



**INSTITUTO POLITÉCNICO NACIONAL**

**Escuela Superior de Cómputo**



**UNIDAD DE APRENDIZAJE:**

Desarrollo de Sistemas Distribuidos

**Tarea 4. Sockets Seguros, Chatgpt**

**INTEGRANTES DEL EQUIPO**

- Chávez Morones Ángel Uriel

**NOMBRE DEL PROFESOR:**

Carlos Pineda

# Parte 1.

## Creación de la máquina virtual Ubuntu

Crear una máquina virtual - Micr...

portal.azure.com/#create/Microsoft.VirtualMachine-ARM

Microsoft Azure

Buscar recursos, servicios y documentos (G+/I)

achavez1601@alumn...  
INSTITUTO POLITECNICO NACIO...

Inicio > Máquinas virtuales >

Crear una máquina virtual

virtual con parámetros predeterminados o bien revise cada una de las pestañas para personalizar la configuración.  
[Más información](#)

Es posible que esta suscripción no sea apta para implementar máquinas virtuales de ciertos tamaños en determinadas regiones.

Detalles del proyecto

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

Suscripción \*

Azure for Students

Grupo de recursos \*

(Nuevo) T4-2020630067\_group

[Crear nuevo](#)

Detalles de instancia

Nombre de máquina virtual \*

T4-2020630067

Región \*

(US) Este de EE. UU. 2

Opciones de disponibilidad

No se requiere redundancia de la infraestructura

Tipo de seguridad

Estándar

Imagen \*

Ubuntu Server 18.04 LTS - x64 Gen2

[Ver todas las imágenes](#) | [Configurar la generación de máquinas virtuales](#)

Arquitectura de VM

Arm64

☒ x64

Ejecución de Azure Spot con descuento

☐

Tamaño \*

Standard\_DS1\_v2 - 1 vcpu, 3.5 GiB de memoria (41,61 US\$/mes)

[Ver todos los tamaños](#)

Cuenta de administrador

Revisar y crear

< Anterior

Siguiente: Discos >

Enviar comentarios

Crear una máquina virtual - Micro X

portal.azure.com/#create/Microsoft.VirtualMachine-ARM

Microsoft Azure

Buscar recursos, servicios y documentos (G+/)

achavezm1601@alumn...  
INSTITUTO POLITECNICO NACIO...

Inicio > Máquinas virtuales >

Crear una máquina virtual

X

Datos básicosDiscosRedesAdministraciónMonitoringOpciones avanzadasEtiquetasRevisar y crear

Las máquinas virtuales de Azure tienen un disco de sistema operativo y un disco temporal para el almacenamiento a corto plazo. Puede asociar discos de datos adicionales. El tamaño de la máquina virtual determina el tipo de almacenamiento que puede usar y la cantidad de datos que permiten los discos. [Más información](#)

VM disk encryption

Azure disk storage encryption automatically encrypts your data stored on Azure managed disks (OS and data disks) at rest by default when persisting it to the cloud.

Cifrado en el host ☐

El cifrado en el host no está registrado para la suscripción seleccionada.

[Más información sobre cómo habilitar esta característica](#)

OS disk

Tipo de disco del sistema operativo \* ☐ HDD estándar (almacenamiento con redundancia local)

Si el rendimiento es crítico para las cargas de trabajo, elija discos SSD Premium para reducir la latencia, IOPS y anchos de banda más altos y expansión de disco. [Más información](#)

Eliminar con VM ☒

Key management ☐ Clave administrada por la plataforma

Habilitar compatibilidad con Ultra Disks ☐

No se admite el disco Ultra para el tamaño de VM seleccionado (Standard\_DS1\_v2) en Este de EE. UU. 2.

Discos de datos para T4-2020630067

Puede agregar y configurar discos de datos adicionales para su máquina virtual o asociar discos existentes. Esta máquina virtual también incluye un disco temporal.

LUN	Nombre	Tamañ...	Tipo de disco	Almacena...	Eliminar con VM
-----	--------	----------	---------------	-------------	-----------------

[Crear y adjuntar un nuevo disco](#) [Asociar un disco existente](#)

Revisar y crear

< Anterior

Siguiente: Redes >

Enviar comentarios

Inicio > Máquinas virtuales >

## Crear una máquina virtual

Datos básicos   Discos   Redes   Administración   **Monitoring**   Opciones avanzadas   Etiquetas   Revisar y crear

Configure monitoring options for your VM.

### Alerts

Enable recommended alert rules ☐

### Diagnostics

Boot diagnostics ☐ Enable with managed storage account (recommended)  
☐ Enable with custom storage account  
☒ Disable

Enable OS guest diagnostics ☐

Revisar y crear

< Anterior

Siguiente: Opciones avanzadas >

Enviar comentarios

Crear una máquina virtual - Micro X

portal.azure.com/#create/Microsoft.VirtualMachine-ARM

Microsoft Azure

Buscar recursos, servicios y documentos (G+)

achavezm1601@alumn...  
INSTITUTO POLITÉCNICO NACIO...

