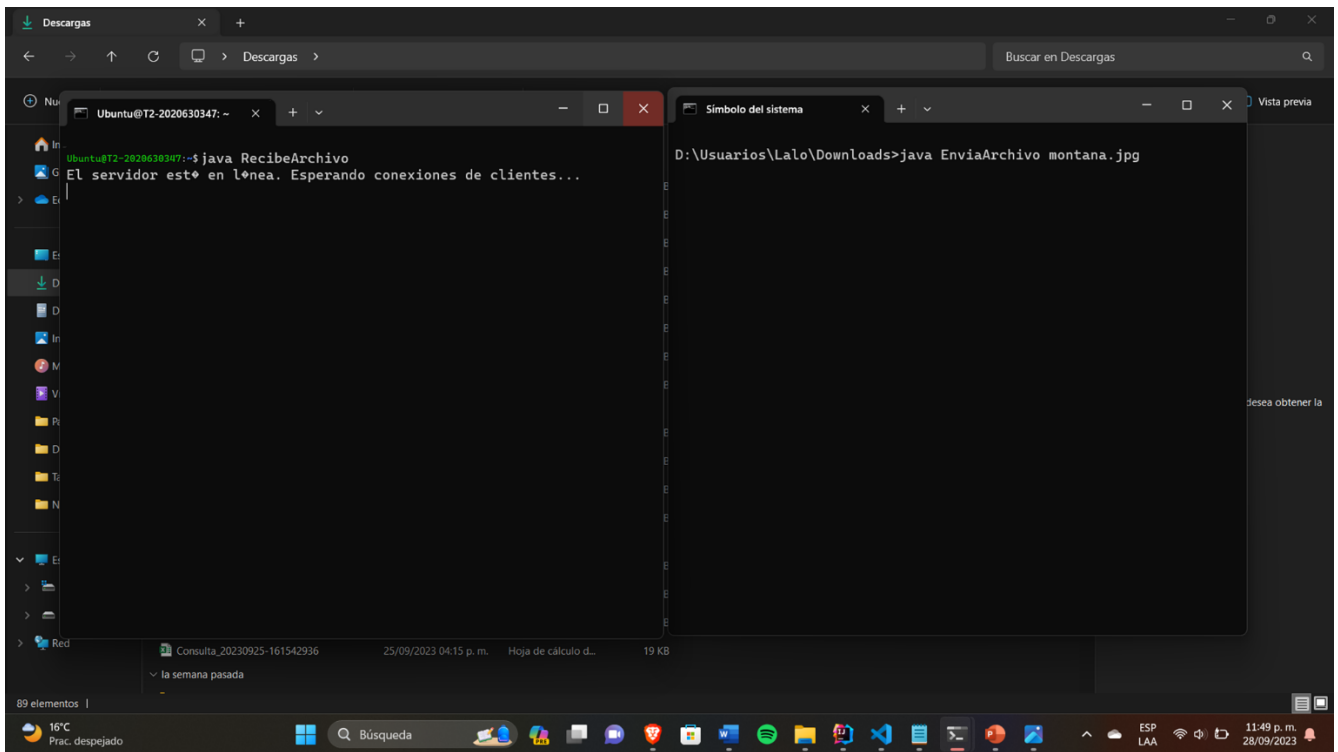
Dando los siguientes resultados

**Envío Ciudad.jpg**









Con esto verificando que ambos paquetes se recibieron sin ninguna pérdida.

## Conclusión:

En este trabajo, hemos desarrollado dos aplicaciones, un servidor llamado "RecibeArchivo" y un cliente llamado "EnviaArchivo", que juntos permiten la transferencia segura y confiable de archivos entre un cliente y un servidor utilizando sockets Datagrama UDP y AES-256 para el cifrado de datos. A continuación, destacamos los puntos clave de nuestra implementación:

**Servidor (RecibeArchivo):** El servidor está diseñado como un servidor multihilo que maneja las solicitudes entrantes de clientes. Implementa una serie de características importantes:

**Confiabilidad:** Utiliza la verificación de suma de comprobación CRC32 para garantizar la integridad de los paquetes recibidos y solicitar reenvíos en caso de errores.

**Descarte de duplicados:** Evita el procesamiento de paquetes duplicados mediante el seguimiento de los identificadores de paquetes recibidos.

**Orden:** Mantiene el orden de los paquetes recibidos utilizando un identificador de paquete esperado.

**Encriptado:** Utiliza el algoritmo de encriptación AES-256 para cifrar los datos transmitidos de manera segura.

**Cliente (EnviaArchivo):** El cliente es responsable de generar una clave AES-256 de manera aleatoria, leer el archivo especificado y enviar el nombre del archivo, su longitud y la clave AES-256 al servidor.

**Generación de clave:** Genera de manera segura una clave AES-256 aleatoria para cada transferencia de archivo.

**Encriptado:** Utiliza el mismo algoritmo AES-256 para cifrar los datos antes de enviarlos al servidor.

**Envío de metadatos:** Envía metadatos importantes al servidor, incluido el nombre del archivo y su longitud, para facilitar la recepción y el procesamiento en el lado del servidor.

En conjunto, estos códigos permiten la transferencia confiable y segura de archivos entre el cliente y el servidor a través de conexiones UDP. La implementación de características como la integridad, el orden, la confiabilidad y el cifrado asegura que los datos se transmitan de manera efectiva y segura entre las partes involucradas. Sin embargo, es importante recordar que en una aplicación en producción, se deben considerar aspectos adicionales como la gestión de errores, la autenticación y la seguridad adicional según los requisitos específicos del sistema.