



Instituto Politécnico Nacional

Escuela Superior de Computo



Tarea 2. Transferencia de archivos utilizando datagramas confiables.

Alumno: Javier Martínez Carranza

Materia: Desarrollo de Sistemas Distribuidos

Grupo: 4CV11

Profesor: Carlos Pineda Guerrero

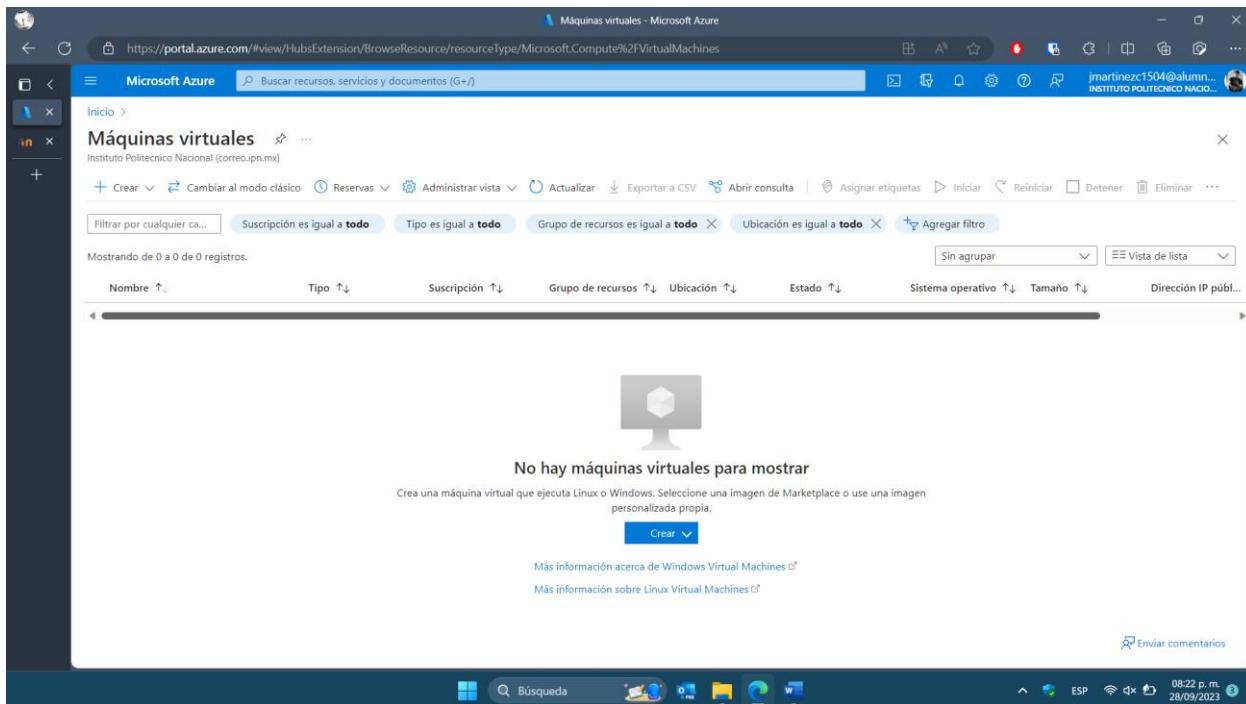
índice

Desarrollo	1
Creación de máquina virtual	1
Conversaciones con las IAs.	1
Códigos	32
Envía Archivo.....	32
Recibe Archivo.....	34
Ejecución y pruebas	37
Conclusión	40

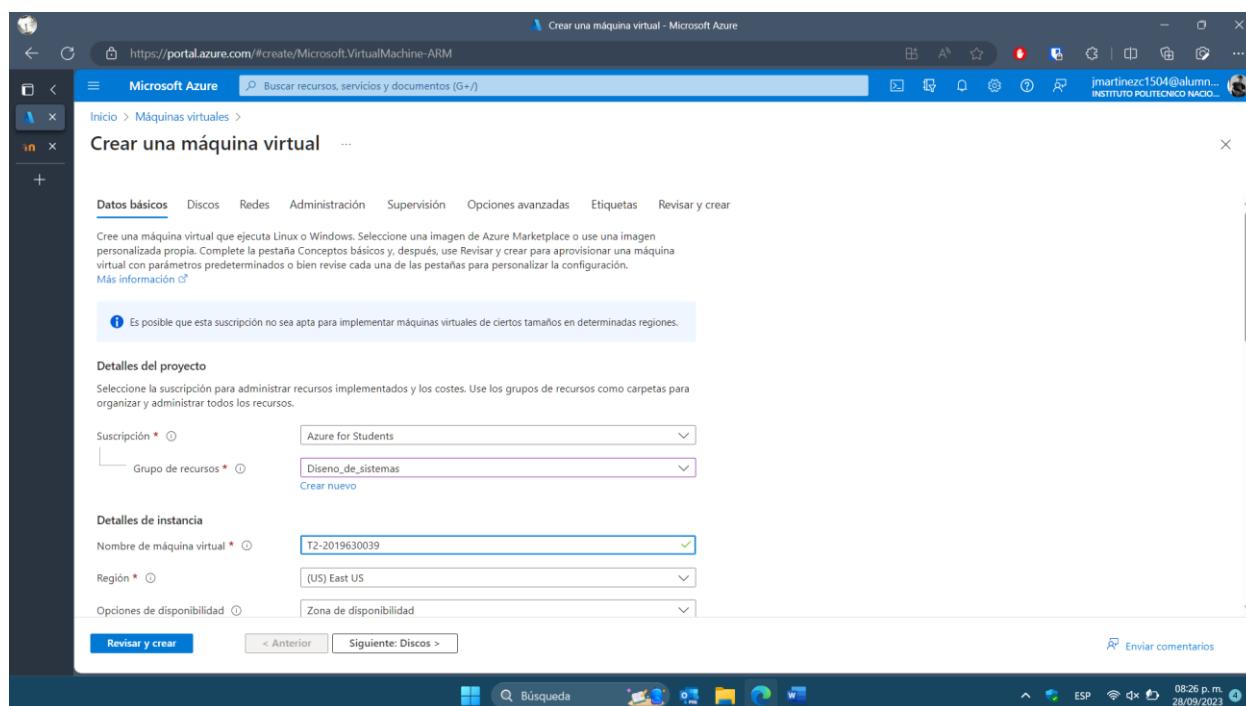
Desarrollo

Creación de máquina virtual

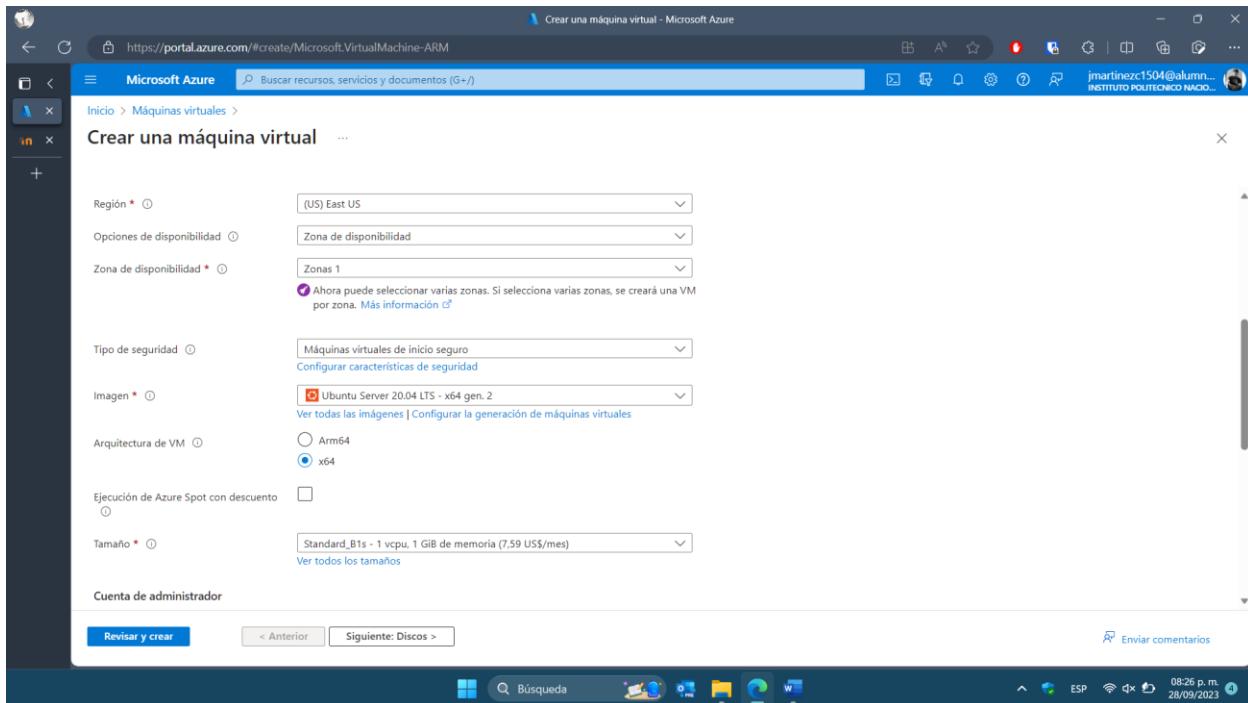
Repasaremos el proceso de creación de la maquina virtual para esta tarea.



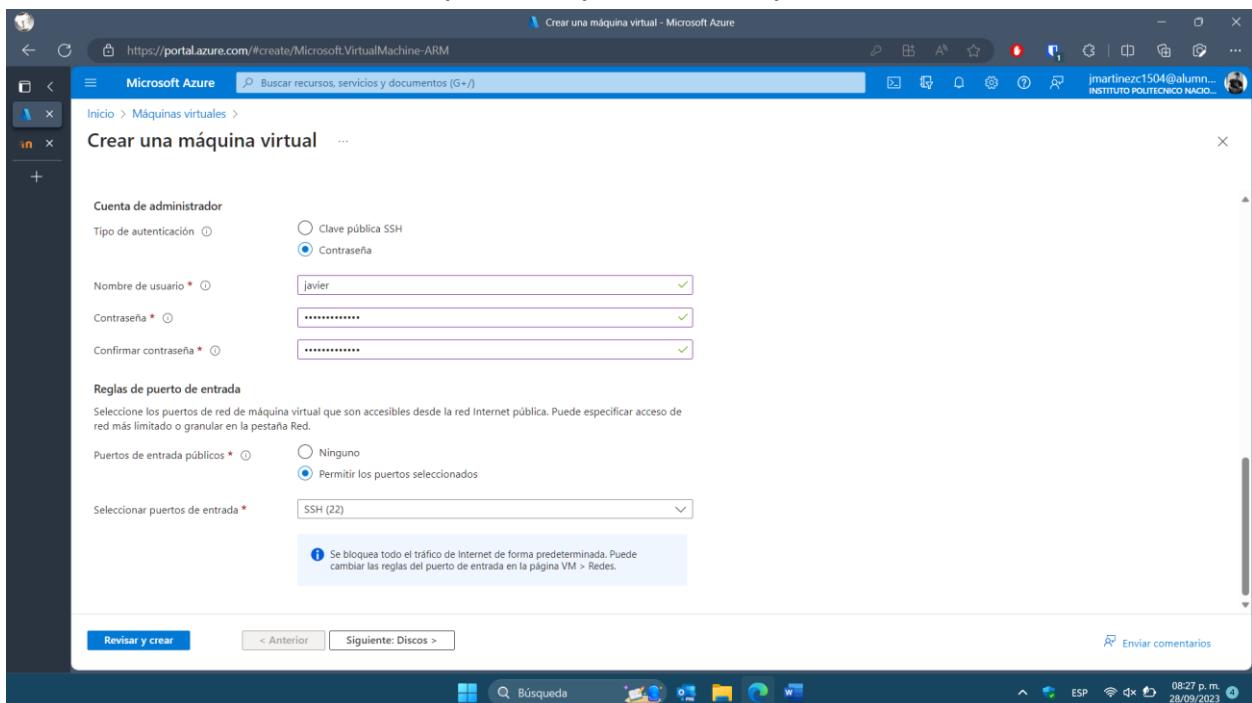
Seleccionamos algún grupo de recursos que tengamos creado, o en su defecto creamos uno nuevo, de igual forma ponemos el nombre como viene indicado en la tarea.



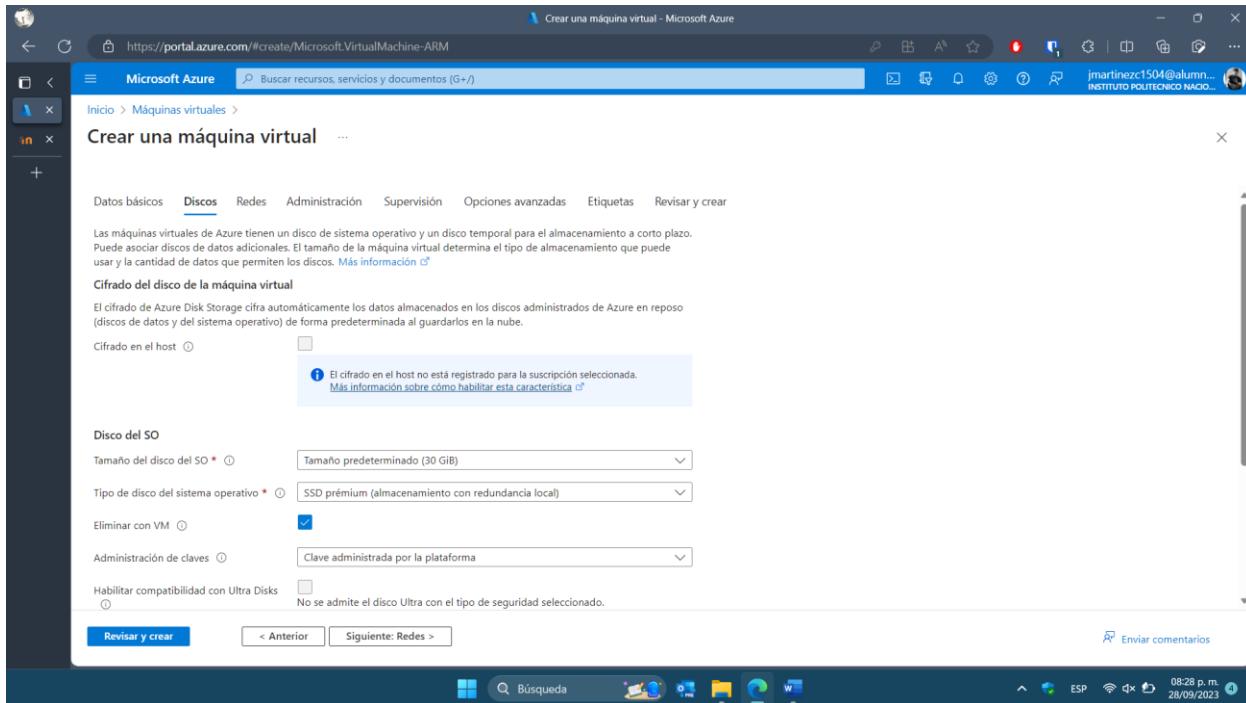
Dejamos la zona por default o en su defecto seleccionamos la que nos cueste más barato, esta máquina funcionará con Ubuntu 20.04 con un tamaño de 1vcpu 1GiB de memoria, el precio por esto será de 7.59\$/mes.



En el tipo de autenticación seleccionaremos contraseña y crearemos un usuario y contraseña para este usuario, de igual forma dejamos por default el puerto SSH(22) con el cual accederemos a la máquina, después de este paso daremos en “Discos”.

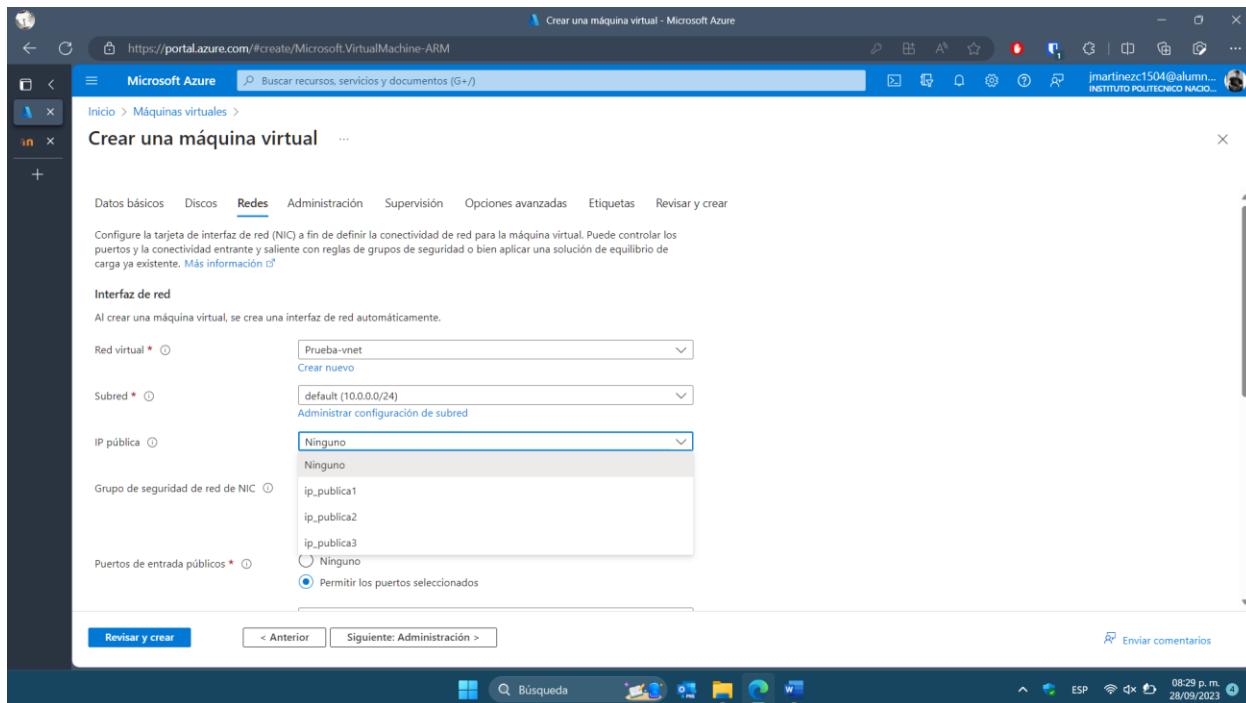


En el apartado de discos dejaremos el tamaño predeterminado de 30GiB, pasaremos a “Redes”.



The screenshot shows the Microsoft Azure portal interface for creating a virtual machine. The top navigation bar includes the Azure logo, search bar, and user information (jmartinezc1504@alumn... INSTITUTO POLITÉCNICO NACIONAL). The main title is "Crear una máquina virtual". The "Discos" tab is selected. The "Disk type" dropdown is set to "SSD premium (almacenamiento con redundancia local)". A note at the bottom left of the form states: "El cifrado en el host no está registrado para la suscripción seleccionada. [Más información sobre cómo habilitar esta característica](#)". Below this, there are fields for "Tamaño del disco del SO" (set to "Tamaño predeterminado (30 GiB)") and "Tipo de disco del sistema operativo" (set to "SSD premium (almacenamiento con redundancia local)"). Other options like "Eliminar con VM" and "Administración de claves" are also visible. At the bottom are "Revisar y crear" and "Siguiente: Redes >" buttons.

En el apartado de Redes/”IP pública” seleccionamos alguna de las 3 disponibles que Azure nos proporciona para usar, en este caso seleccione la primera. Hecho esto damos en “Revisar y Crear”.



The screenshot shows the Microsoft Azure portal interface for creating a virtual machine. The top navigation bar includes the Azure logo, search bar, and user information (jmartinezc1504@alumn... INSTITUTO POLITÉCNICO NACIONAL). The main title is "Crear una máquina virtual". The "Redes" tab is selected. The "IP pública" dropdown is set to "Ninguno". Below it, a list of available network interfaces includes "ip_publica1", "ip_publica2", "ip_publica3", and another "Ninguno" option. The "Permitir los puertos seleccionados" checkbox is checked. At the bottom are "Revisar y crear" and "Siguiente: Administración >" buttons.

Esperamos un momento y si todo es correcto, mostrara la siguiente pantalla, verificamos la información de nuestra maquina virtual y si es correcto, damos en “Crear”

The screenshot shows the 'Create a virtual machine' wizard in Microsoft Azure. The 'Basic information' step has been completed successfully, indicated by a green bar at the top. The configuration details are as follows:

Datos básicos	
Suscripción	Azure for Students
Grupo de recursos	Diseno_de_sistemas
Nombre de máquina virtual	T2-2019630039
Región	East US
Opciones de disponibilidad	Zona de disponibilidad
Zona de disponibilidad	1
Tipo de seguridad	Máquinas virtuales de inicio seguro
Habilitar arranque seguro	Sí
Habilitar vTPM	Sí
Supervisión de integridad	No
Imagen	Ubuntu Server 20.04 LTS - Gen2
Arquitectura de VM	x64
Tamaño	Standard B1s (1 vcpu, 1 GiB de memoria)
Tipo de autenticación	Contraseña
Nombre de usuario	javier
Puertos de entrada públicos	SSH
Azure de acceso puntual	No

At the bottom, there are buttons for 'Create' (highlighted in blue), 'Anterior' (Previous), 'Siguiente' (Next), 'Descargar una plantilla para la automatización' (Download a template for automation), and 'Enviar comentarios' (Send comments).

Después de esperar un tiempo, en nuestras notificaciones se mostrará las opciones para ir al recurso o anclar el panel, en este caso iremos al recurso que es nuestra máquina virtual.