Inicio > Máquinas virtuales >

Crear una máquina virtual

✕

✓ Validación superada

Datos básicos

Discos

Redes

Administración

Monitoring

Opciones avanzadas

Etiquetas

Revisar y crear

❗ El costo que se indica a continuación es una estimación y no el precio final. Use [Calculadora de precios](#) para todas sus necesidades de precios.

Price

1 X Standard DS1 v2  
by Microsoft  
[Terms of use](#) | [Privacy policy](#)

Subscription credits apply ⓘ  
**0,0570 USD/hr**  
[Pricing for other VM sizes](#)

TERMS

By clicking "Crear", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

⚠ Ha establecido los siguientes puertos abiertos para Internet: SSH. Esto solo se recomienda para las pruebas. Si quiere cambiar esta configuración, vuelva a la pestaña de aspectos básicos.

Datos básicos

Suscripción

Azure for Students

Grupo de recursos

(nuevo) T4-2020630067\_group

Nombre de máquina virtual

T4-2020630067

Región

Este de EE. UU. 2

Opciones de disponibilidad

No se requiere redundancia de la infraestructura

Crear

< Anterior

Siguiente >

[Descargar una plantilla para la automatización](#)

Enviar comentarios

T4-2020630067 - Microsoft Azure

portal.azure.com/#@correo.ipn.mx/resource/subscriptions/3bcd83a4-12de-4f3a-89a8-657a22a3e0a6/resourcegroups/T4-2020630067\_grou...

Microsoft Azure

Buscar recursos, servicios y documentos (G+/I)

achavezm1601@akumn...  
INSTITUTO POLITECNICO NACIO...

Inicio >

T4-2020630067  
Máquina virtual

Conectar ▾ ▶ Iniciar ↺ Reiniciar □ Detener 📷 Captura 🗑 Eliminar ↻ Actualizar 📱 Abrir en dispositivos móviles 🗨 Comentarios 📄 CLI / PS

Información general

Registro de actividad

Control de acceso (IAM)

Etiquetas

Diagnosticar y solucionar problemas

Configuración

Redes

Conectar

Discos

Tamaño

Microsoft Defender for Cloud

Recomendaciones de Advisor

Extensiones + aplicaciones

Entrega continua

Disponibilidad y escalado

Configuración

Identidad

Propiedades

Bloqueos

Operaciones

Bastión

Apagado automático

Backup

Recuperación ante desastres

T4-2020630067 estado del agente de máquina virtual no está listo. Solucionar el problema →

Información esencial

Vista JSON

Grupo de recursos (mover) : T4-2020630067\_group

Estado : En ejecución

Ubicación : Este de EE. UU. 2

Suscripción (mover) : Azure for Students

Id. de suscripción : 3bcd83a4-12de-4f3a-89a8-657a22a3e0a6

Etiquetas (editar) : Haga clic aquí para agregar etiquetas.

Sistema operativo : Linux

Tamaño : Standard DS1 v2 (1 vcpu, 3.5 GiB de memoria)

Dirección IP pública : 20.97.252.199

Red virtual/subred : T4-2020630067-vnet/default

Nombre DNS : Sin configurar

Propiedades

Supervisión

Funcionalidades (7)

Recomendaciones

Tutoriales

Máquina virtual

Nombre del equipo : T4-2020630067

Estado de mantenimiento : -

Sistema operativo : Linux

Publicador : Canonical

Oferta : UbuntuServer

Plan : 18\_04-lts-gen2

Generación de VM : V2

Arquitectura de VM : x64

Estado del agente : Not Ready

Versión del agente : Unknown

Grupo host : Ninguno

Host : -

Grupo con ubicación por proximidad : -

Estado de ubicación : N/D

Grupo de reserva de capacidad : -

Redes

Dirección IP pública : 20.97.252.199 ( Interfaz de red T4-2020630067846 )

Dirección IP pública (IPv6) : -

Dirección IP privada : 10.1.0.4

Dirección IP privada (IPv6) : -

Red virtual/subred : T4-2020630067-vnet/default

Nombre DNS : Configurar

Tamaño

Tamaño : Standard DS1 v2

vCPU : 1

RAM : 3.5 GiB

Disco

Disco del SO : T4-2020630067\_OsDisk\_1\_f91043f0b138466e975ff1af847637f6

Cifrado en el host : Deshabilitado

Azure Disk Encryption : No habilitado

Disco de SO efímero : N/D

Microsoft Azure portal showing the configuration of a network security rule (t4-2020630067-nsg) for a virtual machine (T4-2020630067).

The left sidebar shows the navigation menu with options like Información general, Registro de actividad, Control de acceso (IAM), Etiquetas, Diagnosticar y solucionar problemas, Configuración, Redes, Conectar, Discos, Tamaño, Microsoft Defender for Cloud, Recomendaciones de Advisor, Extensiones + aplicaciones, Entrega continua, Disponibilidad y escalado, Configuración, Identidad, Propiedades, Bloqueos, Operaciones, Bastión, Apagado automático, Backup, and Recuperación ante desastres.

The main content area displays the configuration of the network security rule (t4-2020630067-nsg) for the virtual machine (T4-2020630067). The rule is configured for the interface ipconfig1 (Principal) and is currently active. The rule is a "Regla de puerto de entrada" (Inbound port rule) with a priority of 300, allowing SSH access on port 22 over TCP. The rule is named "t4-2020630067-nsg" and is associated with the virtual machine (T4-2020630067).

The right sidebar shows the "Agregar regla de seguridad de entrada" (Add inbound security rule) dialog box. The rule is configured for the interface ipconfig1 (Principal) and is currently active. The rule is a "Regla de puerto de entrada" (Inbound port rule) with a priority of 300, allowing SSH access on port 22 over TCP. The rule is named "t4-2020630067-nsg" and is associated with the virtual machine (T4-2020630067).

Prioridad	Nombre	Puerto	Protocolo	Origen
300	SSH	22	TCP	Cualquiera
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetw
65001	AllowAzureLoadBalancerInBound	Cualquiera	Cualquiera	AzureLoadB
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera

¿Necesita ayuda?  
Información sobre el equilibrio de carga de Azure  
Inicio rápido: cree un equilibrador de carga público para equilibrar la carga Virtual Machines  
Inicio rápido: tráfico web directo con Azure Application Gateway

Dialog box "Agregar regla de seguridad de entrada" (Add inbound security rule) configuration:

- Origen: Any
- Intervalos de puertos de origen: \*
- Destino: Any
- Servicio: Custom
- Intervalos de puertos de destino: 8000
- Protocolo: TCP
- Acción: Permitir
- Prioridad: 310
- Nombre: AllowAnyCustom8000Inbound
- Descripción:

Buttons: Agregar, Cancelar, Enviar comentarios

Paso archivos a Ubuntu

Microsoft Azure portal showing system information and terminal output for a virtual machine.

**System Information:**

- Management: <https://landscape.canonical.com>
- Support: <https://ubuntu.com/advantage>
- System information as of Wed Apr 5 23:16:44 UTC 2023
- System load: 0.03
- Usage of /: 4.5% of 28.89GB
- Memory usage: 6%
- Swap usage: 0%
- Processes: 112
- Users logged in: 0
- IP address for eth0: 10.1.0.4

Expanded Security Maintenance for Applications is not enabled.

updates can be applied immediately.

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

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

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

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

**Terminal Output:**

```
ubuntu@T4-2020630067: ~$ scp C:/users/enduser10/Desktop/ESCOM/Desarrollo_Sistemas_Distribuidos/Tarea4 ubuntu@20.97.252.199:/home
ssh: Could not resolve hostname c: Temporary failure in name resolution
ubuntu@T4-2020630067: ~$ ls
ubuntu@T4-2020630067: ~$ mkdir home
ubuntu@T4-2020630067: ~$ ls
ubuntu@T4-2020630067: ~$ ls
ubuntu@T4-2020630067: ~$ cd home/
ubuntu@T4-2020630067: ~/home$ pwd
/home/ubuntu/home
ubuntu@T4-2020630067: ~/home$ cd ..
ubuntu@T4-2020630067: ~$ ls
SecureMultiThreadedServer.java
ubuntu@T4-2020630067: ~$ rm -r home
ubuntu@T4-2020630067: ~$ ls
SecureMultiThreadedServer.java
ubuntu@T4-2020630067: ~$ ls
SecureMultiThreadedServer.java  keystore.jks
ubuntu@T4-2020630067: ~$
```