The screenshot shows the 'Deployment Details' blade for the VM 'CreateVm-canonical.0001-com-ubuntu-server-focal-2-20230928202508'. The status is 'Implementación correcta' (Deployment successful). The 'Information general' section shows the deployment was successful on 28/9/2023 at 20:30:41. The 'Implementation' section includes links for 'Ir al recurso' (Go to resource) and 'Anclar al panel' (Pin to dashboard). To the right, there are promotional cards for 'Cost Management', 'Microsoft Defender for Cloud', 'Tutorial gratuitos de Microsoft', and 'Trabajar con un experto'.

Desde aquí podremos iniciar, detener o reiniciar nuestra maquina virtual.

The screenshot shows the Microsoft Azure portal interface. The main title bar says "T2-2019630039 - Microsoft Azure". The left sidebar has a tree view with "Información general" selected. The main content area is titled "Información general" and contains the following details:

Información esencial	
Grupo de recursos (mover)	: Diseño de sistemas
Estado	: En ejecución
Ubicación	: East US (Zona 1)
Suscripción (mover)	: Azure for Students
Id. de suscripción	: 954bd92c-20f2-4909-8ee0-98d8f56076eb
Zona de disponibilidad	: 1
Etiquetas (editar)	: Agregar etiquetas

Below this, there are tabs for "Propiedades", "Supervisión", "Funcionalidades (7)", "Recomendaciones", and "Tutoriales". The "Propiedades" tab is active. On the right side, there is a "Redes" section with network interface details:

Redes	
Dirección IP pública	: 20.75.158.31 (Interfaz de red t2-2019630039992_z1)
Dirección IP pública (IPv6)	-
Dirección IP privada	: 10.0.0.4
Dirección IP privada (IPv6)	-
Red virtual/subred	: Prueba-vnet/default
Nombre DNS	: Configurar

Antes de terminar con nuestra creación de la maquina virtual, abriremos unos puertos en el apartado Redes, en este caso abriremos primero el puerto 50000 y 50001 con TCP.

The screenshot shows the Microsoft Azure portal interface. The left sidebar has a tree view with "Redes" selected. The main content area is titled "Redes" and shows the configuration for the interface "t2-2019630039992_z1".

In the center, there is a table titled "Reglas de puerto de entrada" (Inbound port rules) with the following data:

Prioridad	Nombre	Puerto	Protocolo
300	SSH	22	TCP
65000	AllowVnetInBound	Cualquiera	Cualquier
65001	AllowAzureLoadBalancerInBou...	Cualquiera	Cualquier
65500	DenyAllInBound	Cualquiera	Cualquier

To the right, a modal window titled "Agregar regla de seguridad de entrada" (Add inbound security rule) is open. It shows the configuration for a new rule:

Origen	: Any
Intervalos de puertos de origen	: * (empty)
Destino	: Any
Servicio	: Custom
Intervalos de puertos de destino	: 50000
Protocolo	<input checked="" type="radio"/> TCP
<input type="radio"/> UDP	
<input type="radio"/> ICMP	
Acción	<input checked="" type="radio"/> Permitir
<input type="radio"/> Denegar	

At the bottom of the modal, there are "Agregar" (Add) and "Cancelar" (Cancel) buttons.

Una vez abiertos los puertos ya quedará completado el proceso de creación de nuestra máquina virtual, abrimos dos puertos debido a que uno será para la conexión con sockets seguros SSL y en el otro recibiremos datagramas.

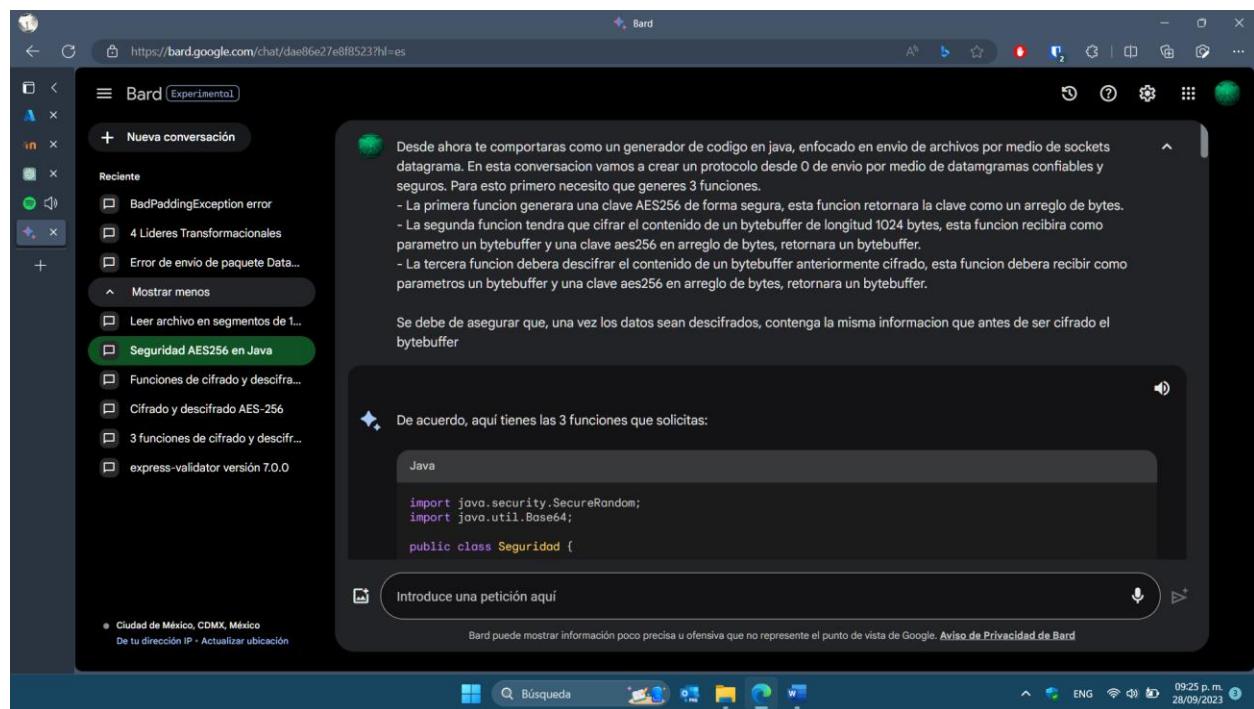
The screenshot shows the Microsoft Azure portal interface for a virtual machine named T2-2019630039. The left sidebar shows navigation options like 'Información general', 'Registro de actividad', and 'Configuración' (selected). The main content area is titled 'Redes' and shows the 'Reglas de puerto de entrada' (Inbound port rules) tab. A success message at the top right says 'Regla de seguridad creada' (Security rule created). The table lists the following rules:

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	SSH	22	TCP	Cualquiera	Cualquiera	Permitir
310	Puerto_50000	50000	TCP	Cualquiera	Cualquiera	Permitir
311	Puerto_50001	50001	Cualquiera	Cualquiera	Cualquiera	Permitir
65000	AllowVnetInBound	Cualquiera	VirtualNetwork	VirtualNetwork	Cualquiera	Permitir
65001	AllowAzureLoadBalancerInBou...	Cualquiera	AzureLoadBalancer	Cualquiera	Cualquiera	Permitir
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	Denegar

Conversaciones con las IAs.

En este apartado veremos como fue el proceso de creación del código tanto de “RecibeArchivo” como de “EnviaArchivo”, para esto, debido a que o existe mucha información de este tipo de ejercicios en internet o las inteligencias artificiales no están tan entrenadas en este aspecto, costo mucho trabajo hacer que generaran el código desde cero en un solo prompt, así que iremos dando contexto sobre lo que queremos hacer yendo de menos a más.

Iniciamos usando Bard para la creación de funciones básicas que usaremos en ambos códigos.



```
Java
import java.security.SecureRandom;
import java.util.Base64;

public class Seguridad {
```

En esta primera petición lo ponemos en contexto, le decimos que use el lenguaje java y que es lo que va a ser el proyecto como tal, le damos un conjunto de pasos, para este caso creara las funciones para crear una clave AES256 y la retornara como un arreglo de bytes, la segunda función será el cifrado de un ByteBuffer con la clave creada y el ultimo el descifrado de un ByteBuffer.

Nos devuelve las funciones pero sin la parte de ByteBuffer, así que se lo haremos saber con la siguiente petición.

The screenshot shows the Bard experimental interface. On the left sidebar, under 'Recente', 'Seguridad AES256 en Java' is selected. The main area displays Java code for generating a random key and performing encryption and decryption using AES-256 in CBC mode. The code uses Cipher.getInstance("AES/CBC/PKCS5Padding") and cipher.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(clave, "AES")) for encryption, and cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(clave, "AES")) for decryption. A message at the bottom says: 'modifica la función cifrar y descifrar para que retornen un ByteBuffer al ser ejecutados.'

```
public class Seguridad {  
    public static byte[] generarClaveAES256() {  
        // Generamos una cadena aleatoria de 32 bytes  
        byte[] clave = new byte[32];  
        new SecureRandom().nextBytes(clave);  
  
        // Convertimos la cadena a base64 para poder usarla en Java  
        return Base64.getEncoder().encode(clave);  
    }  
  
    public static byte[] cifrar(byte[] datos, byte[] clave) throws Exception {  
        // Creamos un cifrador AES en modo CBC  
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");  
        cipher.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(clave, "AES"));  
  
        // Ciframos los datos  
        byte[] bufferCifrado = cipher.doFinal(datos);  
  
        return bufferCifrado;  
    }  
  
    public static byte[] descifrar(byte[] datosCifrados, byte[] clave) throws Exception {  
        // Creamos un descifrador AES en modo CBC  
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");  
        cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(clave, "AES"));  
  
        // Desciframos los datos  
        byte[] bufferDescifrado = cipher.doFinal(datosCifrados);  
    }  
}
```

Introduce una petición aquí

The screenshot shows the Bard experimental interface. The user has entered the instruction: 'modifica la función cifrar y descifrar para que retornen un ByteBuffer al ser ejecutados.' The response provides the modified Java code where the cipher and decipher methods now return a ByteBuffer instead of a byte[] array. The code imports java.nio.ByteBuffer and java.security.SecureRandom, and uses ByteBuffer.allocate(32) for generating the key. The cipher and decipher methods are modified to return ByteBuffer objects using ByteBuffer.wrap(bufferCifrado) and ByteBuffer.wrap(bufferDescifrado) respectively.

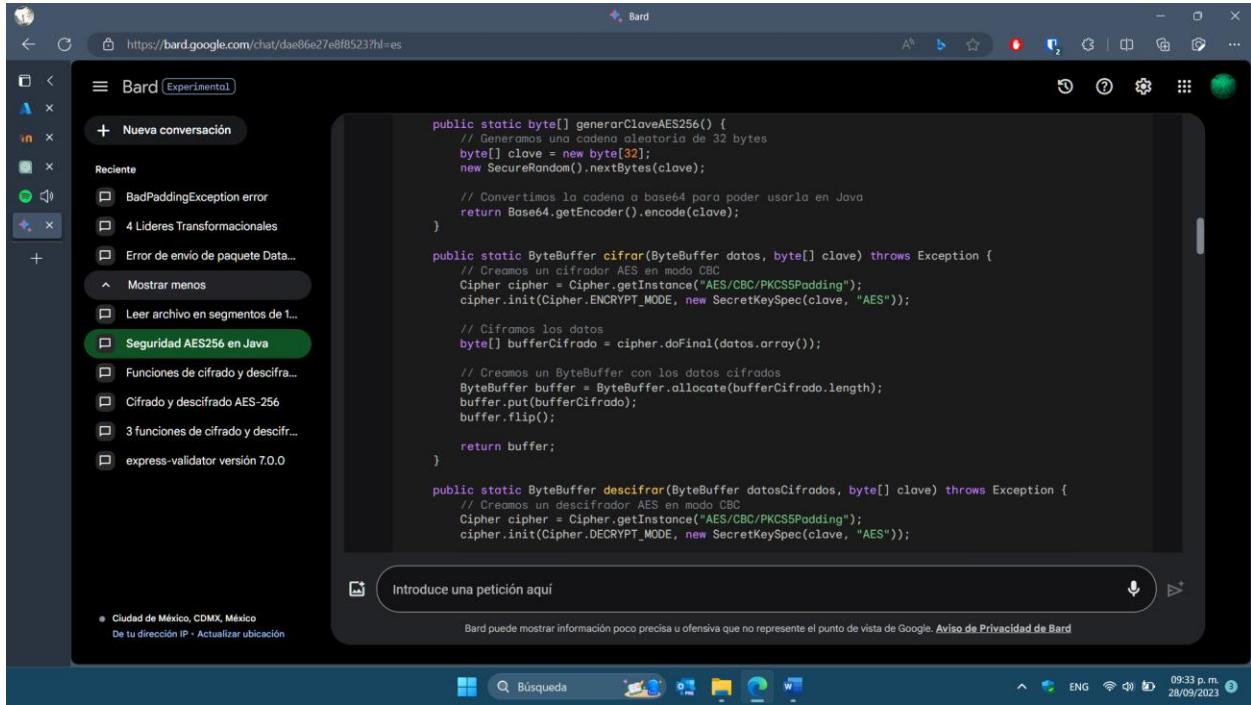
modifica la función cifrar y descifrar para que retornen un ByteBuffer al ser ejecutados.

De acuerdo, aquí tienes las funciones `cifrar()` y `descifrar()` modificadas para que retornen un `ByteBuffer`:

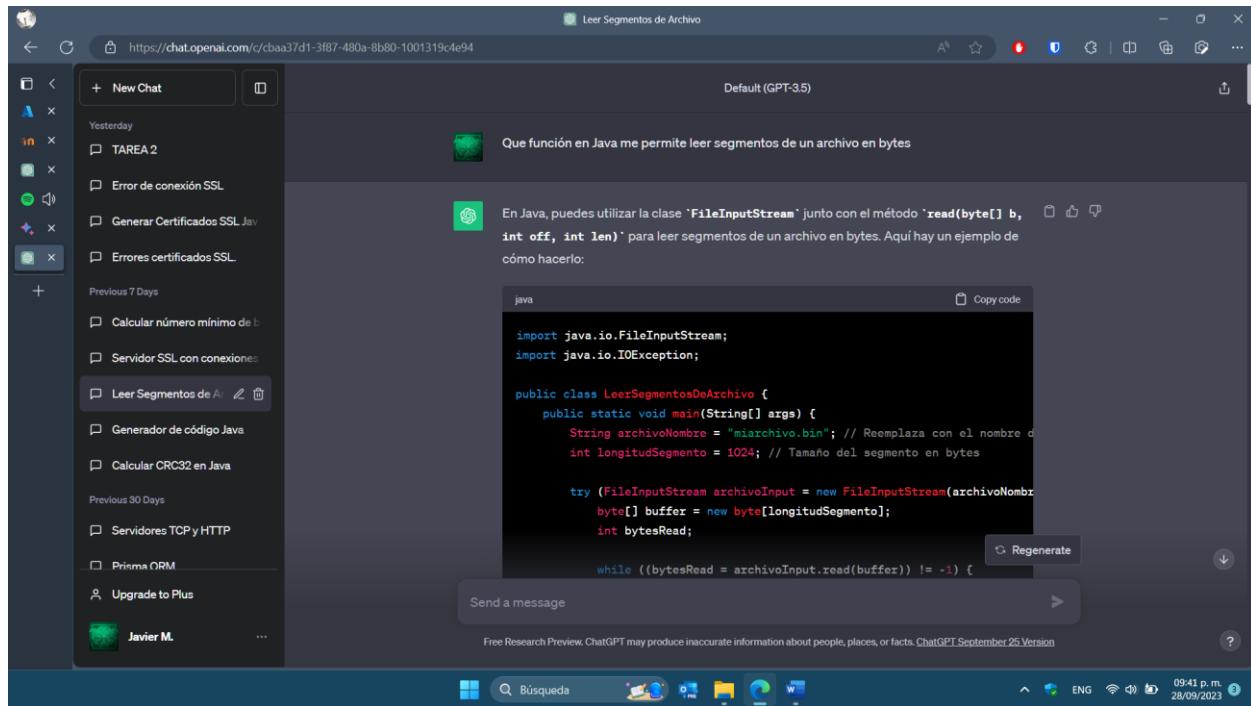
```
import java.nio.ByteBuffer;  
import java.security.SecureRandom;  
import java.util.Base64;  
  
public class Seguridad {  
  
    public static byte[] generarClaveAES256() {  
        // Generamos una cadena aleatoria de 32 bytes  
        byte[] clave = new byte[32];  
        new SecureRandom().nextBytes(clave);  
  
        // Convertimos la cadena a base64 para poder usarla en Java  
        return Base64.getEncoder().encode(clave);  
    }  
  
    public static ByteBuffer cifrar(ByteBuffer datos, byte[] clave) throws Exception {  
        // Creamos un cifrador AES en modo CBC  
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");  
        cipher.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(clave, "AES"));  
  
        // Ciframos los datos  
        byte[] bufferCifrado = cipher.doFinal(datos.array());  
  
        return ByteBuffer.wrap(bufferCifrado);  
    }  
  
    public static ByteBuffer descifrar(ByteBuffer datosCifrados, byte[] clave) throws Exception {  
        // Creamos un descifrador AES en modo CBC  
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");  
        cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(clave, "AES"));  
  
        // Desciframos los datos  
        byte[] bufferDescifrado = cipher.doFinal(datosCifrados.array());  
  
        return ByteBuffer.wrap(bufferDescifrado);  
    }  
}
```

Introduce una petición aquí

Vemos como ahora si nos devuelve las funciones de la forma que queríamos.



Pasamos a la lectura del archivo, para este caso nos pasaremos a chatGPT, dado que fue el que mejor respondió y entendió nuestras peticiones. Para esta parte fuimos de lo mas básico, hasta lo más específico. Como vemos le preguntamos sobre como leería archivos en segmentos de bytes.



El siguiente código es bastante preciso ya que de un archivo va leyendo segmentos de 1024 bytes, en cada iteración se lee y a la par se verifica si se está leyendo algo o ya se leyó todo lo del archivo. Dentro de bytesRead, tenemos la cantidad de bytes que se leyeron en esa iteración y en nuestro buffer tenemos el segmento de información.

The screenshot shows a ChatGPT interface with a sidebar containing various AI-generated topics like 'TAREA 2', 'Error de conexión SSL', etc. The main area displays Java code for reading file segments:

```
public class LeeSegmentosDeArchivo {
    public static void main(String[] args) {
        String archivoNombre = "miarchivo.bin"; // Reemplaza con el nombre de tu archivo
        int longitudSegmento = 1024; // Tamaño del segmento en bytes

        try (FileInputStream archivoInput = new FileInputStream(archivoNombre);
            byte[] buffer = new byte[longitudSegmento];
            int bytesRead;
        ) {
            while ((bytesRead = archivoInput.read(buffer)) != -1) {
                // Aquí puedes procesar el segmento de bytes en 'buffer'
                // 'bytesRead' contiene el número de bytes leídos en este segmento
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Below the code, a note explains its function:

Este código abrirá el archivo especificado ('`miarchivo.bin`' en este ejemplo), leerá segmentos de bytes de tamaño '`longitudSegmento`', y puedes procesar esos segmentos como deseas dentro del bucle '`while`'.

At the bottom right, there's a 'Regenerate' button and a timestamp '09:43 p.m. 28/09/2023'.

A continuación, le especificamos al chat que el usuario pasará el archivo por la línea de comandos.

The screenshot shows the same ChatGPT interface. The user has asked to modify the code so the file is passed as a command-line argument. The AI has provided the modified code and explained how to use it:

Para modificar el código de manera que el archivo sea pasado como argumento de línea de comandos y verificar que el archivo existe en la ubicación proporcionada, puedes hacer lo siguiente:

```
java
import java.io.FileInputStream;
import java.io.IOException;
import java.io.File;

public class LeeSegmentosDeArchivo {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Por favor, proporcione la ubicación del archivo");
            return;
        }

        String archivoNombre = args[0];
        File archivo = new File(archivoNombre);

        if (!archivo.exists()) {
            System.out.println("El archivo no existe");
            return;
        }

        try (FileInputStream archivoInput = new FileInputStream(archivo);
            byte[] buffer = new byte[1024];
            int bytesRead;
        ) {
            while ((bytesRead = archivoInput.read(buffer)) != -1) {
                // Aquí puedes procesar el segmento de bytes en 'buffer'
                // 'bytesRead' contiene el número de bytes leídos en este segmento
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

At the bottom right, there's a 'Regenerate' button and a timestamp '09:49 p.m. 28/09/2023'.

The screenshot shows a ChatGPT interface. On the left, a sidebar lists recent conversations: BufferOverflowException en, Leadership Styles and Applic, Errores y mejoras., TAREA2, Error de conexión SSL, Generar Certificados SSL.Jav, Errores certificados SSL., Leer Segmentos de Ar, Generador de código Java, Upgrade to Plus, and Javier M.. The main area displays a Java code snippet for reading file segments:

```
        return;

    String archivoNombre = args[0];
    File archivo = new File(archivoNombre);

    if (!archivo.exists()) {
        System.out.println("El archivo no existe en la ubicación propuesta");
        return;
    }

    int longitudSegmento = 1024; // Tamaño del segmento en bytes

    try (FileInputStream archivoInput = new FileInputStream(archivo)) {
        byte[] buffer = new byte[longitudSegmento];
        int bytesRead;

        while ((bytesRead = archivoInput.read(buffer)) != -1) {
            // Procesa el segmento de bytes en 'buffer'
            System.out.println("Bytes leidos: " + bytesRead);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

At the bottom right of the code editor, there is a "Regenerate" button. Below the code editor is a message input field with the placeholder "Send a message" and a "Send" button. A small note at the bottom states: "Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version".

Pasamos indicarle unas modificaciones para ir preparando el código que queremos exportar a nuestro programa final.

The screenshot shows a continuation of the ChatGPT session. The user has asked for modifications to the code. The AI has responded with the modified code:

```
quiero que hagas las siguientes modificaciones al código:
1. Crea una variable final estática que se llame LONGITUD, esta variable tendrá el valor de 1012 y reemplazara a la variable "longitudSegmento".
2. Si bytesRead es menor a 1012, copiaras los primeros bytesRead del buffer y lo almacenaras en otro buffer, por ende este buffer solo tendrá los bytesRead y no 1012.
3. Agrega una variable de control que incremente con cada ciclo y agrega un if dentro del ciclo para que, si la variable es igual a 100, imprime "Llego a 100", sino, que siga incrementando.