**Administrador: Windows PowerShell**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\WINDOWS\system32> scp C:\Users\enduser10\Desktop\ESCOM\Desarrollo_Sistemas_Distribuidos\Tarea4\SecureMultiThreadedServer.java ubuntu@20.97.252.199:/home
ubuntu@20.97.252.199's password:
scp: /home/SecureMultiThreadedServer.java: Permission denied
PS C:\WINDOWS\system32> scp C:\Users\enduser10\Desktop\ESCOM\Desarrollo_Sistemas_Distribuidos\Tarea4\SecureMultiThreadedServer.java ubuntu@20.97.252.199:/home/ubuntu
ubuntu@20.97.252.199's password:
SecureMultiThreadedServer.java                               100% 6566   122.0KB/s   00:00
PS C:\WINDOWS\system32> scp C:\Users\enduser10\Desktop\ESCOM\Desarrollo_Sistemas_Distribuidos\Tarea4\keystore.jks ubuntu@20.97.252.199:/home/ubuntu
ubuntu@20.97.252.199's password:
keystore.jks                                                  100% 2742    47.4KB/s   00:00
PS C:\WINDOWS\system32>
```

**Vista JSON**

Operaciones	Estado del agente	Not Ready
Bloqueos	Versión del agente	Unknown
Operaciones	Grupo host	Ninguno
Bastión	Host	-
Apagado automático	Grupo con ubicación por proximidad	-
Backup	Estado de ubicación	N/D
Recuperación ante desastres	Grupo de reserva de capacidad	-

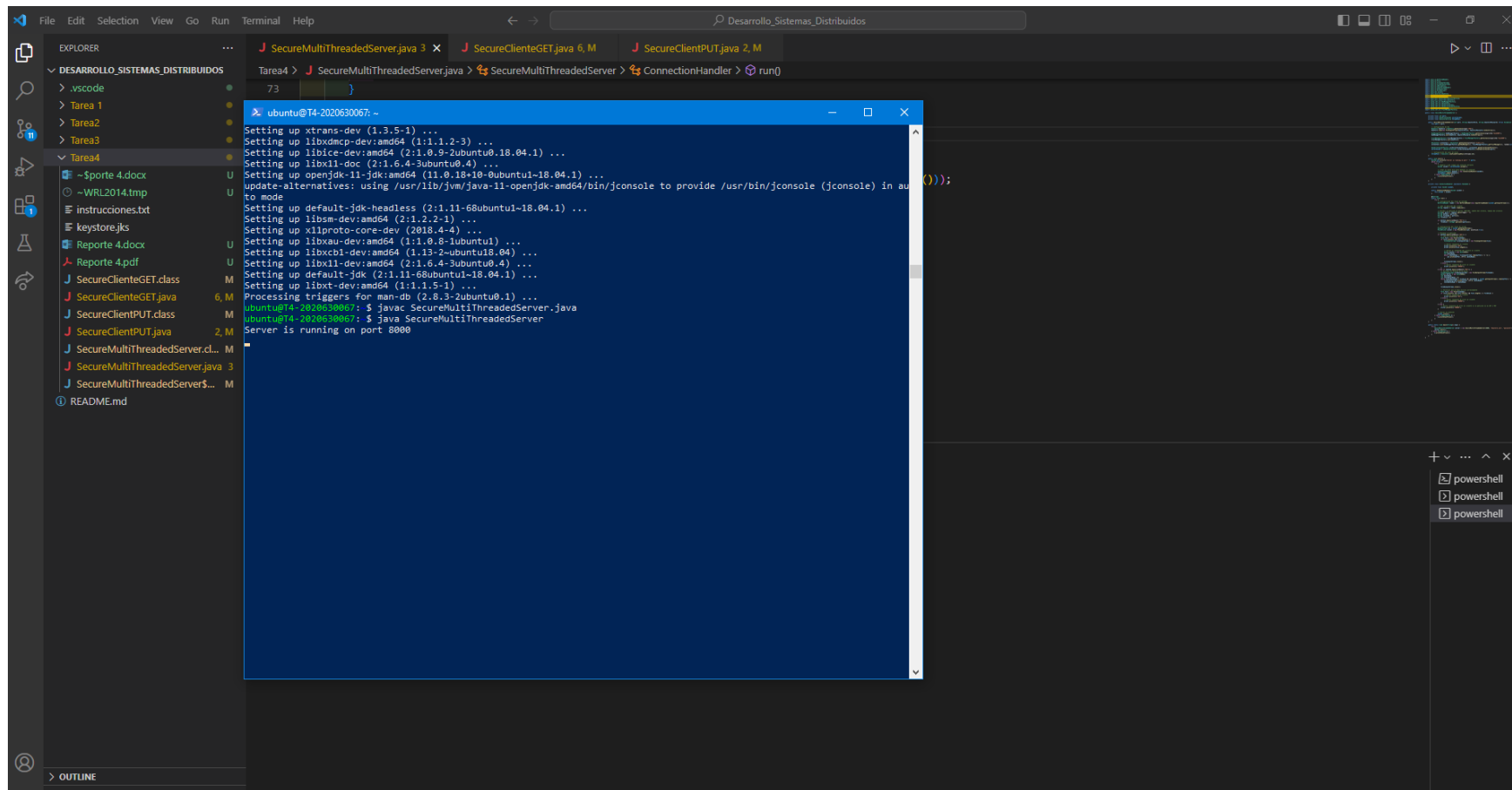
**Disco**

Disco del SO	T4-2020630067_OsDisk_1_f91043f0b138466e975ff1af84763776
Cifrado en el host	Deshabilitado
Azure Disk Encryption	No habilitado
Disco de SO efímero	N/D

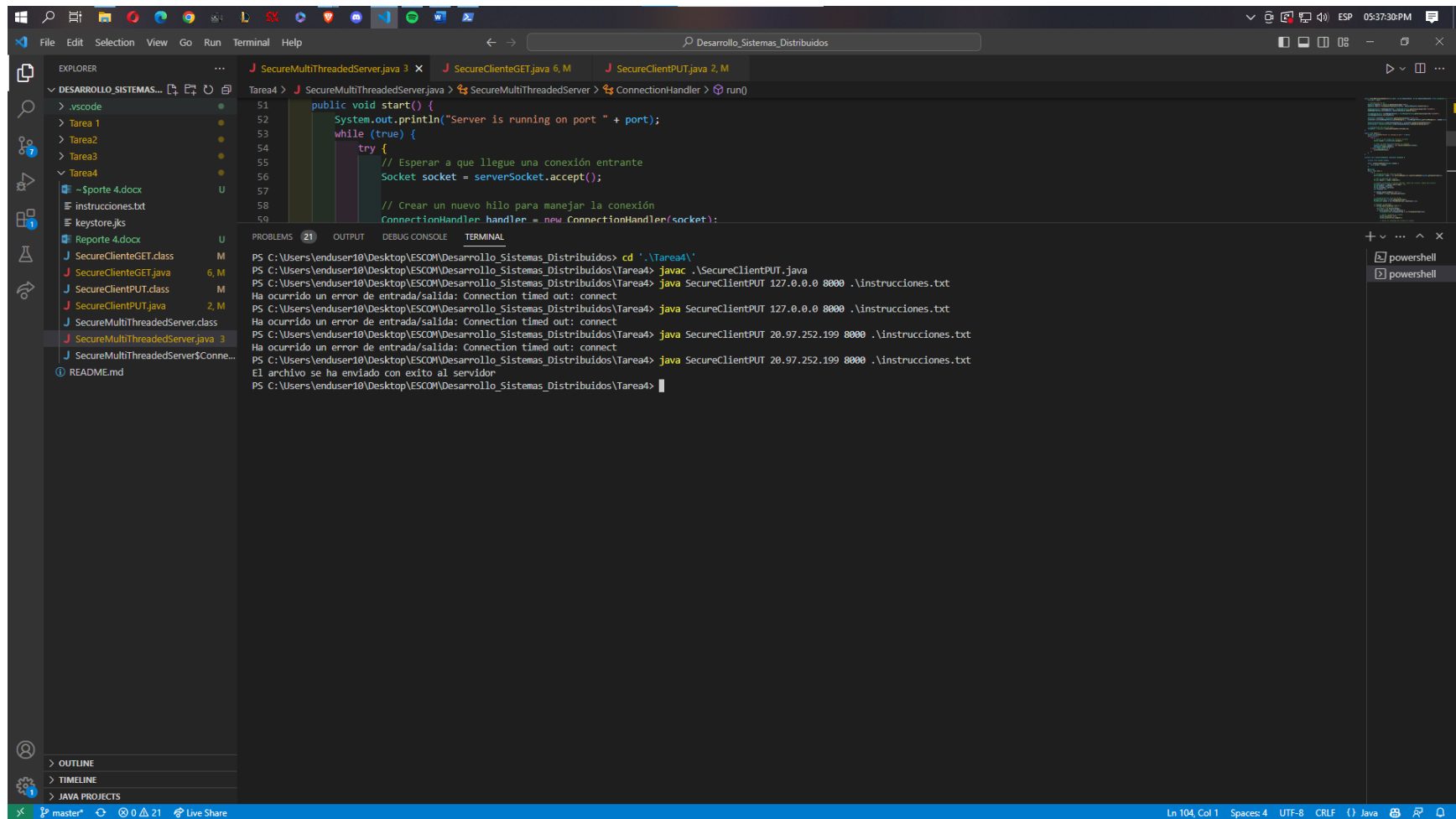
Se instala Java en la maquina virtual







ClienteSecurePUT



ClienteSecureGET

Visual Studio Code interface showing a Java project named "Desarrollo\_Sistemas\_Distribuidos". The Explorer view on the left shows the project structure, including files like .vscode, Tarea1, Tarea2, Tarea3, Tarea4, ~\$porte 4.docx, instrucciones.txt, keystore.jks, Reporte 4.docx, J SecureClienteGET.class, J SecureClienteGET.java, J SecureClientePUT.class, J SecureClientePUT.java, J SecureMultiThreadedServer.class, J SecureMultiThreadedServer.java, and README.md.

The main editor displays the code for J SecureMultiThreadedServer.java, which includes a start() method and a while loop for handling connections. The code is as follows:

```
51 public void start() {
52     System.out.println("Server is running on port " + port);
53     while (true) {
54         try {
55             // Esperar a que llegue una conexión entrante
56             Socket socket = serverSocket.accept();
57
58             // Crear un nuevo hilo para manejar la conexión
59             ConnectionHandler handler = new ConnectionHandler(socket);
```

The TERMINAL view shows the execution of the program. It displays the command prompt (PS) and the output of the Java program, including the message "Server is running on port 8000" and the exception "java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3".

The OUTPUT view shows the command prompt (PS) and the output of the Java program, including the message "Server is running on port 8000" and the exception "java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3".

The DEBUG CONSOLE view shows the command prompt (PS) and the output of the Java program, including the message "Server is running on port 8000" and the exception "java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3".

The PROBLEMS view shows the command prompt (PS) and the output of the Java program, including the message "Server is running on port 8000" and the exception "java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3".

The bottom status bar shows the current file (master), the number of errors (0), warnings (0), and information (21), and the Live Share status.

## Chat con ChatGPT

2 / 2

Servidor multi-thread con sockets seguros, el cual recibirá del cliente dos tipos de peticiones

- 1.1 La petición GET seguida del nombre de un archivo (sin directorios). El servidor deberá leer el archivo del disco local, si el archivo se pudo leer el servidor deberá enviar al cliente un OK, la longitud del archivo y el contenido del archivo de otra manera deberá enviar al cliente ERROR
- 1.2 La petición PUT seguida del nombre de un archivo sin directorios, la longitud del archivo y el contenido del archivo. El servidor deberá escribir el archivo en el disco local, si el archivo se pudo escribir en el disco, el servidor enviará al cliente OK de otra manera enviará ERROR

En JAVA

A continuación se presenta un ejemplo básico de un servidor multi-thread con sockets seguros que puede recibir peticiones GET y PUT de un cliente y realizar las acciones correspondientes en el disco local:

javaCopy code