Aqui está el código modificado según tus indicaciones:
```

```
java
import java.io.FileInputStream;
import java.io.IOException;
import java.io.File;

public class LeerSegmentosDeArchivo {
    // Paso 1: Crear una variable final estática llamada LONGITUD
    private static final int LONGITUD = 1012;

    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Por favor, proporcione la ubicación del archivo");
        }
    }
}
```

The AI has also provided a "Copy code" button. The rest of the interface is identical to the first screenshot, including the sidebar, message input field, and status bar.

The screenshot shows a ChatGPT interface with a sidebar of previous conversations. The main area displays the following Java code:

```
String archivoNombre = args[0];
File archivo = new File(archivoNombre);

if (!archivo.exists()) {
    System.out.println("El archivo no existe en la ubicación proporcionada");
    return;
}

try (FileInputStream archivoInput = new FileInputStream(archivo)) {
    byte[] buffer = new byte[LONGITUD];
    int bytesRead;
    int ciclo = 0; // Paso 3: Variable de control

    while ((bytesRead = archivoInput.read(buffer)) != -1) {
        // Paso 2: Si bytesRead es menor a LONGITUD, copiar los primeros bytesRead bytes
        byte[] subBuffer = new byte[bytesRead];
        System.arraycopy(buffer, 0, subBuffer, 0, bytesRead);

        // Procesa el segmento de bytes en 'subBuffer' (que tiene bytesRead)
        System.out.println("Bytes leidos: " + bytesRead);

        // Paso 3: Verifica si la variable de control llegó a 100
        if (ciclo == 100) {
            System.out.println("Llegó a 100");
            break; // Sale del bucle después de llegar a 100
        } else {
            ciclo++; // Incrementa la variable de control
        }
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

At the bottom, there is a message input field and a note about the research preview.

Nuevamente el chat nos responde de forma un poco confusa, el código no esta correcto en su totalidad, por lo que debemos hacer cambios a este antes de exportarlo.

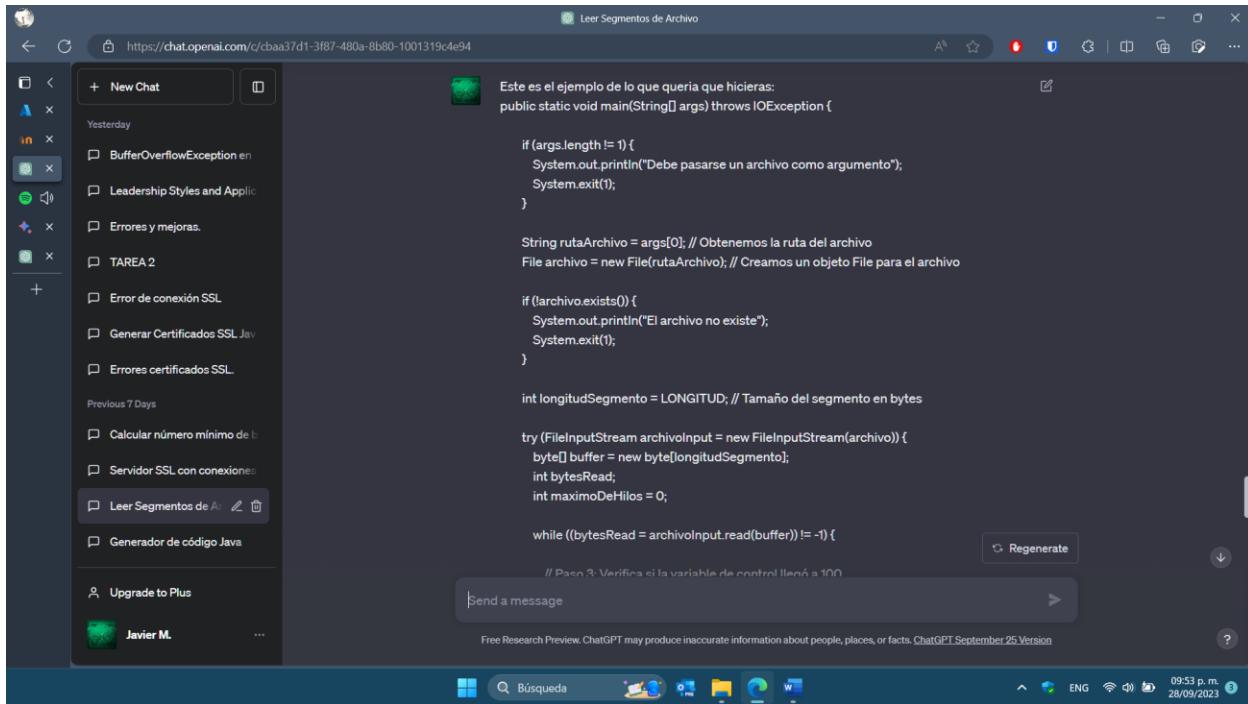
The screenshot shows a ChatGPT interface with a sidebar of previous conversations. The main area displays the corrected Java code:

```
ciclo++; // Incrementa la variable de control
}
} catch (IOException e) {
    e.printStackTrace();
}
}

En este código:
1. Se crea una variable final estática llamada 'LONGITUD' con el valor 1012, que reemplaza a 'longitudSegmento'.
2. Se verifica si 'bytesRead' es menor que 'LONGITUD', y si es así, se crea un nuevo buffer 'subBuffer' que contiene solo los primeros 'bytesRead' del buffer original. Esto asegura que 'subBuffer' tenga solo la cantidad de bytes leídos en esa iteración.
3. Se agrega una variable de control llamada 'ciclo' que se incrementa en cada iteración. Dentro del bucle 'while', se verifica si 'ciclo' es igual a 100 y se imprime "Llegó a 100". Si no es igual a 100, se incrementa la variable de control y el ciclo continúa.
```

At the bottom, there is a message input field and a note about the research preview.

En este caso nosotros hacemos las modificaciones

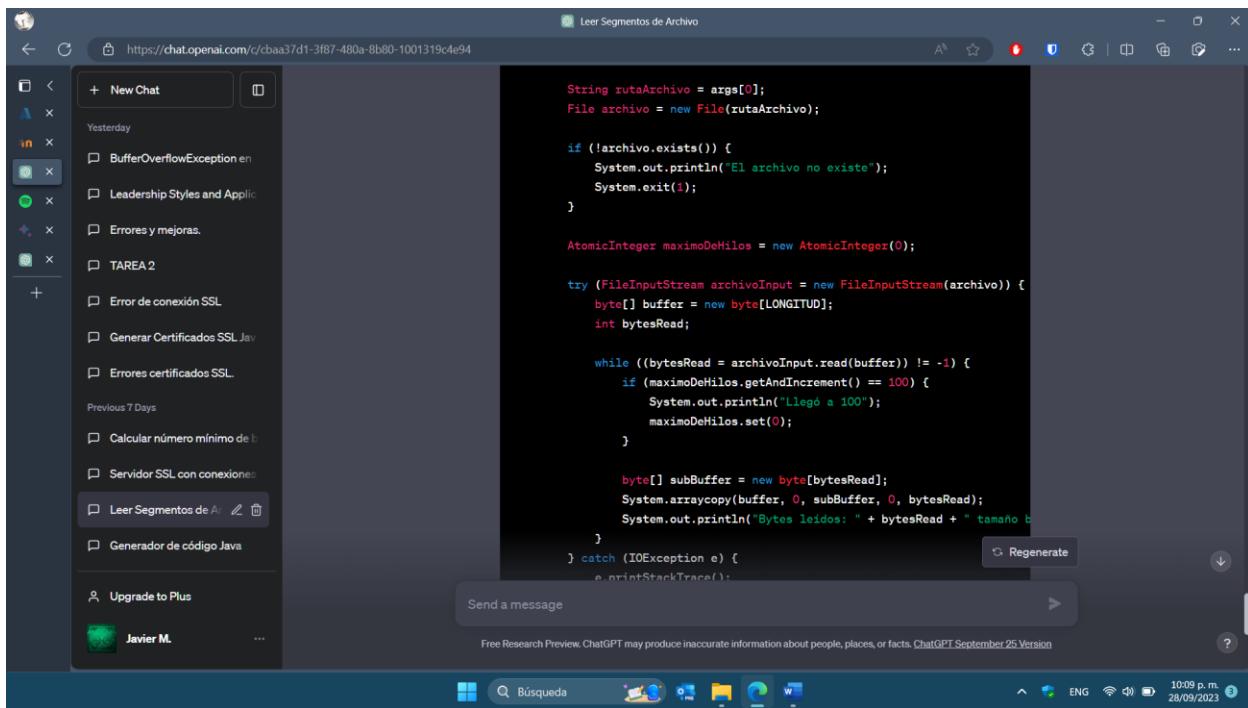
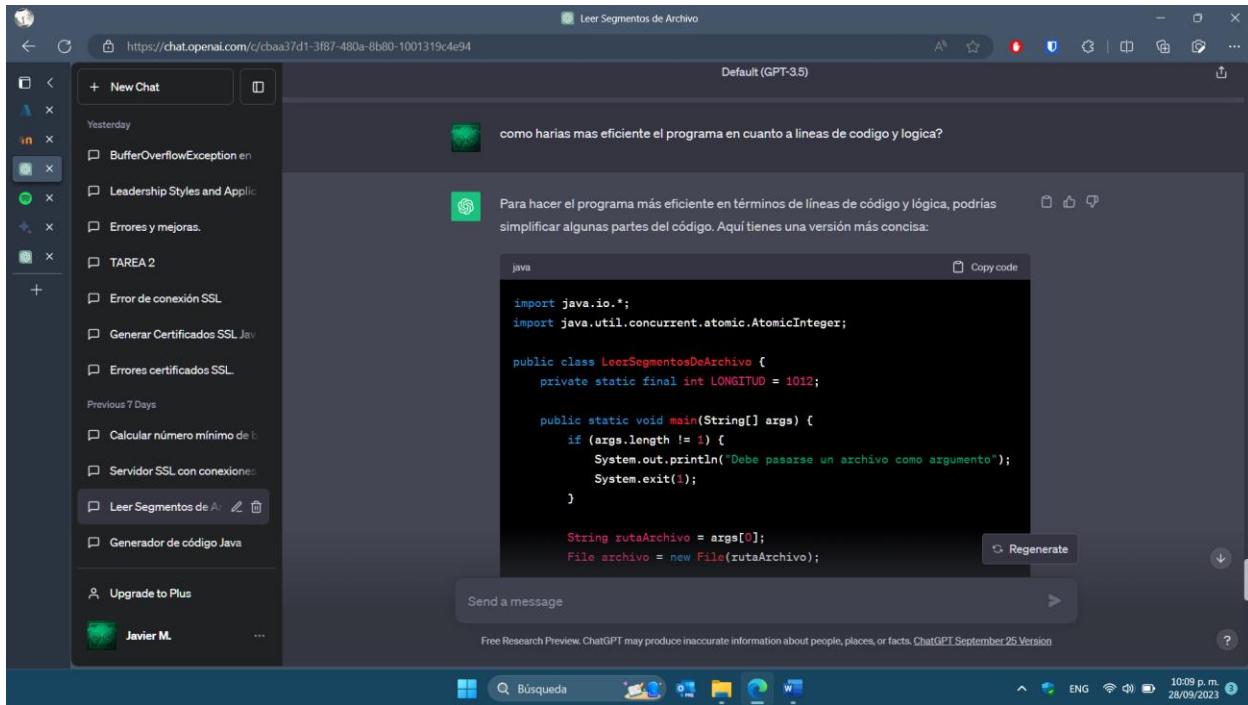


En este caso queríamos que verificara si se había llegado al máximo de hilos y también si la longitud de lo leído en la iteración, acoplamos el buffer a la longitud de lo leído:

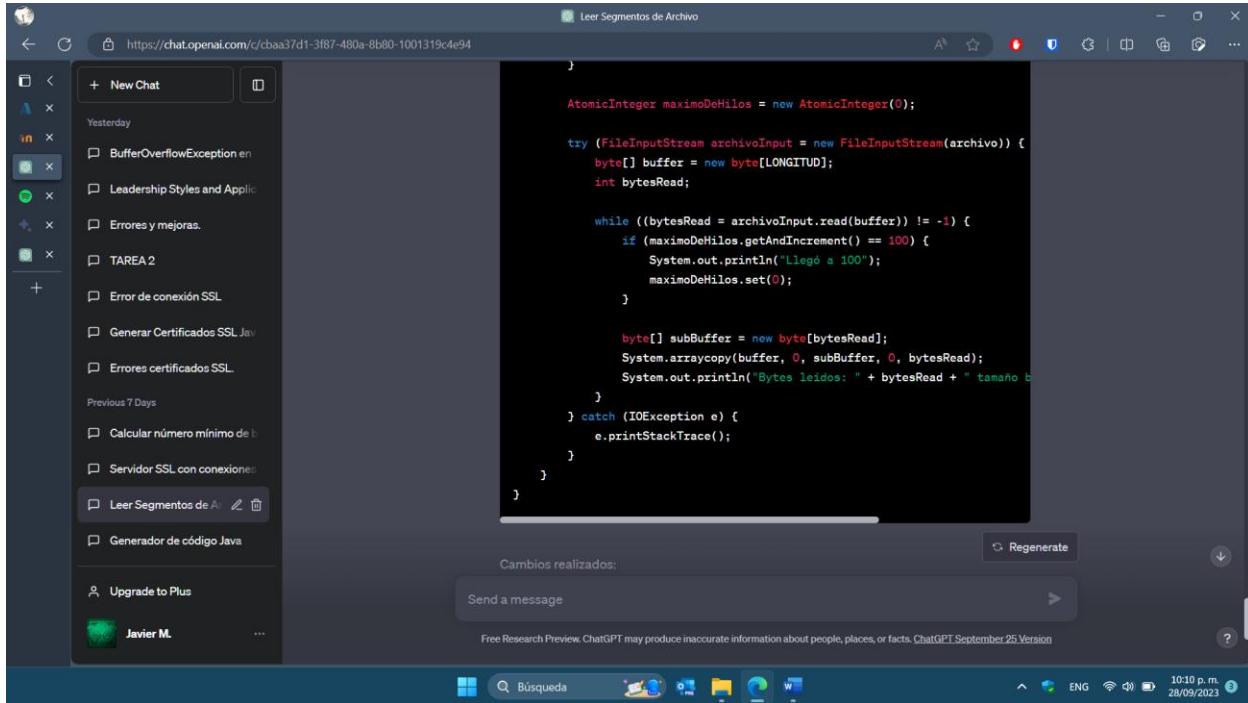
```
while ((bytesRead = archivoInput.read(buffer)) != -1) {
```

```
    if (maximoDeHilos == 100) {  
  
        System.out.println("Llegó a 100");  
        // Aquí se espera a que todos los hilos hayan acabado  
        maximoDeHilos = 0;  
    } else {  
        maximoDeHilos++; // Incrementa la variable de control  
    }  
  
    if (bytesRead < LONGITUD) {  
        byte[] subBuffer = new byte[bytesRead];  
        System.arraycopy(buffer, 0, subBuffer, 0, bytesRead);  
        System.out.println("Bytes leídos: " + bytesRead + " tamano buffer: " +  
subBuffer.length);  
    } else {  
        // Procesa el segmento de bytes en 'buffer'  
        System.out.println("Bytes leídos: " + bytesRead + " tamano buffer: " +  
buffer.length);  
    }  
}
```

Finalmente le preguntamos sobre si es posible optimizar más el código con cuanto a lógica con lo cual nos presenta una opción interesante.



En esta opción nos da la opción de volver nuestra variable que verifica si llego a los 100 hilos un AtomicInteger(), así podemos incrementar y verificar en una sola línea.



The screenshot shows a ChatGPT session titled "Leer Segmentos de Archivo". The sidebar contains a list of previous conversations and a "Ler Segmentos de Arquivo" entry. The main area displays the following Java code:

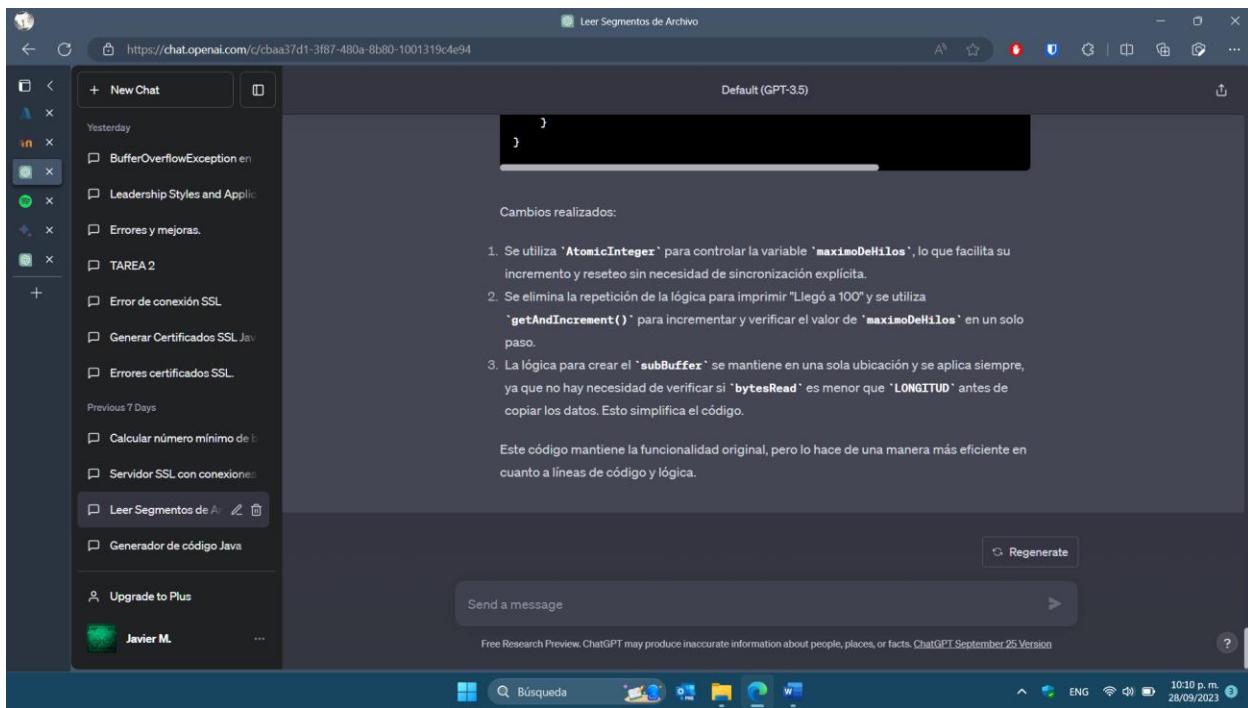
```
AtomicInteger maximoDeHilos = new AtomicInteger(0);

try (FileInputStream archivoInput = new FileInputStream(archivo)) {
    byte[] buffer = new byte[LONGITUD];
    int bytesRead;

    while ((bytesRead = archivoInput.read(buffer)) != -1) {
        if (maximoDeHilos.getAndIncrement() == 100) {
            System.out.println("Llegó a 100");
            maximoDeHilos.set(0);
        }

        byte[] subBuffer = new byte[bytesRead];
        System.arraycopy(buffer, 0, subBuffer, 0, bytesRead);
        System.out.println("Bytes leidos: " + bytesRead + " tamaño b");
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

The code reads a file in segments of size `LONGITUD`. It uses an `AtomicInteger` to keep track of the number of segments read. When it reaches 100 segments, it prints a message and resets the counter. A note at the bottom states: "Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version".



The screenshot shows the same ChatGPT session. The sidebar and URL are identical. The main area now displays a simplified version of the Java code:

```
}
```

A list of changes is shown below the code:

1. Se utiliza 'AtomicInteger' para controlar la variable 'maximoDeHilos', lo que facilita su incremento y reseteo sin necesidad de sincronización explícita.
2. Se elimina la repetición de la lógica para imprimir "Llegó a 100" y se utiliza 'getAndIncrement()' para incrementar y verificar el valor de 'maximoDeHilos' en un solo paso.
3. La lógica para crear el 'subBuffer' se mantiene en una sola ubicación y se aplica siempre, ya que no hay necesidad de verificar si 'bytesRead' es menor que 'LONGITUD' antes de copiar los datos. Esto simplifica el código.

A note at the bottom states: "Este código mantiene la funcionalidad original, pero lo hace de una manera más eficiente en cuanto a líneas de código y lógica." The footer indicates the session is from "28/09/2023" at "10:10 p.m."

Pasamos al apartado del CRC32, para este apartado simplemente pediremos que genere una función para generar un CRC32 y otra para verificar si es correcto un CRC32. Como podemos ver le decimos que la longitud del CRC32 será de 1016 bytes que es la suma de la información más el id (1012+4).

The screenshot shows a ChatGPT session titled "Calcular CRC32 en Java". The sidebar on the left lists various tasks, including "Calcular CRC32 en Java" which is currently selected. The main area displays a message from ChatGPT: "Puedes generar un valor CRC32 en Java utilizando la clase 'java.util.zip.CRC32' que está disponible en el JDK 11. Aquí tienes una función que toma un 'ByteBuffer' de longitud 1016 como parámetro y calcula el CRC32:". Below this is a code block:

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

public class CRC32Generator {

    public static long calcularCRC32(ByteBuffer buffer) {
        CRC32 crc32 = new CRC32();
        crc32.update(buffer);
        return crc32.getValue();
    }

    public static void main(String[] args) {
        ByteBuffer buffer = ByteBuffer.allocate(1016); // Crea un ByteBuffer
        // Llena el ByteBuffer con datos, por ejemplo:
        // buffer.put(data);
        long crcValue = calcularCRC32(buffer);
        System.out.println("CRC32: " + crcValue);
    }
}
```

At the bottom right of the code block is a "Regenerate" button.

The screenshot shows a continuation of the ChatGPT session. The sidebar still has "Calcular CRC32 en Java" selected. The main area now displays a message: "hazme una función que verifique si el crc enviado es correcto con el de la data enviada." Below this, another message says: "Para verificar si un valor CRC32 enviado coincide con el CRC32 calculado para ciertos datos, puedes crear una función en Java como la siguiente:"

```
java
import java.util.zip.CRC32;

public class CRC32Verifier {

    public static boolean verificarCRC32(byte[] data, long crcEnviado) {
        CRC32 crc32 = new CRC32();
        crc32.update(data);
        long crcCalculado = crc32.getValue();
        return crcEnviado == crcCalculado;
    }

    public static void main(String[] args) {
        // Datos recibidos y CRC32 enviado desde alguna fuente externa
        byte[] dataRecibida = obtenerDatosRecibidos(); // Reemplaza esto con tu código
        long crcEnviado = obtenerCRCEnviado(); // Reemplaza esto con tu código
        boolean esCorrecto = verificarCRC32(dataRecibida, crcEnviado);
    }
}
```

At the bottom right of the code block is a "Regenerate" button.

The screenshot shows a ChatGPT session titled "Calcular CRC32 en Java". The sidebar on the left lists various tasks such as "TAREA2", "Error de conexión SSL", "Generar Certificados SSL Java", etc. The main area displays the following Java code:

```

public static void main(String[] args) {
    // Datos recibidos y CRC32 enviado desde alguna fuente externa
    byte[] dataRecibida = obtenerDatosRecibidos(); // Reemplaza esto con
    long crcEnviado = obtenerCRCEnviado(); // Reemplaza esto con el CRC32

    boolean esCorrecto = verificarCRC32(dataRecibida, crcEnviado);

    if (esCorrecto) {
        System.out.println("El CRC32 es correcto.");
    } else {
        System.out.println("El CRC32 no coincide. Los datos pueden haber sido alterados.");
    }
}

// Función ficticia para obtener datos recibidos (debes implementarla)
private static byte[] obtenerDatosRecibidos() {
    // Simplemente reemplaza esto con la obtención real de tus datos
    return new byte[] { 0x01, 0x02, 0x03, 0x04 };
}

// Función ficticia para obtener el CRC32 enviado (debes implementarla)
private static long obtenerCRCEnviado() {
    // Simplemente reemplaza esto con la obtención real del CRC32 recibido
    return 0x12345678;
}

```

At the bottom right of the code editor, there is a "Regenerate" button.

Finalmente pasamos a juntar todas estas funciones con los servidores, pero para esto debemos darle el contexto de lo que queremos que haga con estas funciones y el motivo final.

The screenshot shows a ChatGPT session titled "TAREA 2". The sidebar on the left lists various tasks. The main area displays the following Java code:

```

public static void main(String[] args) throws Exception {
    System.setProperty("javax.net.ssl.keyStore", "keystore_cliente.jks");
    System.setProperty("javax.net.ssl.keyStorePassword", "123456");

    SSLSocketFactory cliente = (SSLSocketFactory) SSLSocketFactory.getDefault();
    Socket conexion = cliente.createSocket("40.88.227.203", 50000);
    DataOutputStream salida = new DataOutputStream(conexion.getOutputStream());

    Thread.sleep(1000);
    conexion.close();
}

```

Below the code, ChatGPT provides an analysis:

Este código en Java parece ser un programa de cliente que establece una conexión SSL con un servidor remoto en la dirección IP "40.88.227.203" en el puerto 50000. A continuación, analizaré el código paso a paso:

1. Se configuran las propiedades del sistema para especificar el archivo de almacenamiento de claves.

At the bottom right of the code editor, there is a "Regenerate" button.

En este momento el chat ya entiende el código principal de un servidor SSL.

Una vez entienda que queremos, le decimos que agregue las funciones que hemos creado para crear claves AES256.

The screenshot shows a ChatGPT session titled "TAREA 2". The user has asked ChatGPT to add functions to generate AES256 keys. ChatGPT has provided the following Java code:

```
Desde este momento, nos referiremos a este código como "EnviaArchivo", haremos modificaciones para que funcione según lo que queremos, las modificaciones son:  
1. Agregar las siguientes funciones al código:  
public static byte[] generarClaveAES256() {  
    SecureRandom random = new SecureRandom(); // Generamos un generador de  
números aleatorios seguro  
    byte[] clave = new byte[32]; // Generamos un arreglo de bytes de 256 bits (32 bytes)  
    random.nextBytes(clave); // Llenamos el arreglo de bytes con números aleatorios  
    return clave;  
}  
  
public static ByteBuffer cifrar(ByteBuffer buffer, byte[] clave) throws Exception {  
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); // Instanciamos un  
cifrador AES-256  
    cipher.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(clave, "AES")); // Inicializamos el  
cifrador con la clave  
    return ByteBuffer.wrap(cipher.doFinal(buffer.array())); // Ciframos el contenido del buffer  
}  
  
public static ByteBuffer descifrar(ByteBuffer buffer, byte[] clave) throws Exception {  
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); // Instanciamos un  
descifrador AES-256  
    cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(clave, "AES")); // Inicializamos el  
descifrador con la clave  
    return ByteBuffer.wrap(cipher.doFinal(buffer.array())); // Desciframos el contenido del  
buffer  
}
```

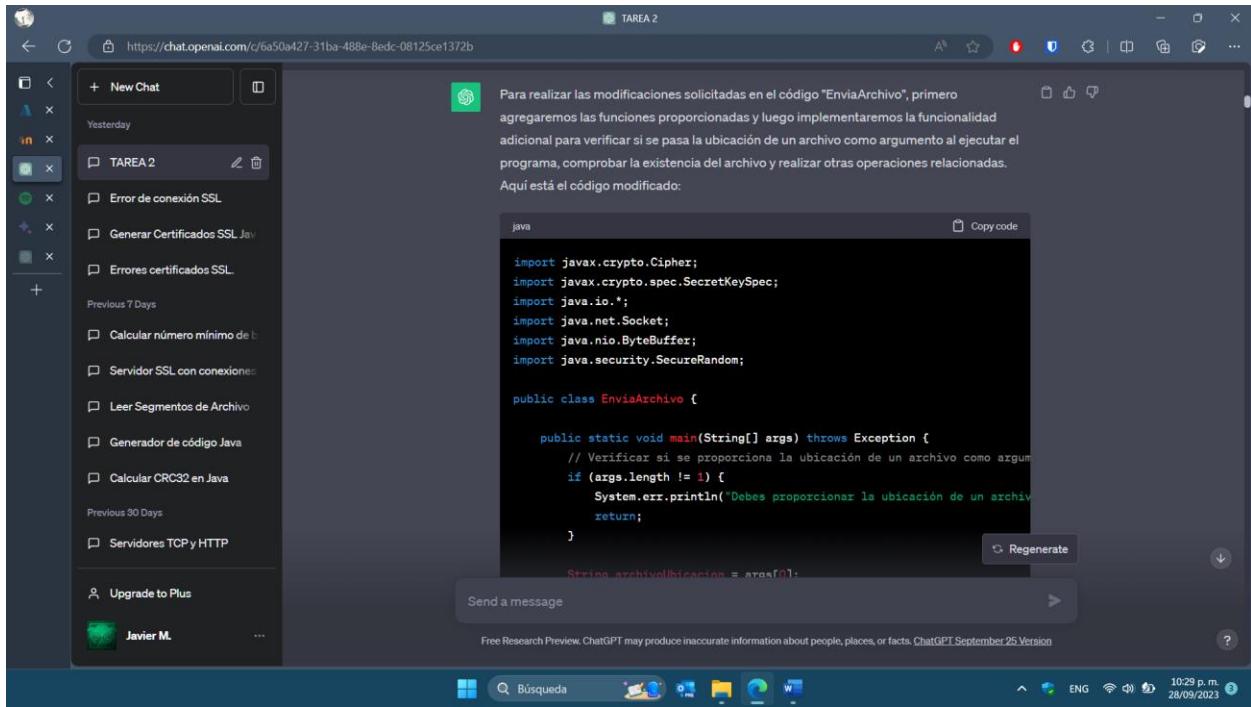
At the bottom right of the code block, there is a "Regenerate" button.

De igual forma vamos preparando al código principal para que use estas funciones con la lectura del archivo.

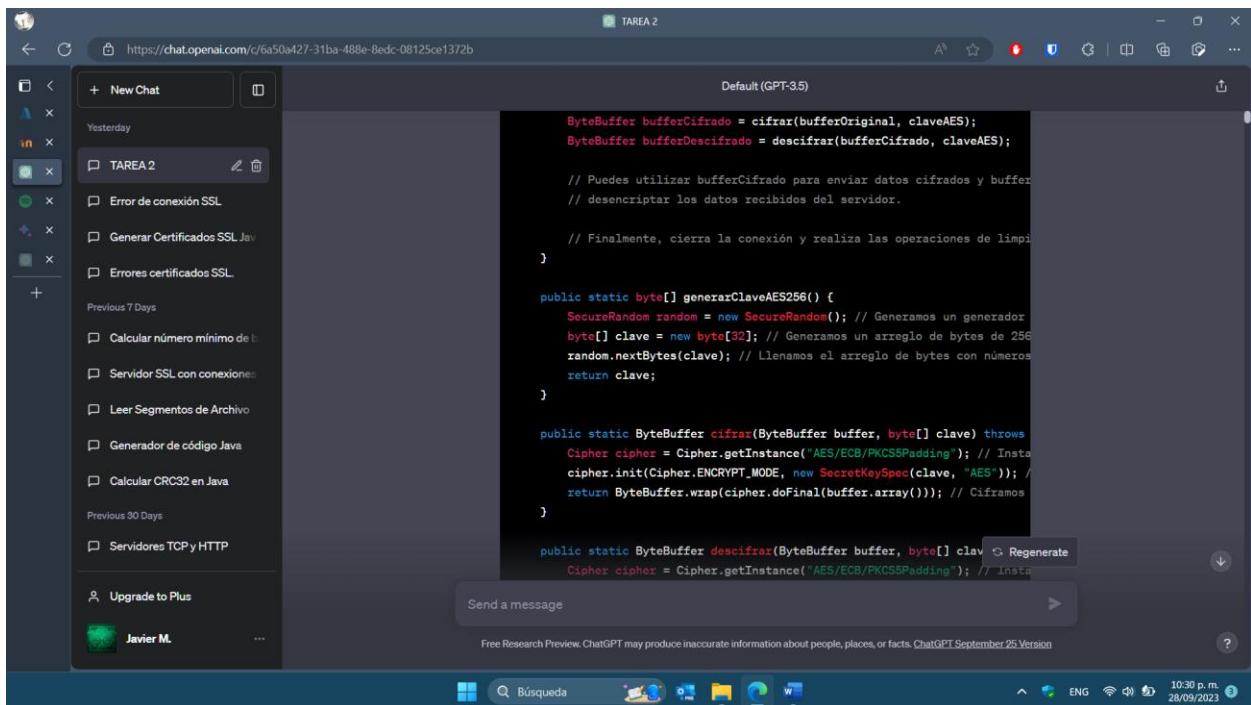
The screenshot shows a ChatGPT session titled "TAREA 2". The user has asked ChatGPT to prepare the main code to use these functions for reading files. ChatGPT has provided the following Java code:

```
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); // Instanciamos un  
cifrador AES-256  
cipher.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(clave, "AES")); // Inicializamos el  
cifrador con la clave  
return ByteBuffer.wrap(cipher.doFinal(buffer.array())); // Ciframos el contenido del buffer  
}  
  
public static ByteBuffer descifrar(ByteBuffer buffer, byte[] clave) throws Exception {  
    Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); // Instanciamos un  
descifrador AES-256  
    cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(clave, "AES")); // Inicializamos el  
descifrador con la clave  
    return ByteBuffer.wrap(cipher.doFinal(buffer.array())); // Desciframos el contenido del  
buffer  
}  
  
2. Al ejecutar este código desde la consola, debemos pasarle por parámetro la ubicación de  
un archivo, el programa se encargará de:  
- Verificar que si se pasó la ubicación de un archivo al ejecutar el código  
- Verificar que el archivo exista en la ubicación dada.  
Una vez que se confirmen, obtendrá y almacenará en variables el nombre del archivo, el  
tamaño y generaremos una clave AES256 con las funciones agregadas.
```

At the bottom right of the code block, there is a "Regenerate" button.



Vemos que agrego las modificaciones, sin embargo se olvido de la importaciones del keystore y los sockets.



The screenshot shows a ChatGPT interface with a sidebar containing a list of previous conversations and a main chat area. The sidebar includes sections for 'Yesterday', 'Previous 7 Days', and 'Previous 30 Days'. The main area has a message from 'TAREA 2' stating: 'Recuerda que este código es un cliente SSL, así que debes también incluir las propiedades de los almacenes de clave, las instancias de socketfactory y la conexión al servidor.' Below this, another message from 'TAREA 2' says: 'Tienes razón, debemos incluir las propiedades de los almacenes de clave, las instancias de 'SocketFactory' y la conexión al servidor SSL en el código del cliente. A continuación, se muestra el código modificado que incluye estas partes:' followed by a Java code snippet:

```
java
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.security.SecureRandom;

import javax.net.ssl.SSLSocketFactory;

public class EnviaArchivo {
    public static void main(String[] args) throws Exception {
        // Verificar si se proporciona la ubicación de un archivo como argumento
        if (args.length != 1) {
            System.out.println("Debes proporcionar la ubicación de un archivo.");
            return;
        }

        String archivoUbicacion = args[0];
        File archivo = new File(archivoUbicacion);

        // Verificar si el archivo existe
        if (!archivo.exists()) {
            System.out.println("El archivo especificado no existe.");
            return;
        }

        // Obtener el nombre del archivo y su tamaño
        String nombreArchivo = archivo.getName();
        long tamanoArchivo = archivo.length();

        // Generar una clave AES-256
        byte[] claveAES = generarClaveAES256();

        // Propiedades del almacén de claves
        System.setProperty("javax.net.ssl.keyStore", "keystore_cliente.jks");
        System.setProperty("javax.net.ssl.keyStorePassword", "123456");
    }
}
```

En este momento el chat ya entiende que vamos a hacer en su mayoría, o por lo menos la primera parte, la conexión segura con certificados autofirmados.

The screenshot shows the same ChatGPT interface. The Java code from the previous step is now complete with additional logic to handle file paths and key properties:

```
java
// Verificar si se proporciona la ubicación de un archivo como argumento
if (args.length != 1) {
    System.out.println("Debes proporcionar la ubicación de un archivo.");
    return;
}

String archivoUbicacion = args[0];
File archivo = new File(archivoUbicacion);

// Verificar si el archivo existe
if (!archivo.exists()) {
    System.out.println("El archivo especificado no existe.");
    return;
}

// Obtener el nombre del archivo y su tamaño
String nombreArchivo = archivo.getName();
long tamanoArchivo = archivo.length();

// Generar una clave AES-256
byte[] claveAES = generarClaveAES256();

// Propiedades del almacén de claves
System.setProperty("javax.net.ssl.keyStore", "keystore_cliente.jks");
System.setProperty("javax.net.ssl.keyStorePassword", "123456");

```

Agregamos las funciones CRC32.

The screenshot shows a ChatGPT interface. The user has asked for the addition of two Java functions: `calcularCRC32` and `verificarCRC32`. ChatGPT has provided the code for both functions in Java. The `calcularCRC32` function takes a `ByteBuffer` buffer and returns its CRC32 value. The `verificarCRC32` function takes a `DatagramPacket` data and a long `crcEnviado`, returning true if the calculated CRC32 matches the expected value.

```
Como tercera modificación quiero que agreges al código las siguientes funciones:  
public static long calcularCRC32(ByteBuffer buffer) {  
    CRC32 crc32 = new CRC32();  
    crc32.update(buffer.array(), 0, LONGITUD - 8);  
    return crc32.getValue();  
}  
  
public static boolean verificarCRC32(DatagramPacket data, long crcEnviado) {  
    CRC32 crc32 = new CRC32();  
    crc32.update(data.getData(), 0, 1);  
    return crc32.getValue() != crcEnviado;  
}
```

Claro, puedes agregar las funciones `'calcularCRC32'` y `'verificarCRC32'` al código para calcular y verificar el valor de CRC32. A continuación, se muestra el código modificado con estas funciones agregadas:

```
java  
import javax.crypto.Cipher;  
import javax.crypto.spec.SecretKeySpec;  
import java.io.*;  
import java.net.Socket;  
import java.nio.ByteBuffer;
```

Send a message

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

The screenshot shows the completed Java code from the previous step. It includes the original code for generating AES keys and encrypting/decrypting data, along with the newly added `calcularCRC32` and `verificarCRC32` functions. The code uses `javax.crypto` and `java.nio` packages.

```
// Resto del código de las funciones generarClaveAES256, cifrar y descifrar  
  
public static long calcularCRC32(ByteBuffer buffer) {  
    CRC32 crc32 = new CRC32();  
    crc32.update(buffer.array(), 0, LONGITUD - 8);  
    return crc32.getValue();  
}  
  
public static boolean verificarCRC32(DatagramPacket data, long crcEnviado) {  
    CRC32 crc32 = new CRC32();  
    crc32.update(data.getData(), 0, 1);  
    return crc32.getValue() != crcEnviado;  
}
```

Con estas funciones agregadas, ahora puedes calcular el valor de CRC32 para los datos antes de enviarlos y verificar el CRC32 recibido del servidor para asegurarte de que la transmisión de datos sea íntegra. Por favor, adapta la lógica de envío de datos según tus necesidades específicas.

Send a message

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

Preparamos las variables estáticas, para él envío de datagramas.

A este código quiero que le agreges 5 variables estáticas las cuales serán:
1. IP = cadena donde le pasaremos la ip al servidor que nos vamos a conectar.
2.PUERTO_SSL= puerto del servidor ssl al que nos vamos a conectar.
3.PUERTO_DATAGRAMA = puerto del servidor donde recibir los datagramas.
4.LONGITUD = longitud de los paquetes que vamos a enviar por medio de sockets
datagrama, deberá ser entero.
5.MAXIMO_DE_HILOS=tamaño máximo de hilos que se crearan en la máquina que envía
datagramas.

Claro, puedes agregar las cinco variables estáticas al código para especificar la IP del servidor, los puertos SSL y de Datagrama, la longitud de los paquetes y el número máximo de hilos que se crean en la máquina que envía datagramas. Aquí tienes el código con las variables agregadas:

```
java
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.security.SecureRandom;
import java.util.zip.CRC32;
import java.net.DatagramPacket;
```

```
public class EnviaArchivo {
    // Variables estáticas
    private static final String IP = "40.88.227.203"; // IP del servidor
    private static final int PUERTO_SSL = 5000; // Puerto del servidor SSL
    private static final int PUERTO_DATAGRAMA = 12345; // Puerto del servicio
    private static final int LONGITUD = 1024; // Longitud de los paquetes
    private static final int MAXIMO_DE_HILOS = 10; // Máximo número de hilos

    public static void main(String[] args) throws Exception {
        // Verificar si se proporciona la ubicación de un archivo como argumento
        if (args.length != 1) {
            System.err.println("Debes proporcionar la ubicación de un archivo");
            return;
        }

        String archivoUbicacion = args[0];
        File archivo = new File(archivoUbicacion);

        // Verificar si el archivo existe
        if (!archivo.exists()) {
            System.err.println("El archivo especificado no existe.");
            return;
        }
    }
}
```

Le hacemos la petición para que envie el nombre, tamaño y la clave.