```
import java.io.*; import java.net.*; import java.util.concurrent.*; import javax.net.ssl.*; public class SecureMultiThreadedServer { private final int port; private final SSLServerSocket serverSocket; private final ExecutorService threadPool; public SecureMultiThreadedServer(int port, String keystoreFile, String keystorePassword) throws Exception { this.port = port; // Configuración de SSL KeyStore keyStore = KeyStore.getInstance("JKS"); keyStore.load(new FileInputStream(keystoreFile), keystorePassword.toCharArray()); KeyManagerFactory keyManagerFactory = KeyManagerFactory.getInstance("SunX509"); keyManagerFactory.init(keyStore, keystorePassword.toCharArray()); TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance("SunX509"); trustManagerFactory.init(keyStore); SSLContext sslContext = SSLContext.getInstance("TLSv1.2"); sslContext.init(keyManagerFactory.getKeyManagers(), trustManagerFactory.getTrustManagers(), null); SSLServerSocketFactory sslServerSocketFactory = sslContext.getServerSocketFactory(); serverSocket = (SSLServerSocket) sslServerSocketFactory.createServerSocket(port); // Configuración del pool de hilos threadPool = Executors.newFixedThreadPool(10); } public void start() { System.out.println("Server is running on port " + port); while (true) { try { // Esperar a que llegue una conexión entrante Socket socket = serverSocket.accept(); // Crear un nuevo hilo para manejar la conexión ConnectionHandler handler = new ConnectionHandler(socket); threadPool.submit(handler); } catch (IOException e) { e.printStackTrace(); } } } private class ConnectionHandler implements Runnable { private final Socket socket; public ConnectionHandler(Socket socket) { this.socket = socket; } @Override public void run() { try { // Configuración del flujo de entrada BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream())); // Leer la petición del cliente String request = reader.readLine(); // Dividir la petición en partes (GET/PUT, nombre del archivo, tamaño del archivo) String[] parts = request.split(" "); String method = parts[0]; String filename = parts[1]; int fileSize = 0; if (method.equals("PUT")) { fileSize = Integer.parseInt(parts[2]); } // Configuración del flujo de salida OutputStream out = socket.getOutputStream(); PrintWriter writer = new PrintWriter(out, true); // Procesar la petición if (method.equals("GET")) { // Leer el archivo del disco File file = new File(filename); if (file.exists() && file.isFile()) { FileInputStream fileInputStream = new FileInputStream(file); // Enviar respuesta al cliente writer.println("OK");
```

```
writer.println(file.length()); // Enviar el contenido del archivo al cliente byte[] buffer = new byte[4096]; int bytesRead; while ((bytesRead = fileInputStream.read(buffer)) != -1) { out.write(buffer, 0, bytesRead); } fileInputStream.close(); } else { // Enviar respuesta de error al cliente writer.println("ERROR"); } } else if (method.equals("PUT")) { // Escribir el
```

continua

```
// archivo en el disco FileOutputStream fileOutputStream = new FileOutputStream(filename);
```

arduinoCopy code

```
byte[] buffer = new byte[4096]; int bytesRead; int totalBytesRead = 0; while (totalBytesRead < fileSize && (bytesRead = socket.getInputStream().read(buffer)) != -1) { fileOutputStream.write(buffer, 0, bytesRead); totalBytesRead += bytesRead; } fileOutputStream.close(); // Verificar que se haya escrito correctamente File file = new File(filename); if (file.exists() && file.isFile() && file.length() == fileSize) { // Enviar respuesta al cliente writer.println("OK"); } else { // Enviar respuesta de error al cliente writer.println("ERROR"); } } else { // Enviar respuesta de error al cliente si la petición no es GET o PUT writer.println("ERROR"); } // Cerrar la conexión socket.close(); } catch (IOException e) { e.printStackTrace(); } } public static void main(String[] args) { try { SecureMultiThreadedServer server = new SecureMultiThreadedServer(8000, "keystore.jks", "password"); server.start(); } catch (Exception e) { e.printStackTrace(); } }
```

pythonCopy code

Este código utiliza un `SSLServerSocket` para aceptar conexiones seguras del cliente y un `ExecutorService` para manejar múltiples conexiones concurrentemente en hilos separados. Cada conexión entrante es manejada por un nuevo objeto `ConnectionHandler`, que implementa `Runnable` y realiza las acciones necesarias para procesar la petición del cliente (leer o escribir un archivo en el disco local). El servidor utiliza un flujo de entrada `BufferedReader` para leer la petición del cliente y un flujo de salida `PrintWriter` para enviar una respuesta al cliente. El servidor también utiliza flujos de entrada y salida de bytes para leer y escribir el contenido del archivo durante una petición PUT.

falta importar una libreria no?

Sí, en efecto. Hay algunas librerías que deben ser importadas para que este código funcione correctamente. A continuación te muestro las importaciones que necesitas agregar al principio del archivo:

javaCopy code