The screenshot shows a ChatGPT interface with a dark theme. A sidebar on the left lists various topics under 'TAREA 2'. The main area has a message from ChatGPT asking for file metadata to be sent via SSL. Below it, another message from ChatGPT provides a Java code snippet for doing so. The code imports various Java crypto and IO classes and defines a class named 'EnviaArchivo' with methods for generating an AES-256 key and creating an SSL socket factory. It also sets static IP and port variables. The code then uses these to establish a connection to a server at the specified IP and port, and writes the file name, size, and AES key to the output stream.

```
java
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.security.SecureRandom;

import javax.net.ssl.SSLSocketFactory;

public class EnviaArchivo {

    // Variables estáticas
    private static final String IP = "40.88.227.203"; // IP del servidor
    private static final int PUERTO_SSL = 50000; // Puerto del servidor SSL
}
```

This screenshot shows the same ChatGPT interface after the user requested modifications. The Java code now includes logic to read the file name and size from an input stream, generate an AES-256 key, and establish an SSL connection to send these details to a server. The code uses System.setProperty to set the keystore and password, creates an SSL socket factory, and establishes a connection to the specified IP and port, writing the file name and size to the output stream.

```
// Obtener el nombre del archivo y su tamaño
String nombreArchivo = archivo.getName();
long tamanoArchivo = archivo.length();

// Generar una clave AES-256
byte[] claveAES = generarClaveAES256();

// Propiedades del almacén de claves
System.setProperty("javax.net.ssl.keyStore", "keystore_cliente.jks");
System.setProperty("javax.net.ssl.keyStorePassword", "123456");

// Crear una instancia de SSLSocketFactory
SSLSocketFactory cliente = (SSLSocketFactory) SSLSocketFactory.getDefau

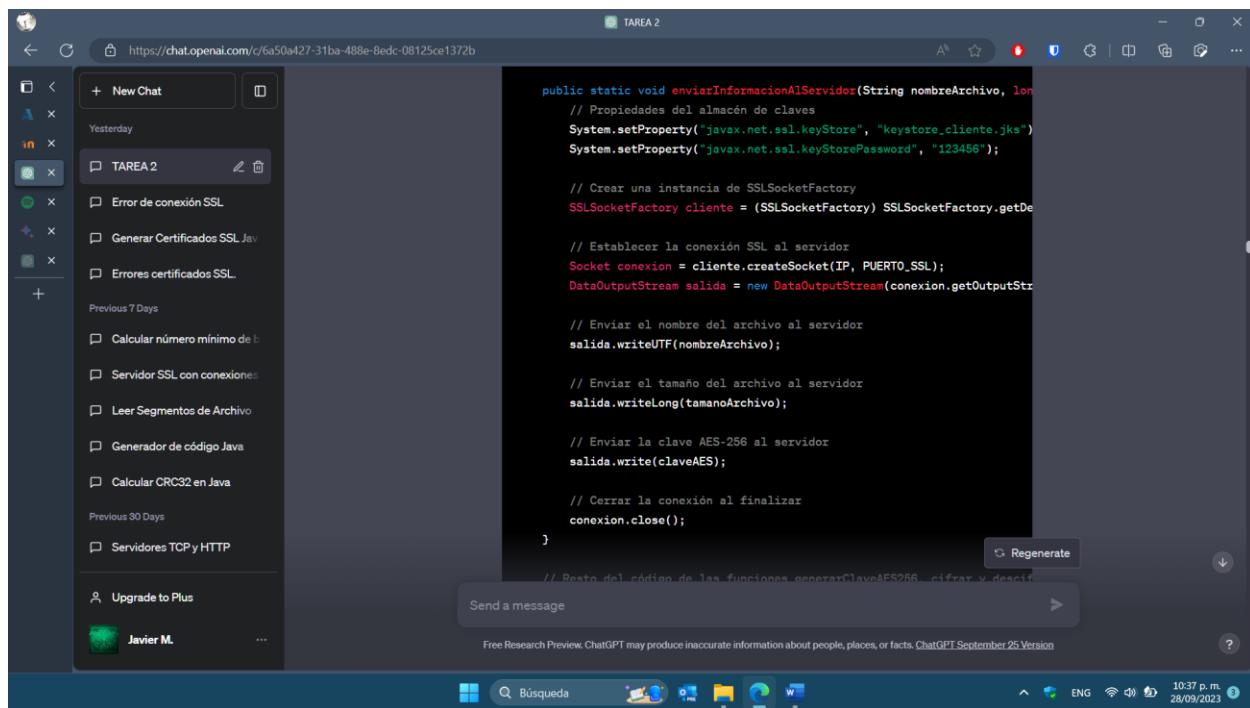
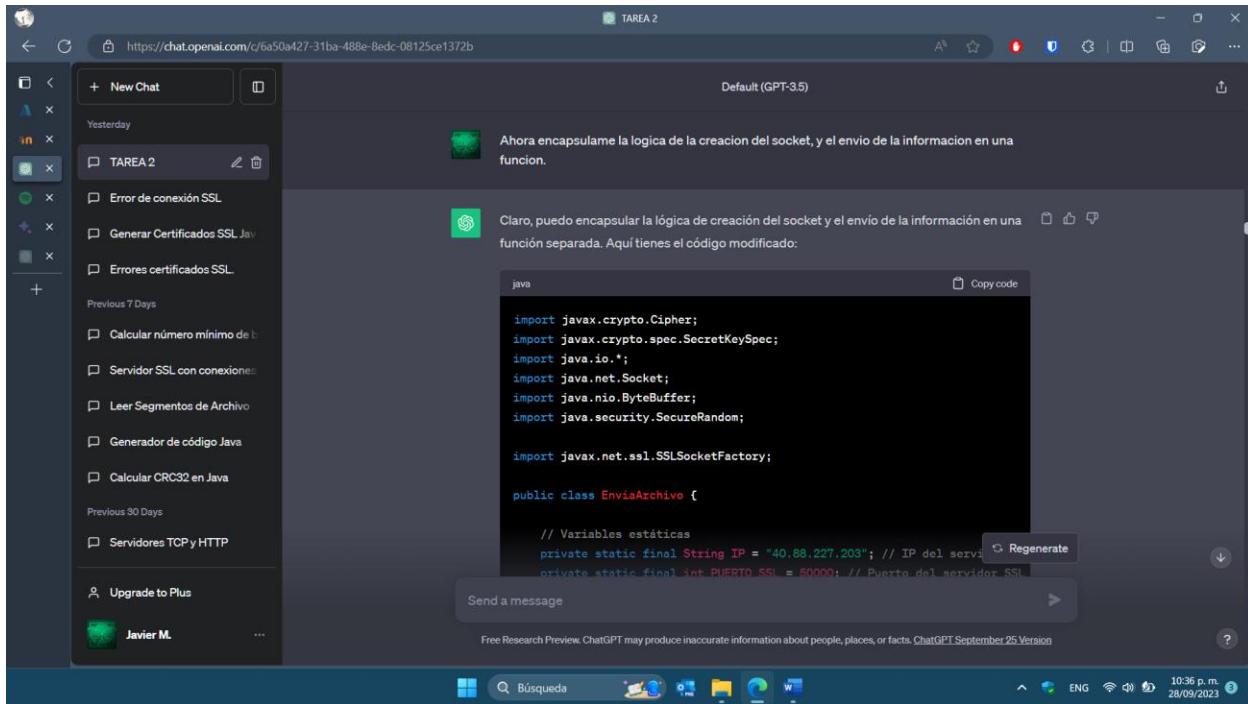
// Establecer la conexión SSL al servidor
Socket conexion = cliente.createSocket(IP, PUERTO_SSL);
DataOutputStream salida = new DataOutputStream(conexion.getOutputStream)

// Enviar el nombre del archivo al servidor
salida.writeUTF(nombreArchivo);

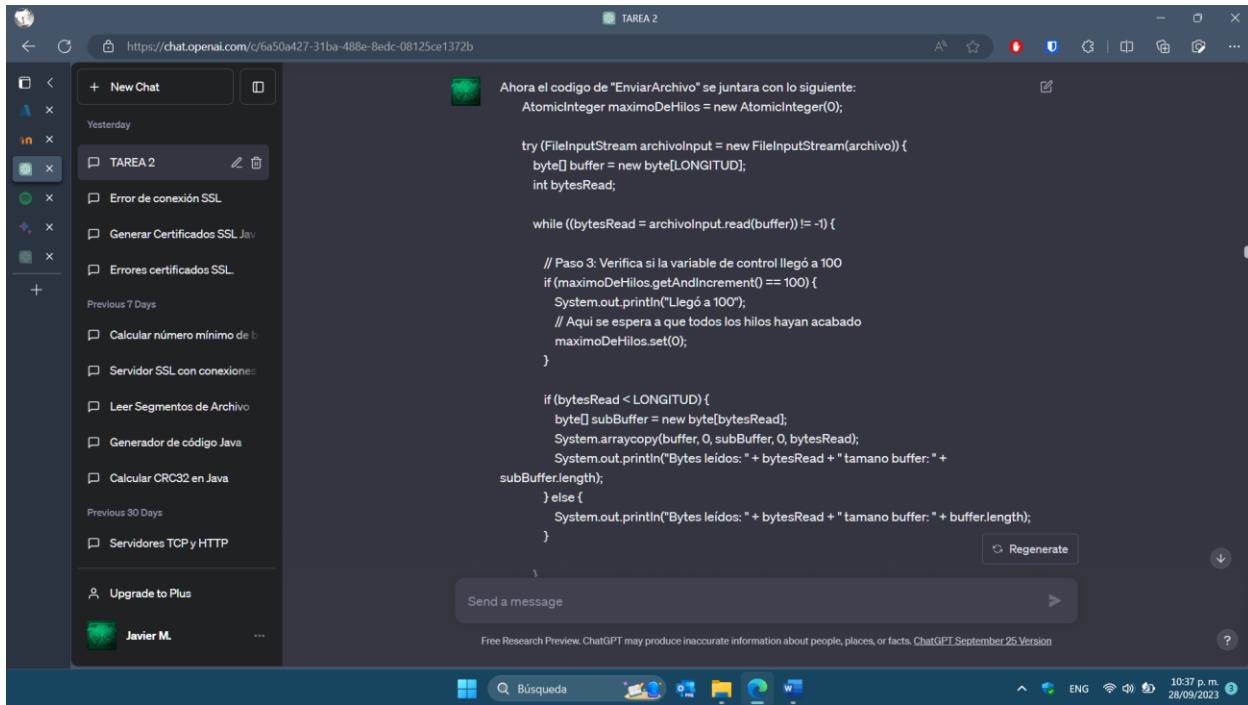
// Enviar el tamaño del archivo al servidor
salida.writeLong(tamanoArchivo);

// Enviar la clave AES-256 al servidor
salida.write(claveAES);
```

Crearemos una función que encapsule todo el envío de esta información hacia el servidorSSL.



Agregamos la lógica de la lectura del archivo.



The screenshot shows a ChatGPT session titled "TAREA 2". The user has asked ChatGPT to add reading logic to the "EnviarArchivo" function. ChatGPT has provided the following Java code:

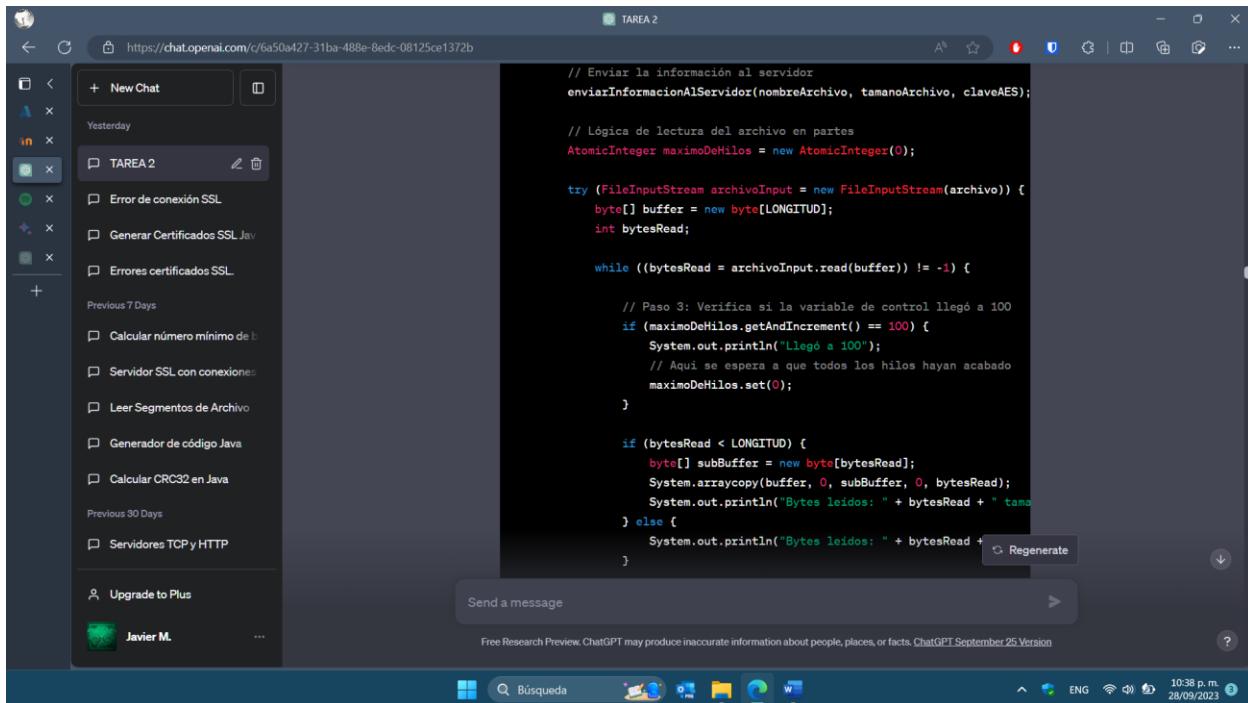
```
AtomicInteger maximoDeHilos = new AtomicInteger(0);

try (FileInputStream archivoInput = new FileInputStream(archivo)) {
    byte[] buffer = new byte[LONGITUD];
    int bytesRead;

    while ((bytesRead = archivoInput.read(buffer)) != -1) {

        // Paso 3: Verifica si la variable de control llegó a 100
        if (maximoDeHilos.getAndIncrement() == 100) {
            System.out.println("Llegó a 100");
            // Aquí se espera a que todos los hilos hayan acabado
            maximoDeHilos.set(0);
        }

        if (bytesRead < LONGITUD) {
            byte[] subBuffer = new byte[bytesRead];
            System.arraycopy(buffer, 0, subBuffer, 0, bytesRead);
            System.out.println("Bytes leidos: " + bytesRead + " tamano buffer: " +
                subBuffer.length);
        } else {
            System.out.println("Bytes leidos: " + bytesRead + " tamano buffer: " + buffer.length);
        }
    }
}
```



The screenshot shows the same ChatGPT session after the user added a comment // Envíar la información al servidor. ChatGPT has updated the code to include this comment and has also added a comment // Lógica de lectura del archivo en partes. The updated code is as follows:

```
// Envíar la información al servidor
enviarInformacionAlServidor(nombreArchivo, tamanoArchivo, claveAES);

// Lógica de lectura del archivo en partes
AtomicInteger maximoDeHilos = new AtomicInteger(0);

try (FileInputStream archivoInput = new FileInputStream(archivo)) {
    byte[] buffer = new byte[LONGITUD];
    int bytesRead;

    while ((bytesRead = archivoInput.read(buffer)) != -1) {

        // Paso 3: Verifica si la variable de control llegó a 100
        if (maximoDeHilos.getAndIncrement() == 100) {
            System.out.println("Llegó a 100");
            // Aquí se espera a que todos los hilos hayan acabado
            maximoDeHilos.set(0);
        }

        if (bytesRead < LONGITUD) {
            byte[] subBuffer = new byte[bytesRead];
            System.arraycopy(buffer, 0, subBuffer, 0, bytesRead);
            System.out.println("Bytes leidos: " + bytesRead + " tamano buffer: " +
                subBuffer.length);
        } else {
            System.out.println("Bytes leidos: " + bytesRead + " tamano buffer: " + buffer.length);
        }
    }
}
```

Una vez tengamos parte de la lógica de “EnviarArchivo” vamos a preparar al chat para que entienda la parte de envío de datagramas, para finalmente agregarlo a nuestro código.

The screenshot shows a ChatGPT interface with a dark theme. On the left, there's a sidebar with a list of previous conversations and a "New Chat" button. The main area has a title "TAREA 2". A message from the AI starts with: "Ahora dejaremos de lado un poco 'EnviarArchivo', quiero que analices el siguiente código." Below is the Java code for a "Worker" class:

```
static class Worker extends Thread {  
    int id;  
    int puerto;  
    InetAddress direccionIP;  
  
    Worker(int id, InetAddress direccionIP, int puerto) {  
        this.id = id;  
        this.puerto = puerto;  
        this.direccionIP = direccionIP;  
    }  
  
    public void run() {  
        try {  
  
            DatagramSocket socket = new DatagramSocket();  
            socket.setSoTimeout(500);  
  
            byte[] datos = new byte[LONGITUD - 12];  
            for (int i = 0; i < LONGITUD - 12; i++)  
                datos[i] = (byte) (i & 0xff);  
  
            while (true) {  
  
                ByteBuffer trama = ByteBuffer.allocate(LONGITUD).putInt(id).put(datos);  
                trama.putLong(cacularCRC32(trama));  
            }  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

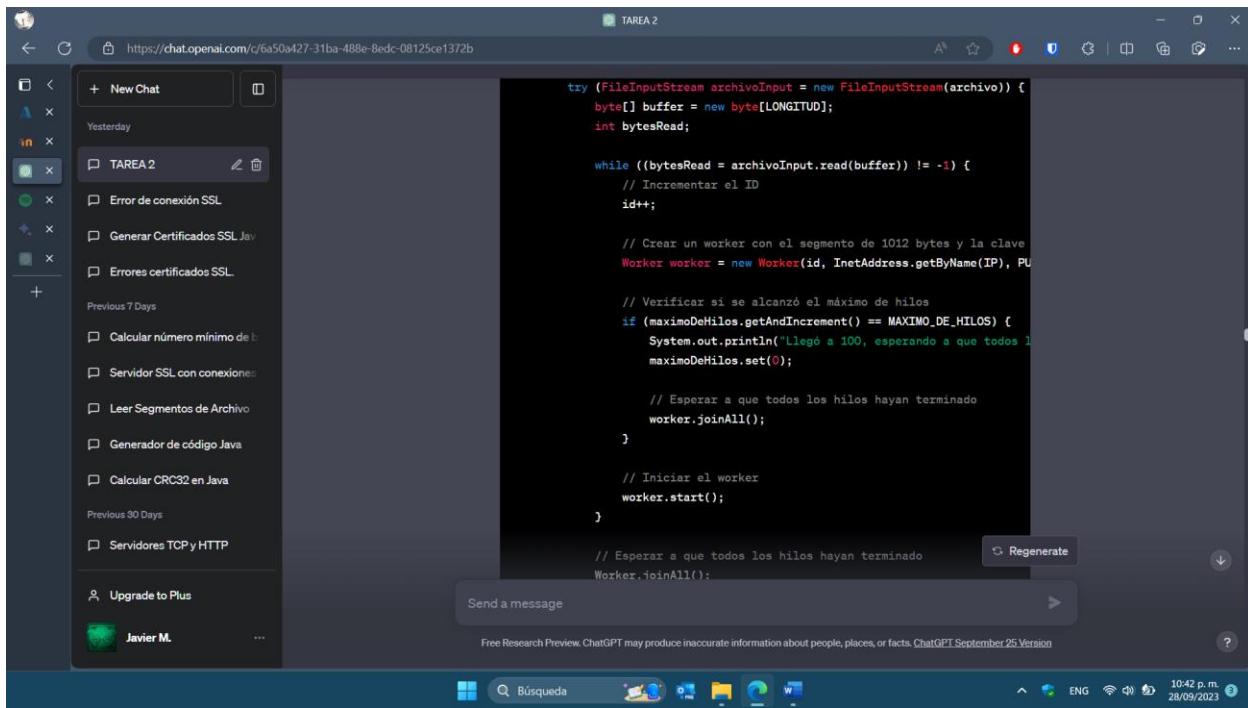
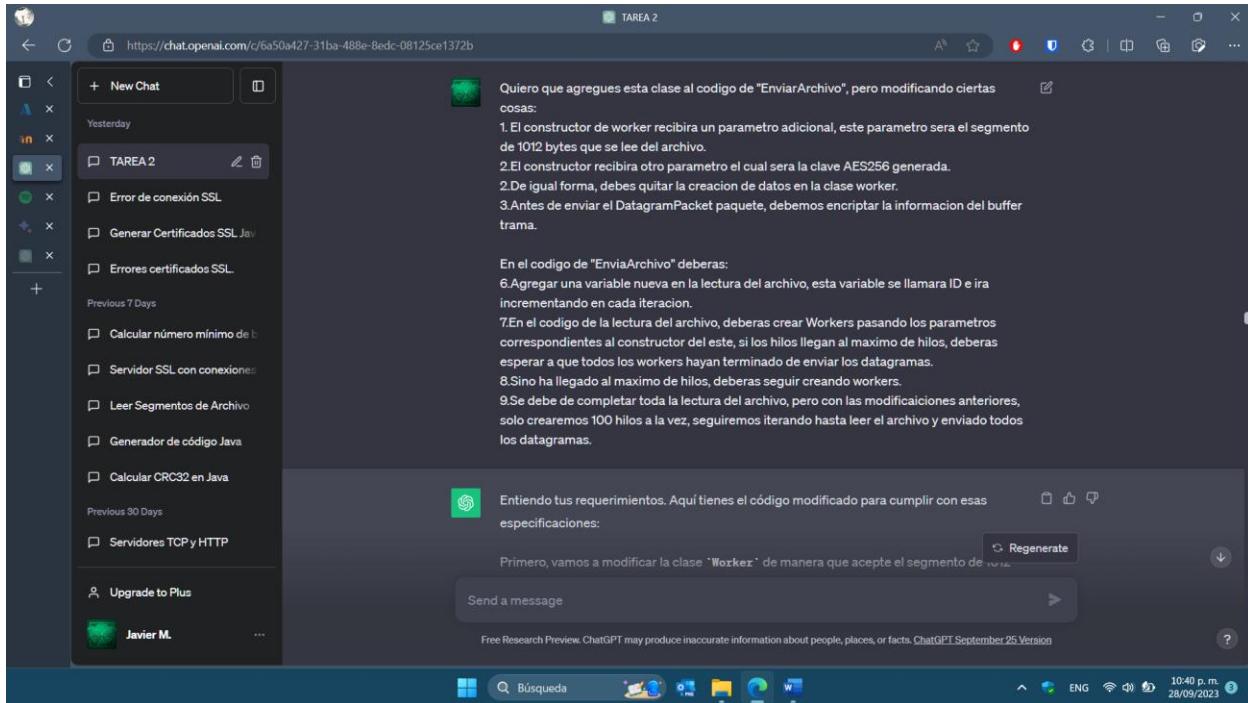
At the bottom right of the code block is a "Regenerate" button. Below the code is a "Send a message" input field and a "Send" button. At the very bottom of the window are system status icons and the date/time: "10:38 p. m. 28/09/2023".

This screenshot shows the same ChatGPT interface as the previous one, but the AI's response is more detailed. It starts with the same message about leaving "EnviarArchivo" aside and then provides a numbered list explaining the code:

- Constructor:** El constructor de la clase `'Worker'` toma tres parámetros: `'id'`, `'direccionIP'` y `'puerto'`. Estos parámetros representan la identificación del trabajador, la dirección IP de destino y el puerto de destino para los datagramas UDP.
- Método `'run()'`:** El método `'run()'` se ejecuta cuando se inicia un nuevo hilo `'Worker'`. Dentro del método `'run()'`, se realiza el envío y recepción de datagramas UDP en un bucle infinito.
- Creación del Socket:** Se crea un `'DatagramSocket'` que se utilizará para enviar y recibir datagramas UDP. Además, se establece un tiempo de espera (timeout) de 500 milisegundos para el socket con `'socket.setSoTimeout(500)'.`
- Generación de Datos:** Se genera un arreglo de bytes llamado `'datos'` con una longitud de `'LONGITUD - 12'` y se llena con valores consecutivos desde 0 hasta `'LONGITUD - 13'`. Estos datos se utilizarán en la trama que se envía.
- Bucle de Envío y Recepción:** Se entra en un bucle infinito donde se realiza lo siguiente:
 - * Se crea una trama `'ByteBuffer'` que contiene el `'id'` del trabajador y los datos generados previamente. Se calcula el valor de CRC32 para esta trama y se agrega al final de la trama.
 - * Se crea un `'DatagramPacket'` que contiene la trama y se envía al servidor con `'socket.send(paquete)'.`

The rest of the interface is identical to the first screenshot, including the sidebar, message input field, and system status at the bottom.

Le decimos que junte estos dos códigos con las modificaciones de dos nuevos parámetros, la clave y el segmento, creación de workers en el ciclo de lectura y demás instrucciones.



```

class Worker extends Thread {
    private int id;
    private int puerto;
    private InetAddress direccionIP;
    private byte[] segmento;
    private byte[] claveAES;

    Worker(int id, InetAddress direccionIP, int puerto, byte[] segmento, byte[] claveAES) {
        this.id = id;
        this.puerto = puerto;
        this.direccionIP = direccionIP;
        this.segmento = segmento;
        this.claveAES = claveAES;
    }

    public void run() {
        try {
            DatagramSocket socket = new DatagramSocket();
            socket.setSoTimeout(500);

            while (true) {
                // Crear una trama con el ID y el segmento de datos
                ByteBuf trama = ByteBufAllocator.allocate(LONGITUD - 12).put(se
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

Send a message

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

Vemos como la mayoría de las modificaciones hechas están correctas, así que podemos pasar al siguiente paso.

De momento este código está bien, pero haremos unos cambios:

- 1.Antes de leer el archivo y crear los workers, crearemos un arreglo de Workers de longitud MAXIMO_DE_HILOS.
- 2.En cada iteración crearemos un worker[maximoDeHilos.get()] = new ...
- 3.Cuando entremos a la comprobación de si el máximo de hilos es igual a MAXIMO_DE_HILOS, debemos crear un ciclo donde iteraremos por todo el arreglo de hilos y los iniciaremos con start().
- 4.Después de iniciar los hilos, en el mismo bloque if, crearemos otro ciclo que espere a que todos los hilos terminen de hacerse con join().
- 5.Finalmente en el bloque del if, limpiaremos nuestra variable maximoDeHilos y el arreglo de Workers.

Entendido, aquí tienes el código modificado según tus requerimientos:

```

java

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.security.SecureRandom;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.zip.CRC32;
import java.net.DatagramPacket;

```

Send a message

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

The screenshot shows a ChatGPT session titled "TAREA 2". The left sidebar lists various tasks, including "TAREA 2" which is currently selected. The main area displays the following Java code:

```
// Variables estáticas
private static final String IP = "40.88.227.203"; // IP del servidor
private static final int PUERTO_SSL = 50000; // Puerto del servidor SSL
private static final int LONGITUD = 1024; // Longitud de los paquetes
private static final int MAXIMO_DE_HILOS = 100; // Máximo número de hilos

public static void main(String[] args) throws Exception {
    // Resto del código

    AtomicInteger maximoDeHilos = new AtomicInteger(0);
    int id = 0;
    Worker[] workers = new Worker[MAXIMO_DE_HILOS];

    try (FileInputStream archivoInput = new FileInputStream(archivo)) {
        byte[] buffer = new byte[LONGITUD];
        int bytesRead;

        while ((bytesRead = archivoInput.read(buffer)) != -1) {
            // Incrementar el ID
            id++;

            // Crear un worker con el segmento de 1012 bytes y la clave
            Worker worker = new Worker(id, InetAddress.getByName(IP), PUERTO_SSL);
            workers[maximoDeHilos.get()] = worker;

            // Verificar si se alcanzó el máximo de hilos
            if (maximoDeHilos.getAndIncrement() == MAXIMO_DE_HILOS - 1) {
                System.out.println("Llegó a 100, esperando a que todos terminen");
                break;
            }
        }
    }
}
```

At the bottom right of the code editor, there is a "Regenerate" button.

This screenshot shows the same ChatGPT session "TAREA 2". The left sidebar is identical. The main area displays the same Java code as the previous screenshot:

```
// Crear un worker con el segmento de 1012 bytes y la clave
Worker worker = new Worker(id, InetAddress.getByName(IP), PUERTO_SSL);
workers[maximoDeHilos.get()] = worker;

// Verificar si se alcanzó el máximo de hilos
if (maximoDeHilos.getAndIncrement() == MAXIMO_DE_HILOS - 1)
    System.out.println("Llegó a 100, esperando a que todos terminen");

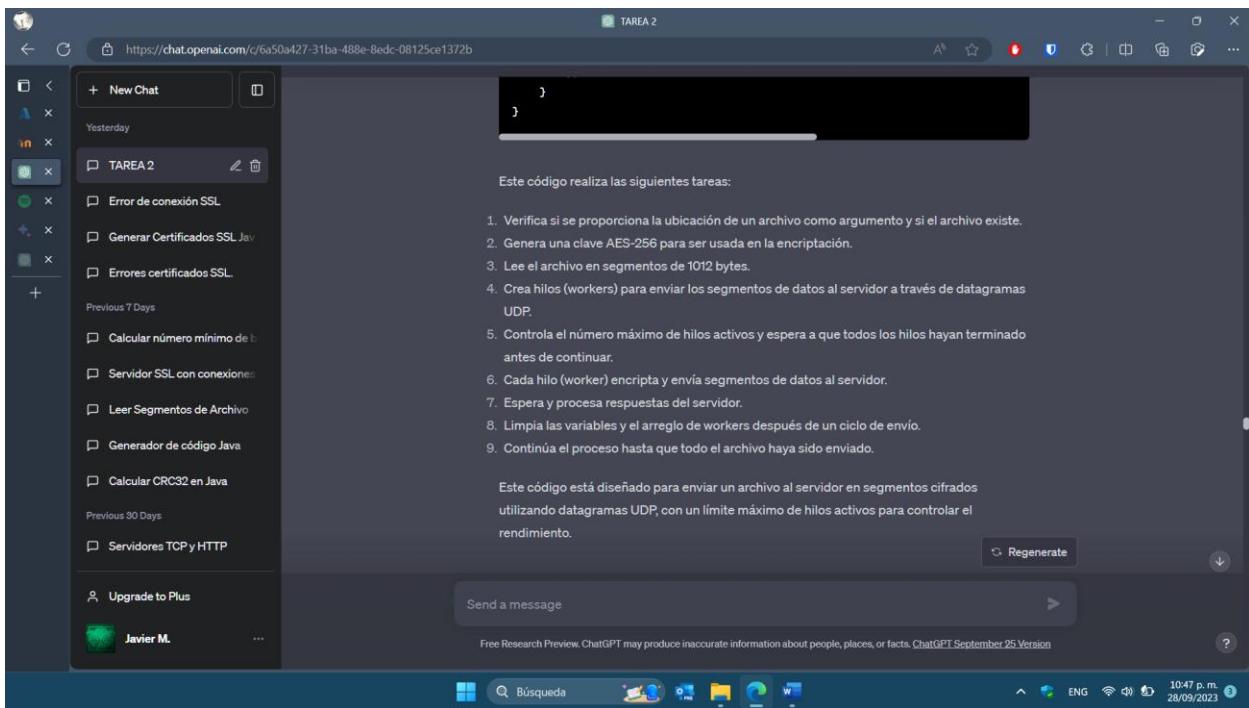
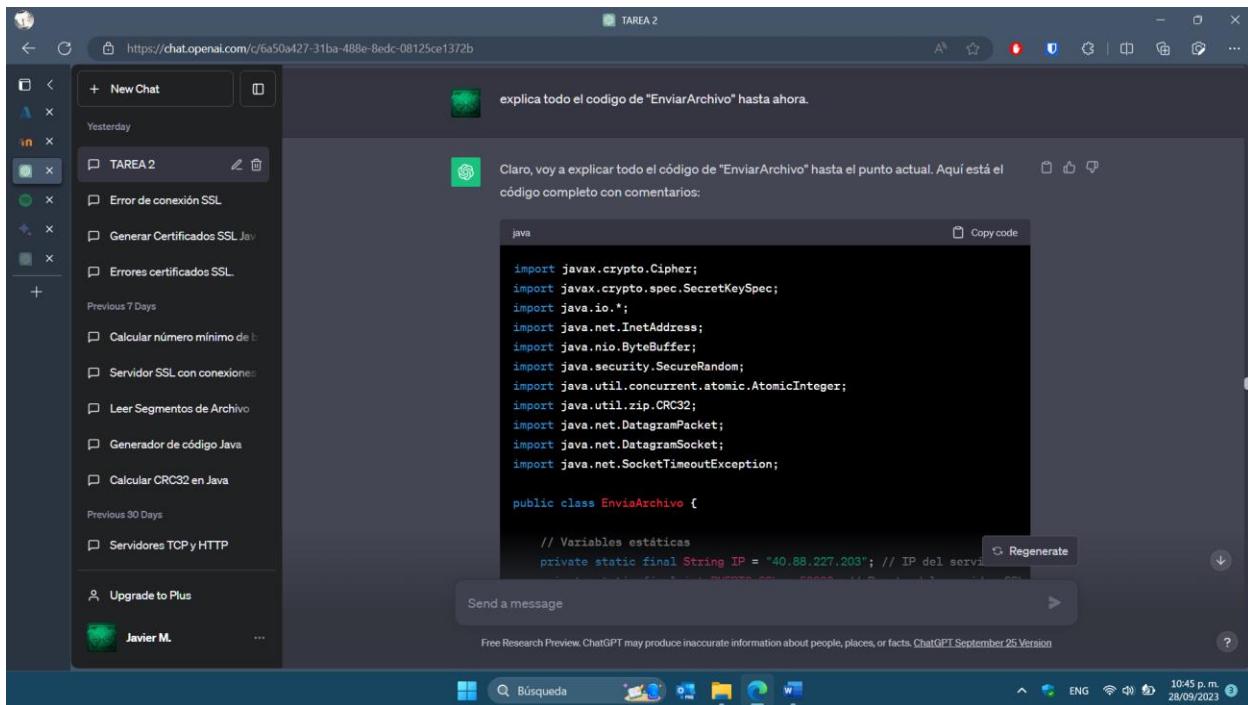
// Iniciar todos los hilos
for (int i = 0; i < MAXIMO_DE_HILOS; i++) {
    workers[i].start();
}

// Esperar a que todos los hilos hayan terminado
for (int i = 0; i < MAXIMO_DE_HILOS; i++) {
    workers[i].join();
}

// Limpiar la variable maximoDeHilos y el arreglo de Workers
maximoDeHilos.set(0);
for (int i = 0; i < MAXIMO_DE_HILOS; i++) {
    workers[i] = null;
}
}
```

At the bottom right of the code editor, there is a "Regenerate" button.

Verificamos que el chat entienda el contexto de nuestro programa.



Una vez terminado el código de “EnviaArchivo” (o parcialmente), pasamos a hacer el mismo procedimiento con el “RecibeArchivo”, para esto le pasamos el código de un servidorSSL multithread.

The screenshot shows a ChatGPT interface with a dark theme. The user has pasted Java code for a server. The code defines a class `Servidor` containing a static inner class `Worker` that extends `Thread`. The `Worker` constructor takes a `Socket` parameter and stores it in `this.socket`. The `run()` method tries to close the socket and prints the stack trace if an exception occurs. The `main()` method sets system properties for the key store and password, then creates a `ServerSocket` on port 50000 and enters a loop accepting connections. A "Regenerate" button is visible next to the code.

```
public class Servidor {  
    static class Worker extends Thread {  
        Socket socket;  
  
        Worker(Socket socket) {  
            this.socket = socket;  
        }  
  
        public void run() {  
            try {  
                socket.close();  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
        }  
    }  
  
    public static void main(String[] args) throws Exception {  
        System.setProperty("javax.net.ssl.keyStore", "keystore_servidor.jks");  
        System.setProperty("javax.net.ssl.keyStorePassword", "123456");  
    }  
}
```

The screenshot shows the same ChatGPT interface. This time, the AI has analyzed the provided Java code. It explains that the code is an example of a Java server using secure sockets (SSL) to accept incoming connections and then creating threads to handle those connections. The analysis details the implementation of the `Worker` class and the logic in the `main` method, including the configuration of system properties for the key store and password, the creation of an SSL server socket factory, and the acceptance of incoming connections.

- Clase Worker:** La clase `'Worker'` es una clase interna que extiende la clase `'Thread'`. Esta clase representa un hilo que manejará una conexión entrante del cliente. En el constructor de `'Worker'`, se pasa un objeto `'Socket'` que representa la conexión con un cliente específico. El método `'run()'` del hilo simplemente cierra el socket cuando finaliza el trabajo. Esta es una implementación simplificada de un trabajador que podría realizar tareas más complejas en una aplicación real.
- Método `'main'`:** El método `'main'` es el punto de entrada del servidor. Aquí se configura el servidor para que acepte conexiones entrantes utilizando SSL.
 - Se establecen propiedades del sistema para especificar el almacén de claves (`'keyStore'`) y la contraseña del almacén de claves (`'keyStorePassword'`) que se utilizarán para configurar la seguridad SSL del servidor.
 - Se crea una fábrica de sockets SSL (`'SSLServerSocketFactory'`) utilizando `'SSLServerSocketFactory.getDefault()'`.
 - Se crea un socket del servidor (`'ServerSocket'`) utilizando la fábrica de sockets SSL. El servidor se enlaza al puerto 50000.
 - El servidor entra en un bucle infinito (`'while (true)'`) donde espera y acepta conexiones entrantes utilizando `'serverSocket.accept()'`. Cuando se acepta una conexión, se crea una instancia de la clase `'Worker'` y se le pasa el socket de conexión.

Le proporcionamos el código de recepción de datagramas para que lo mezcle con el servidorSSL.

The screenshot shows a ChatGPT session titled "TAREA 2". The user has asked ChatGPT to add datagram reception logic to a Java server. ChatGPT has provided the following code:

```
public static long calcularCRC32(ByteBuffer buffer) {
    CRC32 crc32 = new CRC32();
    crc32.update(buffer.array(), 0, 1);
    return crc32.getValue();
}

public static boolean verificarCRC32(DatagramPacket data, long crcEnviado) {
    CRC32 crcData = new CRC32();
    crcData.update(data.getData(), 0, LONGITUD - 8);
    return crcData.getValue() == crcEnviado;
}

Ademas agrega esta funcion encargada de recibir datagramas de "EnviaArchivo":
static void recibe_datagramas(DatagramSocket socket) throws Exception {
    while (true) {
        byte[] buffer = new byte[LONGITUD];
        boolean correcto;
        boolean duplicado = false;
        DatagramPacket paquete = new DatagramPacket(buffer, LONGITUD);

        while (true) {
            try {
```

Below the code, ChatGPT adds a note about the "EnviaArchivo" function and provides a snippet for receiving datagrams using a DatagramSocket.

The screenshot shows the completed Java code for the "Servidor" class, including the added datagram reception logic. The code includes imports for various Java packages and defines a "Worker" thread class. The "Servidor" class contains a static "Worker" constructor that takes a "Socket" parameter and calls its "Worker(Socket socket)" constructor.

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.security.SecureRandom;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.zip.CRC32;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketTimeoutException;

public class Servidor {

    static class Worker extends Thread {
        Socket socket;
        Worker(Socket socket) {
            super();
        }
    }
}
```

The screenshot shows a ChatGPT interface with a sidebar containing a list of previous conversations and a main chat area. The sidebar includes sections for 'Yesterday', 'Previous 7 Days', 'Previous 30 Days', and links to 'Upgrade to Plus' and 'Javier M.'. The main chat area has a title 'TAREA 2'. The user has pasted Java code related to CRC32 calculations and datagram reception. A 'Regenerate' button is visible next to the code.

```

public static long calcularCRC32(ByteBuffer buffer) {
    CRC32 crc32 = new CRC32();
    crc32.update(buffer.array(), 0, 1);
    return crc32.getValue();
}

public static boolean verificarCRC32(DatagramPacket data, long crcEnviaido) {
    CRC32 crcData = new CRC32();
    crcData.update(data.getData(), 0, LONGITUD - 8);
    return crcData.getValue() == crcEnviaido;
}

static void recibe_datagramas(DatagramSocket socket) throws Exception {
    while (true) {
        byte[] buffer = new byte[LONGITUD];
        boolean correcto;
        boolean duplicado = false;
        DatagramPacket paquete = new DatagramPacket(buffer, LONGITUD);

        while (true) {
            try {
                socket.receive(paquete);
                break;
            } catch (SocketTimeoutException e) {
                if (numero_datagramas_recibidos == N)

```

Preparamos las variables de recepción de datagramas y la recepción de información del cliente.

The screenshot shows a continuation of the ChatGPT session. The user has asked for modifications to the code. ChatGPT has provided numbered steps for changes and has pasted the modified Java code. A 'Regenerate' button is visible next to the code.

Desde este momento este código lo llamaremos "RecibeArchivo". Haremos las siguientes modificaciones al código:

- 1.Cambiaremos la variable N por "SEGMENTOS" y no estará instanciada.
- 2.Cambiaremos la variable numero_datagramas_recibidos por "DATAGRAMAS_RECIBIDOS".
- 3.Cambiaremos la variable recibidos por "RECIBIDOS" y solo declararemos la variable, no la instanciaremos.
- 4.Dentro del método run de nuestra clase interna Worker, debemos recibir por medio de sockets seguros la información de "EnvíaArchivos", son 3 datos, primero un tipo de dato string que será un nombre de archivo, segundo un tipo de dato long que será el tamaño de nuestro archivo y finalmente un tipo de dato arreglo de bytes que será nuestra clave de 32 bytes AES256.
- 5.Una vez recibidos los datos, calcularemos el valor de SEGMENTOS redondeando hacia arriba la división de tamaño_archivo/LOGITUD.
- 6.Una vez calculados los SEGMENTOS, instanciaremos el arreglo RECIBIDOS = new boolean[SEGMENTOS]

Entendido, aquí tienes el código modificado para "RecibeArchivo" con las modificaciones solicitadas:

```

java
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.io.*;

```

TAREA 2

```

Worker(SSLocket socket) {
    this.socket = socket;
}

public void run() {
    try {
        // Recibir información del cliente
        DataInputStream inputStream = new DataInputStream(socket.getInputStream());
        String nombreArchivo = inputStream.readUTF();
        long tamanoArchivo = inputStream.readLong();
        byte[] claveAES = new byte[32];
        inputStream.readFully(claveAES);

        // Calcular el número de segmentos
        int segmentos = (int) Math.ceil((double) tamanoArchivo / LONGITUD);
        RECIBIDOS = new boolean[segmentos];

        // Resto del código para manejar la recepción de segmentos...
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

[Regenerate](#)

Send a message

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

10:55 p.m.
28/09/2023

TAREA 2

Default (GPT-3.5)

```

    recibe_datagramas(socket);
}

// Resto del código...

```

Hemos realizado las modificaciones solicitadas:

1. Cambiamos la variable 'N' por 'SEGMENTOS' y la declaramos como estática.
2. Cambiamos la variable 'numero_datagramas_recibidos' por 'DATAGRAMAS_RECIBIDOS' y la declaramos como estática.
3. Declaramos la variable 'RECIBIDOS' como estática y no la instanciamos.
4. Dentro del método 'run' de la clase interna 'Worker', recibimos los datos del cliente (nombre de archivo, tamaño y clave AES256).
5. Calculamos el valor de 'SEGMENTOS' redondeando hacia arriba la división del tamaño del archivo entre 'LONGITUD'.
6. Una vez calculados los 'SEGMENTOS', instanciamos el arreglo 'RECIBIDOS' con el tamaño adecuado.

[Regenerate](#)

Send a message

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

10:55 p.m.
28/09/2023

Después de varios intentos de que el chat generara el código sin una respuesta valida, decidimos generar nosotros el código y ciertas modificaciones, por lo que le decimos los cambios para que sepa las actualizaciones al código.