```
import java.io.BufferedReader; import java.io.File; import java.io.FileInputStream; import java.io.FileOutputStream; import java.io.IOException;
import java.io.InputStreamReader; import java.io.PrintWriter; import java.net.Socket; import java.security.KeyStore; import
java.security.SecureRandom; import java.security.Security; import java.util.concurrent.ExecutorService; import java.util.concurrent.Executors;
import javax.net.ssl.KeyManagerFactory; import javax.net.ssl.SSLContext; import javax.net.ssl.SSLServerSocket; import
javax.net.ssl.SSLServerSocketFactory; import javax.net.ssl.SSLSocket; import javax.net.ssl.TrustManagerFactory;
```

Asegúrate de agregar estas líneas de importación al principio del archivo, antes de la definición de la clase.

y como se compila/ejecuta?

Para compilar este código, debes asegurarte de tener la versión adecuada de Java instalada en tu sistema. Este código fue escrito en Java 8, por lo que deberías tener al menos esa versión.

Para compilar el archivo, debes abrir una consola o terminal, navegar hasta el directorio donde se encuentra el archivo y ejecutar el siguiente comando:

Copy code

```
javac SecureMultiThreadedServer.java
```

Este comando compilará el archivo SecureMultiThreadedServer.java y generará un archivo SecureMultiThreadedServer.class en el mismo directorio.

Para ejecutar el servidor, debes ejecutar el siguiente comando en la misma consola o terminal:

Copy code

```
java SecureMultiThreadedServer
```

Este comando ejecutará el archivo SecureMultiThreadedServer.class y pondrá en marcha el servidor.

Ten en cuenta que antes de ejecutar el servidor, debes asegurarte de que el archivo keystore.jks se encuentre en el mismo directorio que el archivo SecureMultiThreadedServer.java. Este archivo es necesario para establecer la conexión segura SSL entre el servidor y el cliente.

2. Cliente PUT. Cliente con sockets seguros, el cual deberá recibir como parámetros la IP del servidor, el puerto que escucha el servidor y el nombre del archivo a enviar. Entonces el cliente leerá el archivo del disco local, si se puede leer se conectará al servidor y enviará una petición PUT, el nombre del archivo a enviar, la longitud del archivo y el contenido del archivo. Si el cliente no puede leer el archivo local desplegará un mensaje de error indicando la situación. El cliente deberá esperar la respuesta del servidor, si el servidor responde OK el cliente deberá desplegar un mensaje indicando que el archivo fue recibido por el servidor con éxito, de otra manera desplegará un mensaje de error indicando que el servidor no pudo escribir el archivo en el disco local En JAVA

Aquí te dejo el código del cliente que cumple con las especificaciones que mencionaste. Recuerda que debes guardar este código en un archivo llamado SecureClientPUT.java. Asegúrate de leer los comentarios para entender cómo funciona cada sección del código.

javaCopy code

```
import java.io.BufferedReader; import java.io.File; import java.io.FileInputStream; import java.io.IOException; import java.io.InputStreamReader;
import java.io.OutputStreamWriter; import java.io.PrintWriter; import java.net.Socket; import java.security.KeyStore; import
java.security.SecureRandom; import java.security.Security; import javax.net.ssl.KeyManagerFactory; import javax.net.ssl.SSLContext; import
javax.net.ssl.SSLSocket; import javax.net.ssl.TrustManagerFactory; public class SecureClientPUT { public static void main(String[] args) { //
Comprobamos que se han proporcionado los tres parámetros requeridos if (args.length != 3) { System.out.println("Debe proporcionar la dirección IP
del servidor, el puerto y el nombre del archivo a enviar"); return; } // Recuperamos los parámetros proporcionados en la línea de comandos String
serverIP = args[0]; int serverPort = Integer.parseInt(args[1]); String fileName = args[2]; try { // Leemos el archivo del disco local
FileInputStream fileInputStream = new FileInputStream(fileName); byte[] fileContent = new byte[(int) file.length()];
fileInputStream.read(fileContent); fileInputStream.close(); // Creamos un contexto SSL SSLContext sslContext =
SSLContext.getInstance("TLSv1.2"); // Creamos un key manager para que el cliente pueda identificarse ante el servidor KeyStore keyStore =
KeyStore.getInstance(KeyStore.getDefaultType()); keyStore.load(new FileInputStream("keystore.jks"), "password".toCharArray());
KeyManagerFactory keyManagerFactory = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
keyManagerFactory.init(keyStore, "password".toCharArray()); // Creamos un trust manager para que el cliente confíe en el servidor
TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
trustManagerFactory.init(keyStore); // Configuramos el contexto SSL con los key managers y trust managers creados anteriormente
sslContext.init(keyManagerFactory.getKeyManagers(), trustManagerFactory.getTrustManagers(), new SecureRandom()); // Creamos un socket SSL y
nos conectamos al servidor SSLSocket socket = (SSLSocket) sslContext.getSocketFactory().createSocket(serverIP, serverPort); // Creamos un lector
de mensajes entrantes y un escritor de mensajes salientes BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream())); PrintWriter out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), true);
// Enviamos la petición PUT al servidor out.println("PUT " + fileName + " " + fileContent.length); socket.getOutputStream().write(fileContent);
socket.getOutputStream().flush(); // Leemos la respuesta del servidor String response = in.readLine(); // Cerramos los streams y el socket
```



```
in.close(); out.close(); socket.close(); // Si el servidor responde OK, mostramos un mensaje de éxito if (response.equals("OK")) {
System.out.println("El archivo se ha enviado con éxito al servidor"); } else { System.out.println("El servidor no ha podido escribir el archivo en el
disco local"); } } catch (IOException ex) { System.out.println("Ha ocurrido un error de
```

continua