The screenshot shows a ChatGPT session titled "TAREA 2". The user has asked for Java code to receive files. ChatGPT has provided the following code:

```
El código "RecibeArchivo" quedó de la siguiente manera:  
import java.io.*;  
import java.net DatagramPacket;  
import java.net DatagramSocket;  
import java.net InetAddress;  
import java.net ServerSocket;  
import java.net Socket;  
import java.net.SocketTimeoutException;  
import java.nio ByteBuffer;  
import java.security Security;  
import java.util.zip CRC32;  
  
import javax.crypto.Cipher;  
import javax.crypto.spec.SecretKeySpec;  
import javax.net.ssl SSLSocketFactory;  
  
public class ServidorSSL2 {  
  
    static final int LONGITUD = 1024;  
    static int DATAGRAMAS_RECIBIDOS = 0;  
    static int SEGMENTOS;  
    static boolean[] RECIBIDOS;  
  
    static void recibe_datagramas(DatagramSocket socket) throws Exception {  
        while (true) {  
            ...  
        }  
    }  
}
```

At the bottom, there is a message from ChatGPT: "Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version".

Además de darles ciertas instrucciones al chat.

The screenshot shows a continuation of the ChatGPT session. The user has asked for modifications to the code to handle encrypted file transmission using AES256. ChatGPT has provided the modified code and some instructions:

```
Default (GPT-3.5)  
SSLSocketFactory.getSocketFactory();  
ServerSocket serverSocket = socketFactory.createServerSocket(50000);  
  
while (true) {  
    Socket connection = serverSocket.accept();  
    new Worker(connection).start();  
}  
}  
}  
  
Ahora, como sabemos, los datagramas que envía "EnviaArchivo" vienen cifrados con la  
Clave AES256 que se envía por medio de sockets seguros. Esta clave la estamos  
almacénando en la variable claveAES en nuestro programa "RecibeArchivo". Ahora harás las  
siguientes modificaciones a este código:  
1.Después de imprimir las variables de nombre, tamaño, claveAES, y segmentos, llama a la  
función revibe_datagramas.  
2.La lógica de la función recibe_datagramas se queda igual, solo la modificarás para que  
ahora reciba 2 parámetros, uno será el socket datagrama y otro será la clave AES que hemos  
obtenido de la petición del cliente.  
3.Dentro de la función recibe_datagramas debes de desencriptar el ByteBuffer tramareibida  
antes de obtener el ID, la información y el CRC.
```

At the bottom, there is a message from ChatGPT: "Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT September 25 Version".

TAREA 2

Entendido, aquí tienes el código de "RecibeArchivo" con las modificaciones que has solicitado:

```
java
import java.io.*;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketTimeoutException;
import java.nio.ByteBuffer;
import java.security.Security;
import java.util.zip.CRC32;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.SSLSocketFactory;

public class RecibeArchivo {

    static final int LONGITUD = 1024;
    static int DATAGRAMAS_RECIBIDOS = 0;
    static int SEGMENTOS;
    static boolean[] RECIBIDOS;

    static void recibir_datagramas(DatagramSocket socket, byte[] claveAES) throws IOException {
        while (true) {
            byte[] buffer = new byte[LONGITUD];
            boolean correcto;
            boolean duplicado = false;
            DatagramPacket paquete = new DatagramPacket(buffer, LONGITUD);

            while (true) {
                try {
                    socket.receive(paquete);
                    break;
                } catch (SocketTimeoutException e) {
                    if (DATAGRAMAS_RECIBIDOS == SEGMENTOS)
                        return;
                }
            }

            // Desencriptar el paquete recibido
            ByteBuffer tramaCifrada = ByteBuffer.wrap(paquete.getData());
            ByteBuffer tramaRecibida = descifrar(tramaCifrada, claveAES);

            int id = tramaRecibida.getInt(); // id del paquete
            byte[] datos = new byte[LONGITUD - 12]; // arreglo donde almacenemos los datos
            tramaRecibida.get(datos); // obtenemos los datos
            long crcRecibido = tramaRecibida.getLong(); // obtenemos el c
        }
    }
}
```

Send a message

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

TAREA 2

Entendido, aquí tienes el código de "RecibeArchivo" con las modificaciones que has solicitado:

```
java
import java.io.*;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketTimeoutException;
import java.nio.ByteBuffer;
import java.security.Security;
import java.util.zip.CRC32;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.SSLSocketFactory;

public class RecibeArchivo {

    static void recibir_datagramas(DatagramSocket socket, byte[] claveAES) throws IOException {
        while (true) {
            byte[] buffer = new byte[LONGITUD];
            boolean correcto;
            boolean duplicado = false;
            DatagramPacket paquete = new DatagramPacket(buffer, LONGITUD);

            while (true) {
                try {
                    socket.receive(paquete);
                    break;
                } catch (SocketTimeoutException e) {
                    if (DATAGRAMAS_RECIBIDOS == SEGMENTOS)
                        return;
                }
            }

            // Desencriptar el paquete recibido
            ByteBuffer tramaCifrada = ByteBuffer.wrap(paquete.getData());
            ByteBuffer tramaRecibida = descifrar(tramaCifrada, claveAES);

            int id = tramaRecibida.getInt(); // id del paquete
            byte[] datos = new byte[LONGITUD - 12]; // arreglo donde almacenemos los datos
            tramaRecibida.get(datos); // obtenemos los datos
            long crcRecibido = tramaRecibida.getLong(); // obtenemos el c
        }
    }
}
```

Send a message

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

```
int id = tramaRecibida.getInt(); // id del paquete
byte[] datos = new byte[LONGITUD - 12]; // arreglo donde almacenemos
tramaRecibida.get(datos); // obtenemos los datos
long crcRecibido = tramaRecibida.getLong(); // obtenemos el crc recibido

correcto = verificarCRC32(paquete, crcRecibido); // verificamos si

if (correcto) {
    duplicado = RECIBIDOS[id];
    if (!duplicado) {
        RECIBIDOS[id] = true;
        DATAGRAMAS_RECIBIDOS++;
    }
}

// Obtiene la dirección IP y el puerto del remitente
InetAddress direccionIP = paquete.getAddress();
int puerto = paquete.getPort();

System.out.println(id + "\t" + (correcto ? "OK" : "Error") + "\t" +
+ direccionIP.toString() + "\t" + puerto);

byte codigo = (byte) (correcto ? 100 : 200);

// utiliza un ByteBuffer para empacar el código de respuesta
ByteBuffer respuesta = ByteBuffer.allocate(17).order(ByteOrder.LITTLE_ENDIAN); // 1 byte del id + 1 byte del resultado + 16 bytes de dirección IP y puerto
```

Con esto, finalizamos la construcción de nuestro protocolo para enviar datagramas, ciertas modificaciones se hicieron después a mano propia, ya que el chat a este punto no generaba soluciones correctas o modificaba parte del código que no es necesario.

Códigos

Envía Archivo

The screenshot shows the Visual Studio Code interface with the title bar "Desarrollo de Sistemas Distribuidos". The left sidebar shows "Tareas 2 > Envía Archivo.java >". The main editor area contains Java code for sending files. The code imports various Java packages related to cryptography and sockets. It defines a static class "Worker" that extends "Thread". The "run" method initializes a "DatagramSocket", sets its timeout to 500ms, and then enters a loop where it creates a "ByteBuffer" for the message, encrypts it using AES-256, creates a "DatagramPacket" with the encrypted trama, and sends it to the specified IP and port. It then receives a response, checks its integrity using CRC32, and prints an error if it fails. The code also handles exceptions like "SocketTimeoutException". The right sidebar shows the file tree with files like "Envía Archivo.java", "RecibeArchivo.java", and "Tarea2.docx". The bottom status bar shows "Ln 1, Col 28" and "11:07 p.m. 28/09/2023".

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.SSLSocketFactory;
import java.io.*;
import java.net.InetAddress;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.security.SecureRandom;
import java.util.Arrays;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.zip.CRC32;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketTimeoutException;
```

```
public class EnvíaArchivo {
```

```
    static class Worker extends Thread {
```

```
        private int id;
```

```
        private int puerto;
```

```
        private InetAddress direcciónIP;
```

```
        private byte[] segmento;
```

```
        private byte[] claveAES;
```

```
        Worker(int id, InetAddress direcciónIP, int puerto, byte[] segmento, byte[] claveAES) {
```

```
            this.id = id;
```

```
            this.puerto = puerto;
```

```
            this.direcciónIP = direcciónIP;
```

```
            this.segmento = segmento;
```

```
            this.claveAES = claveAES;
```

```
        }
```

```
        public void run() {
```

```
            System.out.println("Se inicio el worker[" + id + "]");
```

```
            try {
```

```
                DatagramSocket socket = new DatagramSocket();
```

```
                socket.setSoTimeout(500);
```

```
                while (true) {
```

```
                    ByteBuffer trama = ByteBuffer.allocate(LONGITUD).putInt(id).put(segmento);
```

```
                    trama.putLong(calcularCRC32(trama));
```

```
                    ByteToBuffer tramaEncriptada = cifrar(trama, claveAES); // Encriptar la trama usando AES-256
```

```
                    DatagramPacket paquete = new DatagramPacket(tramaEncriptada.array(), LONGITUD, direcciónIP, puerto);
```

```
                    socket.send(paquete); // enviamos el paquete encriptado con la clave
```

```
                    byte[] respuesta = new byte[9];
```

```
                    DatagramPacket paqueteRespuesta = new DatagramPacket(respuesta, respuesta.length);
```

```
                    try {
```

```
                        socket.receive(paqueteRespuesta);
```

```
                    } catch (SocketTimeoutException e) {
```

```
                        System.out.println("id=" + id + "\t" + "No se recibió respuesta, re-envia datagrama");
```

```
                        continue;
```

```
                    }
```

```
                    ByteToBuffer tramaRespuesta = ByteToBuffer.wrap(paqueteRespuesta.getData());
```

```
                    byte codigo = tramaRespuesta.get();
```

```
                    long crcRespuesta = tramaRespuesta.getLong();
```

```
                    if (verificarCRC32(paqueteRespuesta, crcRespuesta)) {
```

```
                        System.out.println("id=" + id + "\t" + "Error de integridad en la respuesta, re-envia datagrama");
```

```
                        continue;
```

```
                    }
```

```
                    if (codigo == 100)
```

```
                } finally {
```

Apartado para el envío de datagramas.

This screenshot shows the same Java code as the previous one, but the section involving the creation and sending of a "DatagramPacket" is highlighted with a yellow background. The code is identical to the first screenshot, detailing the encryption of the message using AES-256 and the sending of the encrypted datagram. The right sidebar and status bar are also visible.

```
        DatagramSocket socket = new DatagramSocket();
```

```
        socket.setSoTimeout(500);
```

```
        while (true) {
```

```
            ByteBuffer trama = ByteBuffer.allocate(LONGITUD).putInt(id).put(segmento);
```

```
            trama.putLong(calcularCRC32(trama));
```

```
            ByteToBuffer tramaEncriptada = cifrar(trama, claveAES); // Encriptar la trama usando AES-256
```

```
            DatagramPacket paquete = new DatagramPacket(tramaEncriptada.array(), LONGITUD, direcciónIP, puerto);
```

```
            socket.send(paquete); // enviamos el paquete encriptado con la clave
```

```
            byte[] respuesta = new byte[9];
```

```
            DatagramPacket paqueteRespuesta = new DatagramPacket(respuesta, respuesta.length);
```

```
            try {
```

```
                socket.receive(paqueteRespuesta);
```

```
            } catch (SocketTimeoutException e) {
```

```
                System.out.println("id=" + id + "\t" + "No se recibió respuesta, re-envia datagrama");
```

```
                continue;
```

```
            }
```

```
            ByteToBuffer tramaRespuesta = ByteToBuffer.wrap(paqueteRespuesta.getData());
```

```
            byte codigo = tramaRespuesta.get();
```

```
            long crcRespuesta = tramaRespuesta.getLong();
```

```
            if (verificarCRC32(paqueteRespuesta, crcRespuesta)) {
```

```
                System.out.println("id=" + id + "\t" + "Error de integridad en la respuesta, re-envia datagrama");
```

```
                continue;
```

```
            }
```

```
            if (codigo == 100)
```

```
        } finally {
```

Un tema importante es que, para hacer la conexión con el servidor, se debe importar el trustStore al cliente, con el certificado del servidor agregado, de esta forma no habrá ningún error al conectarnos.

```

1 EnviaArchivo.java 1, U  RecibeArchivo.java
Tarea 2 > EnviaArchivo.java > ...
81     } catch (Exception e) {
82         System.out.println("El worker[" + id + "] tuvo un error y es: ");
83         e.printStackTrace();
84     }
85 }
86 }
87
88 Run | Debug
89 public static void main(String[] args) throws Exception {
90
91     if (args.length != 1) {
92         System.err.println("Debes proporcionar la ubicación de un archivo como argumento.");
93         return;
94     }
95
96     String archivoUbicacion = args[0];
97     File archivo = new File(archivoUbicacion);
98
99     if (!archivo.exists()) {
100        System.err.println("El archivo especificado no existe.");
101        return;
102    }
103
104    String nombreArchivo = archivo.getName(); // Obtenemos el nombre del archivo y su tamaño
105    long tamanoArchivo = archivo.length(); // Obtenemos el tamaño del archivo
106    byte[] claveAES = generarClaveAES256(); // Generar una clave AES-256
107
108    // Propiedades del almacén de claves
109    System.setProperty("key.javax.net.ssl.keyStore", value:"keystore_cli.jks");
110    System.setProperty("key.javax.net.ssl.keyStorePassword", value:"123456");
111    System.setProperty("key.javax.net.ssl.trustStore", value:"truststore_cli.jks");
112    System.setProperty("key.javax.net.ssl.trustStorePassword", value:"123456");
113
114    System.out.println(Arrays.toString(claveAES));
115    enviarInformacionAlServidor(nombreArchivo, tamanoArchivo, claveAES); // Enviamos la información por medio de
116                                            // sockets seguros.
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149

```

Ln 1, Col 28 Spaces: 3 UTF-8 CRLF ⓘ Java Go Live ⓘ
11:09 p.m. 28/09/2023 ⓘ

```

1 EnviaArchivo.java 1, U  RecibeArchivo.java
Tarea 2 > EnviaArchivo.java > EnviaArchivo > main(String[])
113 System.out.println(Arrays.toString(claveAES));
114 enviarInformacionAlServidor(nombreArchivo, tamanoArchivo, claveAES); // Enviamos la información por medio de
115                                            // sockets seguros.
116
117 AtomicInteger hilos = new AtomicInteger(initialValue:0);
118 Worker[] workers = new Worker[MAXIMO_DE_HILOS];
119 int id = 0;
120
121 try (FileInputStream archivoInput = new FileInputStream(archivo)) {
122     byte[] buffer = new byte[LONGITUD - 12];
123     int bytesRead;
124     InetAddress direccionIP = InetAddress.getByName(IP);
125
126     System.out.println("Enviando info");
127
128     while ((bytesRead = archivoInput.read(buffer)) != -1) {
129
130         if (bytesRead > 0) {
131             workers[id] = new Worker(id, direccionIP, PUERTO_DATAGRAMA, buffer, claveAES);
132             workers[id].start();
133         }
134
135         id++;
136     }
137
138 } catch (IOException e) {
139     e.printStackTrace();
140 }
141
142 for (int i = 0; i < id; i++) {
143     workers[i].join();
144 }
145
146
147
148 public static byte[] generarClaveAES256() {
149     SecureRandom random = new SecureRandom(); // Generamos un generador de números aleatorios seguro

```

Ln 135, Col 18 Spaces: 3 UTF-8 CRLF ⓘ Java Go Live ⓘ
11:10 p.m. 28/09/2023 ⓘ

```

161     public static byte[] descifrar(ByteBuffer buffer, byte[] clave) throws Exception {
162         Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); // Instanciamos un desificador AES-256
163         cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(clave, "AES")); // Inicializamos el desificador con la clave
164         return ByteBuffer.wrap(cipher.doFinal(buffer.array())); // Desciframos el contenido del buffer
165     }
166
167     public static long calcularCRC32(ByteBuffer buffer) {
168         CRC32 crc32 = new CRC32();
169         crc32.update(buffer.array(), off:0, LONGITUD - 8);
170         return crc32.getValue();
171     }
172
173     public static boolean verificarCRC32(DatagramPacket data, Long crcEnviado) {
174         CRC32 crc32 = new CRC32();
175         crc32.update(data.getData(), off:0, len:1);
176         return crc32.getValue() == crcEnviado;
177     }
178
179     public static void enviarInformacionAlServidor(String nombreArchivo, Long tamanoArchivo, byte[] claveAES)
180             throws Exception {
181
182         SSLSocketFactory cliente = (SSLSocketFactory) SSLSocketFactory.getDefault();
183         Socket conexion = cliente.createSocket(IP, PUERTO_SSL);
184         DataOutputStream salida = new DataOutputStream(conexion.getOutputStream());
185
186         salida.writeUTF(nombreArchivo); // Enviar el nombre del archivo al servidor
187         salida.writeLong(tamanoArchivo); // Enviar el tamaño del archivo al servidor
188         salida.write(claveAES); // Enviar la clave AES-256 al servidor
189
190         conexion.close(); // Cerrar la conexión al finalizar
191     }
192 }
193

```

Recibe Archivo

```

1 import java.io.*;
2 import java.net.DatagramPacket;
3 import java.net.DatagramSocket;
4 import java.net.InetAddress;
5 import java.net.ServerSocket;
6 import java.net.Socket;
7 import java.net.SocketTimeoutException;
8 import java.nio.ByteBuffer;
9 import java.util.Arrays;
10 import java.util.zip.CRC32;
11
12 import javax.crypto.Cipher;
13 import javax.crypto.spec.SecretKeySpec;
14 import javax.net.ssl.SSLSocketFactory;
15
16 public class RecibeArchivo {
17
18     static final int LONGITUD = 1024;
19     static int DATAGRAMAS_RECIBIDOS = 0;
20     static int SEGMENTOS;
21     static boolean[] RECIBIDOS;
22
23     static void recibe Datagramas(DatagramSocket socket, byte[] claveAES) throws Exception {
24         System.out.println("Entramos a la funcion recibe_datagrama");
25         while (true) {
26             ByteBuffer buffer = new ByteBuffer[LONGITUD];
27             boolean correcto;
28             boolean duplicado = false;
29             DatagramPacket paquete = new DatagramPacket(buffer, LONGITUD);
30
31             while (true) {
32                 try {
33                     socket.receive(paquete);
34                 break;
35             } catch (SocketTimeoutException e) {
36                 if (DATAGRAMAS_RECIBIDOS == SEGMENTOS)
37                     return;
38             }
39         }
40     }
41
42     static void enviaArchivo() {
43
44     }
45 }
46

```

Desciframos la trama cifrada.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure for "Tarea 2" containing "EnviaArchivo.java", "RecibeArchivo.java", "utils", and "certificado_serv.pem".
- Code Editor:** Displays the content of "RecibeArchivo.java". The code handles receiving a file transfer, extracting the file name from the package, and verifying it using CRC32. It also includes logic to handle collisions and build a response message.
- Status Bar:** Shows the current file is "main", line 14, column 45, and the date/time as 11:11 p.m. on 28/09/2023.

```
1 EnviaArchivo.java 1, U 2 RecibeArchivo.java U X
Tarea 2 > RecibeArchivo.java > ...
41     ByteBuffer tramaCifrada = ByteBuffer.wrap(paquete.getData());
42     System.out.println(Arrays.toString(tramaCifrada.array()));
43     ByteBuffer tramaRecibida = descifrar(tramaCifrada, claveAES);
44     int id = tramaRecibida.get();
45     byte[] datos = new byte[LONGITUD - 12]; // arreglo donde almacenemos la informacion
46     tramaRecibida.get(datos); // obtenemos los datos
47     Long crcRecibido = tramaRecibida.getLong(); // obtenemos el crc recibido
48
49     System.out.println("Se recibio la trama " + id + " Con informacion: " + Arrays.toString(datos));
50
51     correcto = verificarCRC32(paquete, crcRecibido); // verificamos si el crc del paquete es igual al crc recibido
52
53     if (correcto) {
54         duplicado = RECIBIDOS[id];
55         if (!duplicado) {
56             RECIBIDOS[id] = true;
57             DATAGRAMAS_RECIBIDOS++;
58         }
59     }
60
61     // obtiene la direccion IP y el puerto del remitente
62     InetAddress direccionIP = paquete.getAddress();
63     int puerto = paquete.getPort();
64
65     System.out.println(id + "\t" + (correcto ? "OK" : "Error") + "\t" + (duplicado ? "Duplicado" : "\t") + "\t"
66     + direccionIP.toString() + "\t" + puerto);
67
68     byte codigo = (byte) (correcto ? 100 : 200);
69
70     // utiliza un ByteBuffer para empacar el código de respuesta y el CRC
71     ByteBuffer tramRespuesta = ByteBuffer.allocate(capacidad9); // 1 byte del código de respuesta y 8 bytes para el CRC de
72     // la trama
73     tramRespuesta.put(codigo);
74     tramRespuesta.putLong(cacularCRC32(tramRespuesta)); // Creamos el CRC de
75
76     DatagramPacket paqueteRespuesta = new DatagramPacket(tramRespuesta.array(), length9, direccionIP, puerto);
77
78 main* ⚡ ④ 8 0 Git Graph ⚡ #832561
Búsqueda 11:11 p.m. 28/09/2023
In 14, Col 45 Spaces: 3 UTF-8 CRLF {} Java Go Live
11:11 p.m. 28/09/2023
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure for "Tarea 2" containing "EnviaArchivo.java", "RecibeArchivo.java", "utils", and "certificado_serv.pem".
- Code Editor:** Displays the content of "RecibeArchivo.java". The code includes methods for encrypting and decrypting buffers, reading data from a DataInputStream, and managing a Worker thread for socket operations.
- Status Bar:** Shows the current file is "main", line 14, column 45, and the date/time as 11:12 p.m. on 28/09/2023.

```
1 EnviaArchivo.java 1, U 2 RecibeArchivo.java U X
Tarea 2 > RecibeArchivo.java > ...
87     public static boolean verificarCRC32(DatagramPacket data, long crcEnviado) { ...
88
89     public static ByteBuffer cifrar(ByteBuffer buffer, byte[] clave) throws Exception {
90         Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); // Instanciamos un cifrador AES-256
91         cipher.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(clave, "AES")); // Inicializamos el cifrador con la clave
92         return ByteBuffer.wrap(cipher.doFinal(buffer.array())); // Ciframos el contenido del buffer
93     }
94
95     public static ByteBuffer descifrar(ByteBuffer buffer, byte[] clave) throws Exception {
96         Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); // Instanciamos un descifrador AES-256
97         cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(clave, "AES")); // Inicializamos el descifrador con la clave
98         return ByteBuffer.wrap(cipher.doFinal(buffer.array())); // Desciframos el contenido del buffer
99     }
100
101    public static void read(DataInputStream f, byte[] b, int posicion, int longitud) throws IOException {
102
103        while (longitud > 0) {
104            int n = f.read(b, posicion, longitud);
105            posicion += n;
106            longitud -= n;
107        }
108    }
109
110    static class Worker extends Thread {
111        private Socket socket;
112
113        Worker(Socket socket) {
114            this.socket = socket;
115        }
116
117        public void run() {
118            try {
119                // Recibe información del cliente
120                DataInputStream inputStream = new DataInputStream(socket.getInputStream());
121                String nombreArchivo = inputStream.readUTF();
122            } catch (IOException e) {
123                e.printStackTrace();
124            }
125        }
126    }
127
128 main* ⚡ ④ 8 0 Git Graph ⚡ #832561
Búsqueda 11:12 p.m. 28/09/2023
In 14, Col 45 Spaces: 3 UTF-8 CRLF {} Java Go Live
11:12 p.m. 28/09/2023
```

Recibimos los datos del cliente, como la clave, el tamaño y la clave.

```

123 public void run() {
124     try {
125         // Recibir información del cliente
126         DataInputStream inputStream = new DataInputStream(socket.getInputStream());
127         String nombreArchivo = inputStream.readUTF();
128         long tamanoArchivo = inputStream.readLong();
129
130         byte[] claveAES = inputStream.readBytes(len:32);
131
132         SEGMENTOS = (int) Math.ceil(double) tamanoArchivo / LONGITUD; // Calcular el número de segmentos
133         RECIBIDOS = new boolean[SEGMENTOS]; // Inicializar el arreglo RECIBIDOS
134
135         System.out.println(nombreArchivo);
136         System.out.println(tamanoArchivo);
137         System.out.println(Arrays.toString(claveAES));
138         System.out.println(SEGMENTOS);
139         socket.close();
140
141     try {
142
143         DatagramSocket socketDatagrama = new DatagramSocket(port:50001);
144         socketDatagrama.setSoTimeout(timeout:500);
145
146         recibir_datagramas(socketDatagrama, claveAES);
147
148         socketDatagrama.close();
149     } catch (Exception e) {
150         e.printStackTrace();
151     }
152
153     } catch (Exception e) {
154         e.printStackTrace();
155     }
156
157 }
158

```

File: EnviaArchivo.java 1 U RecibeArchivo.java U ...

Tarea 2 > RecibeArchivo.java > ...

Line 14, Col 45 Spaces: 3 UTF-8 CRLF ⓘ Java ⓘ Go Live ⓘ

11:12 p.m. 28/09/2023

```

151     }
152
153     } catch (Exception e) {
154         e.printStackTrace();
155     }
156
157 }
158
Run | Debug
public static void main(String[] args) throws Exception {
159
160     System.setProperty(key:"javax.net.ssl.keyStore", value:"keystore_serv.jks");
161     System.setProperty(key:"javax.net.ssl.keyStorePassword", value:"1234567");
162
163     SSLServerSocketFactory socketFactory = (SSLServerSocketFactory) SSLServerSocketFactory.getDefault();
164     ServerSocket serverSocket = socketFactory.createServerSocket(50000);
165
166     while (true) {
167         Socket conexion = serverSocket.accept();
168         Worker con = new Worker(conexion);
169         con.start();
170         con.join();
171     }
172
173 }
174
175 }
176

```

File: EnviaArchivo.java 1 U RecibeArchivo.java U ...

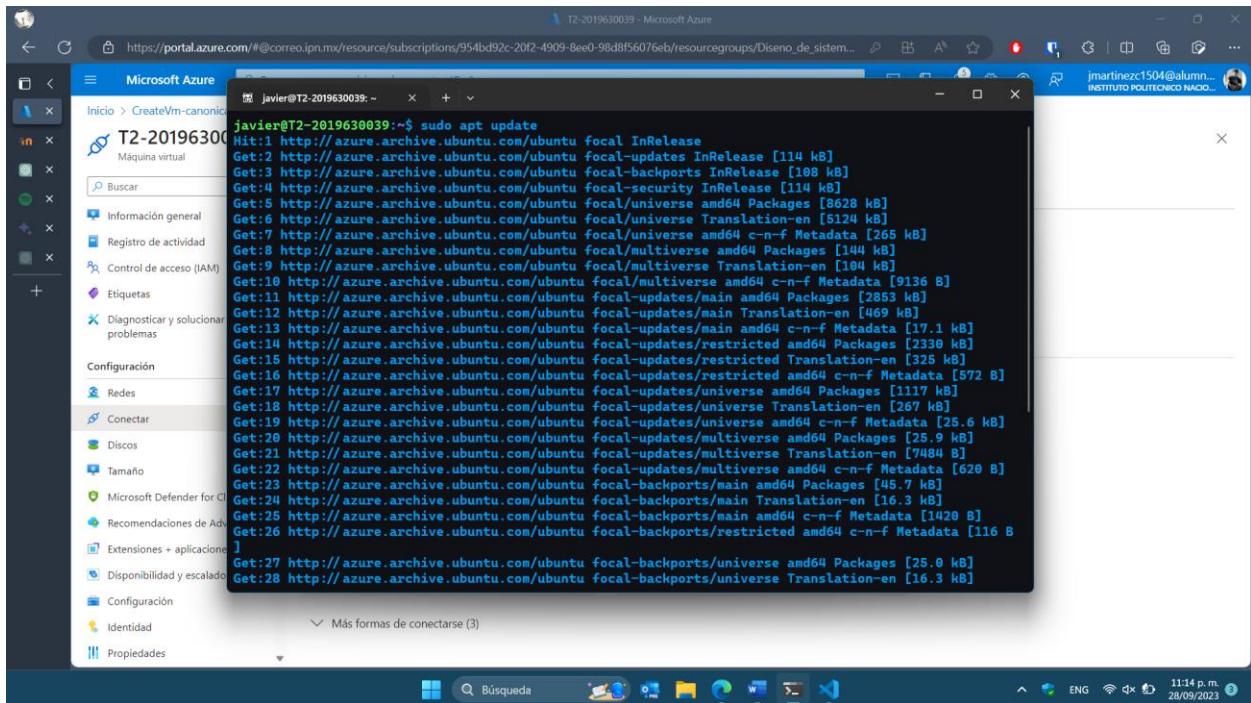
Tarea 2 > RecibeArchivo.java > ...

Line 14, Col 45 Spaces: 3 UTF-8 CRLF ⓘ Java ⓘ Go Live ⓘ

11:12 p.m. 28/09/2023

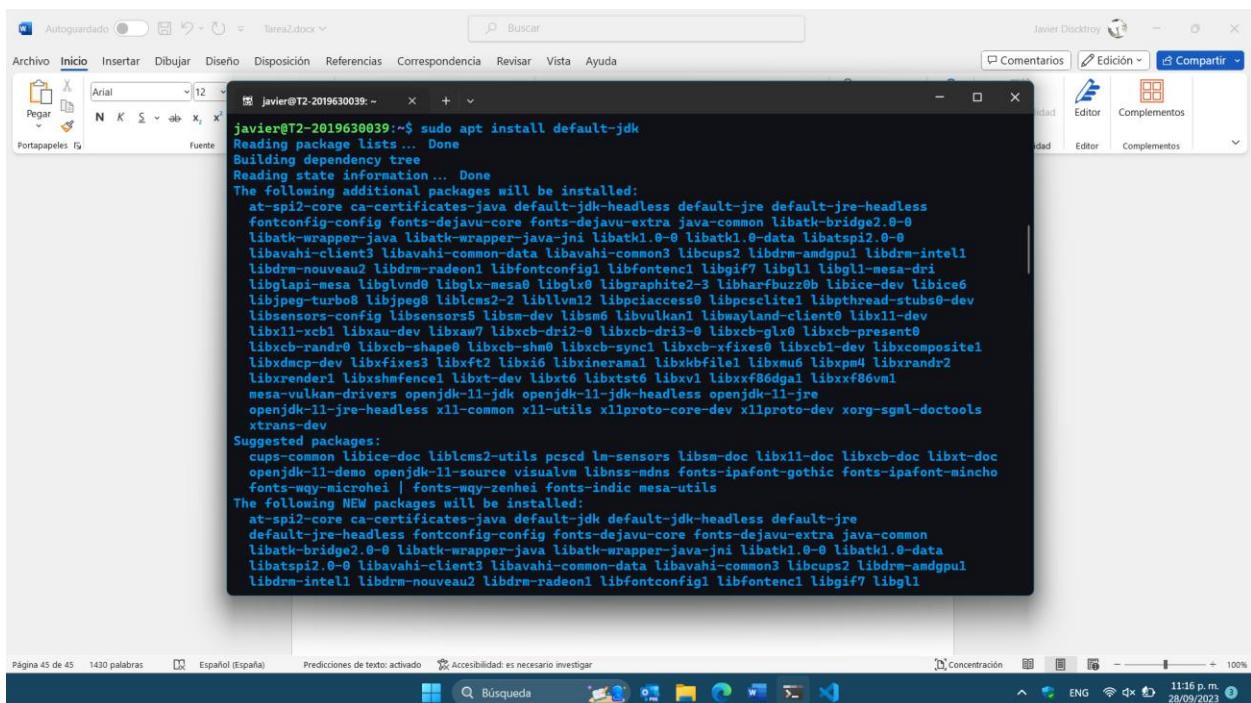
Ejecución y pruebas

Actualizamos el administrador de paquetes de Ubuntu.



```
javier@T2-2019630039:~$ sudo apt update
Get:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease [114 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease [108 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 kB]
Get:11 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2853 kB]
Get:12 http://azure.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [469 kB]
Get:13 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [17.1 kB]
Get:14 http://azure.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [2330 kB]
Get:15 http://azure.archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [325 kB]
Get:16 http://azure.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 c-n-f Metadata [5728 kB]
Get:17 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1117 kB]
Get:18 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [267 kB]
Get:19 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [25.6 kB]
Get:20 http://azure.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [25.9 kB]
Get:21 http://azure.archive.ubuntu.com/ubuntu focal-updates/multiverse Translation-en [7484 kB]
Get:22 http://azure.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 c-n-f Metadata [620 kB]
Get:23 http://azure.archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [45.7 kB]
Get:24 http://azure.archive.ubuntu.com/ubuntu focal-backports/main Translation-en [16.3 kB]
Get:25 http://azure.archive.ubuntu.com/ubuntu focal-backports/main amd64 c-n-f Metadata [1420 kB]
Get:26 http://azure.archive.ubuntu.com/ubuntu focal-backports/restricted amd64 c-n-f Metadata [116 kB]
Get:27 http://azure.archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [25.0 kB]
Get:28 http://azure.archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [16.3 kB]
```

Instalamos Java 11.0



```
javier@T2-2019630039:~$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  at-spi2-core ca-certificates-java default-jdk-headless default-jre default-jre-headless
  fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common libatk-bridge2.0-0
  libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0
  libavahi-client3 libavahi-common-data libavahi-common3 libcurl2 libdrm-amdgpu libdrm-intel
  libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libgif7 libgl1 libgl1-mesa-dri
  libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-2 libharfbuzz0a libice-dev libice6
  libjpeg-turbo libjpeg8 liblcms2-2 liblvm12 libpciaccess0 libpccsclite1 libpthread-stubs0-dev
  libssensors-config libssensors5 libsm-dev libsm6 libvulkan1 libwayland-client0 libx11-dev
  libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0
  libxcb-randr1 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposit1
  libxdmcp-dev libxfixes3 libxtf2 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2
  libxrender1 libxshmfence1 libxt6 libxtst6 libxv1 libxf86dg1 libxf86v1
  mesa-vulkan-drivers openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
  openjdk-11-jre-headless x11-common x11-utils x11proto-core-dev x11proto-dev xorg-sgml-doctools
  xtrans-dev
Suggested packages:
  cups-common libice-doc liblcms2-utils pscod lm-sensors libsm-doc libx11-doc libxcb-doc libxt-doc
  openjdk-11-demo openjdk-11-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei | fonts-wqy-zhenhei fonts-indic mesa-utils
The following NEW packages will be installed:
  at-spi2-core ca-certificates-java default-jdk default-jdk-headless default-jre
  default-jre-headless fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common
  libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data
  libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcurl2 libdrm-amdgpu
  libdrm-intel libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libglif7 libgl1
```

Importamos el código a la máquina virtual.

Tarea 2 > RecibeArchivo.java U

```
GNU nano 4.8          RecibeArchivo.java

1 import java.io.*;
2 import java.net.DatagramPacket;
3 import java.net.DatagramSocket;
4 import java.net.InetAddress;
5 import java.net.ServerSocket;
6 import java.net.Socket;
7 import java.net.SocketTimeoutException;
8 import java.nio.ByteBuffer;
9 import java.util.Arrays;
10 import java.util.zip.CRC32;
11
12 import javax.crypto.Cipher;
13 import javax.crypto.spec.SecretKeySpec;
14 import javax.net.ssl.SSLSocketFactory;
15
16 public class RecibeArchivo {
17
18     static final int LONGITUD = 1024;
19     static int DATAGRAMAS_RECIBIDOS = 0;
20     static int SEGMENTOS;
21     static boolean[] RECIBIDOS;
22
23     static void recibe_datos() {
24         System.out.println("Entramos a la funcion recibe_datos");
25         while (true) {
26             byte[] buffer = new byte[LONGITUD];
27             boolean correcto;
28             boolean duplicado;
29             DatagramPacket paquete;
30
31             while (true) {
32                 try {
33                     socket.receive(buffer);
34                     break;
35                 } catch (SocketTimeoutException e) {
36                     if (DATAGRAMAS_RECIBIDOS == SEGMENTOS)
37                         return;
38                 }
39             }
40             correcto = buffer[0] == 1;
41             duplicado = RECIBIDOS[correcto];
42
43             if (correcto & !duplicado) {
44                 RECIBIDOS[correcto] = true;
45                 DATAGRAMAS_RECIBIDOS++;
46                 System.out.println("Recibido segmento " + DATAGRAMAS_RECIBIDOS);
47             }
48         }
49     }
50
51     static void recibe_datagramas(DatagramSocket socket, byte[] claveAES) throws Exception {
52         System.out.println("Entramos a la funcion recibe_datagrama");
53         while (true) {
54             byte[] buffer = new byte[LONGITUD];
55
56             while (true) {
57                 try {
58                     socket.receive(buffer);
59                     break;
60                 } catch (SocketTimeoutException e) {
61                     continue;
62                 }
63             }
64             String mensaje = new String(buffer);
65             System.out.println(mensaje);
66         }
67     }
68 }
```

Get Help Write Out Where Is Cut Text Justify Cur Pos Undo
Exit Read File Replace Paste Text To Spell Go To Line Redo

Ln 176, Col 1 (6601 selected) Spaces: 3 UTF-8 CRLF Java Go Live

File Edit Selection View Go Run Terminal Help ↺ ↻ Desarrollo de Sistemas Distribuidos

EXPLORER ...

- enviaArchivo.java
- cliente-Servidor-inicial
- Cliente-Servidor-inicial
- Servidor y Cliente SSL
- Servidores HTTP y HTTPS
- Sockets Datagrama
- Tarea 1
- Tarea 2
 - utils
 - cliente.docx
 - certificado_serv.pem
 - EnviaArchivo.java
 - keystore_cli.jks
 - RecibeArchivo.java
 - Tarea2.docx
 - truststore_cli.jks
 - .gitignore
 - README.md

11:19 p.m.
28/09/2023

Corriendo ambos programas vemos cómo va bien, sin embargo a la mitad del proceso ocurre un error con la trama enviada y que está cifrada.

Este es un error derivado a que no se pudo desencriptar de forma correcta cierta trama, el chat no pudo solucionar este error y esto hacía que el cliente no recibiera la respuesta a las tramas correctamente.

```

3, -34, 10, -108, -113, -89, -22, -58, -82, -124, 116, -48, 78, 96, -106, -3, -91, -27, 76, 33, 15, -95, -4, -48, -112, -111, 19, 114, 88, -53, -68, 113, -114, -10, 85, -184, -50, -105, 28, -68, -41, -58, 34, 102, 17, -66, -118, -64, 83, -19, 26, -46, -94, 86, 17, 14, -64, -122, -38, -63, 73, -78, 14, 12, , 87, -49, -73, 75, 48, -23, 37, -18, 10, -126, 84, -46, 125, -6, 38, -37, -75, 107, -119, -39, -14, 8, 22, 88, 117, 39, -46, 13, -31, -47, 108, -51, 15, -58, -37, 113, -115, -68, 85, 95, -6, 88, 12, 1, -124, -38, 18, -31, 53, 1 3, -123, 69, 121, -88, 78, -63, 17, 2, -36, 74, 69, -11, 17, -4, -83, 75, -4 4, 47, -6, -8, 118, 118, -23, -107, -84, 34, 63, -42, 68, 31, 71, 43, 20, -6 4, 0, -87, 117, 185, -125, 68, -40, -124, -10, 91, -39, -77, -111, -109, -99 , -118, 71, -77, 121, -25, 96, -5, -71, 123, 108, -27, 25, -47, -13, 117, 79 , 126, 44, 59, -68, -10, -180, 72, 40, -85, 126, -58, 89, -125, 105, 51, -74 , 104, 181, 79, 33, -68, -40, 77, 47, 47, 7, 61, 43, 58, -74, 116, -118, -10 4, -17, -2, 102, -43, 8, 42, 18, 118, -87, 0, 125, -7, -92, 95, -103, 94, 21 , -107, -92, -83, -96, -5, 39, -8, 71, 86, 107, 101, 65, -43, 40, -25, -98, 74, 105, 88, 113, 114, -21, -92, -3, 34, 78, -34, 31, 119, -61, -122, -32, -28, 25, 44, 105, -4, -57, 17, -42, 68, 98, -16, 92, 106, -73, -54, 103, -68, 42, 28, -24, 75, -85, 100, 12, -48, 49, 88, -43, 37, 25, 2, 122, -18, -93, -75, 65, -119, 89, 59, -68, -79, -44, 124, 94, 98, -124, -97, 79, -19, -11, 6, -90, 113, 56, 27, 110, -23, -110, 116, -4, -124, -37, 86, -84, 24, 26, -38, -72, 82, 121, 81, 11, 85, 10, 30, -77, 51, 100, 54, 118, -70, -66, 12, 71, 85 , 76, 66, 111, -77, 16, -120, 46, 87, 96, 115, 58, -72, 76, 112, 118, 106, 8 , 82, 12, 83, -62, -82, 106, 87, 60, -97, -48, 59, -109, 46, -43, 2, 29, 78, -65, -103, -109, -89, -41, 22, 101, 121, -9, -76, 56, 87, -41, 2, -2, -33, -108, -29, -29, -86, 98, 12, 68, -5, 60, 8, 39, -23, -33, -128, -69, 72] javax.crypto.BadPaddingException: Given final block not properly padded. Such issues can arise if a bad key is used during decryption.
    at java.base/com.sun.crypto.provider.CipherCore.unpad(CipherCore.java:981)
    at java.base/com.sun.crypto.provider.CipherCore.fillOutputBuffer(CipherCore.java:1062)
    at java.base/com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:853)
    at java.base/com.sun.crypto.provider.AESEncryptor.engineDoFinal(AESEncryptor.java:446)
    at java.base/javax.crypto.Cipher.doFinal(Cipher.java:2202)
    at RecibeArchivo.descifrar(RecibeArchivo.java:103)
    at RecibeArchivo.recibe_datagramas(RecibeArchivo.java:43)
    at RecibeArchivo$Worker.run(RecibeArchivo.java:146)
[[24]]

```

Conclusión

Para esta tarea el chat en ocasiones resultaba más un problema que una herramienta útil, no logre averiguar el prompt correcto con el cual el chat y yo nos entendiéramos correctamente, esto derivo en muchos fallos que al final tenía que resolver yo, esto es algo lógico ya que siempre se debe de checar la información que da el chat, sin embargo, había ocasiones en donde el chat cambiaba todo el código para hacer cosas que no tenían sentido. En general el algoritmo es bastante claro, sin embargo, el error del encRIPTADO me limito mucho al momento de poder desarrollar las demás partes faltantes, no logre encontrar el problema en este código ya que todo parece estar bien y funciona hasta cierto punto.

En general creo que fue una buena práctica ya que se vieron y aprendieron muchas cosas sobre la marcha, sin embargo algo que parecía sencillo se fue volviendo más y más complicado de implementar y lograr que todo funcione.