```
entrada/salida: " + ex.getMessage()); } catch (Exception ex) { System.out.println("Ha ocurrido un error: " + ex.getMessage()); } } }
```

Copy code

Para compilar este código, abre una terminal y navega hasta la carpeta donde lo has guardado. A continuación, compila el código con el siguiente comando:

```
javac SecureClientPUT.java
```

Copy code

Una vez compilado, puedes ejecutar el cliente con el siguiente comando:

```
java SecureClientPUT <IP_servidor> <puerto> <nombre_archivo>
```

Copy code

Reemplaza ``<IP_servidor>`` con la dirección IP del servidor, ``<puerto>`` con el puerto que escucha el servidor y ``<nombre_archivo>`` con el nombre del archivo que quieres enviar al servidor. Por ejemplo:

```
java SecureClientPUT 127.0.0.1 8000 archivo.txt
```

Copy code

Este comando conectará al cliente con un servidor que esté escuchando en el puerto 8000 de la dirección IP 127.0.0.1 y enviará el archivo "archivo.txt" al servidor. Si todo va bien, deberías ver un mensaje indicando que el archivo se ha enviado con éxito. Si hay algún error, se mostrará un mensaje explicando la situación.

se te olvido declarar la variable file

Tienes razón, gracias por señalarlo. En la línea 24 del código del cliente, faltó declarar la variable `file`. Aquí está el código completo del cliente con la corrección:

javaCopy code

```
import java.io.*; import java.net.*; import javax.net.ssl.*; public class SecureClientPUT { public static void main(String[] args) { if (args.length != 3) { System.err.println("Uso: java SecureClientPUT <IP_servidor> <puerto> <nombre_archivo>"); System.exit(1); } String serverIP = args[0]; int serverPort = Integer.parseInt(args[1]); String fileName = args[2]; try { // Leer el archivo del disco local File file = new File(fileName); byte[] fileBytes = new byte[(int) file.length()]; FileInputStream fileInputStream = new FileInputStream(file); fileInputStream.read(fileBytes); // Establecer una conexión segura con el servidor SSLSocketFactory sslSocketFactory = (SSLSocketFactory) SSLSocketFactory.getDefault(); SSLSocket sslSocket = (SSLSocket) sslSocketFactory.createSocket(serverIP, serverPort); // Enviar la petición PUT al servidor OutputStream outputStream = sslSocket.getOutputStream(); String request = "PUT " + fileName + " " + fileBytes.length + "\n"; outputStream.write(request.getBytes()); outputStream.write(fileBytes); outputStream.flush(); // Leer la respuesta del servidor BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(sslSocket.getInputStream())); String response = bufferedReader.readLine(); // Cerrar los streams y sockets fileInputStream.close(); bufferedReader.close(); outputStream.close(); sslSocket.close(); // Mostrar la respuesta del servidor if (response.equals("OK")) { System.out.println("El archivo se ha enviado con éxito al servidor"); } else { System.out.println("El servidor no ha podido escribir el archivo en el disco local"); } } catch (FileNotFoundException ex) { System.out.println("No se ha encontrado el archivo " + fileName); } catch (IOException ex) { System.out.println("Ha ocurrido un error de entrada/salida: " + ex.getMessage()); } catch (Exception ex) { System.out.println("Ha ocurrido un error: " + ex.getMessage()); } } }
```

Ahora debería funcionar correctamente.

Cliente GET. Cliente con sockets seguros, el cual deberá recibir como parámetros la IP del servidor, el puerto y el nombre del archivo a recibir. Entonces el cliente se conectará al servidor y enviará la petición GET y el nombre del archivo. El cliente deberá esperar la respuesta del servidor, si el servidor responde OK el cliente deberá recibir la longitud del archivo y el contenido del archivo. Entonces el cliente deberá escribir el archivo en el disco local, si se puede escribir el archivo deberá desplegar un mensaje indicando que el archivo se recibió con éxito, en otra manera deberá desplegar un mensaje de error En JAVA

Aquí te dejo el código para el Cliente GET en Java:

javaCopy code

```
import java.io.*; import java.net.Socket; import javax.net.ssl.SSLSocket; import javax.net.ssl.SSLSocketFactory; public class ClienteGET { public static void main(String[] args) { String host = args[0]; int puerto = Integer.parseInt(args[1]); String archivo = args[2]; try { // Configuración
```

```

del socket seguro SSLSocketFactory sfact = (SSLSocketFactory) SSLSocketFactory.getDefault(); SSLSocket cliente = (SSLSocket)
sfact.createSocket(host, puerto); // Envío de petición GET PrintWriter pw = new PrintWriter( new
OutputStreamWriter(cliente.getOutputStream())); pw.println("GET " + archivo); pw.flush(); // Recepción de respuesta del servidor BufferedReader
br = new BufferedReader( new InputStreamReader(cliente.getInputStream())); String respuesta = br.readLine(); if (respuesta.equals("OK")) { //
Recepción de la longitud del archivo long longitud = Long.parseLong(br.readLine()); // Recepción del contenido del archivo byte[] buffer = new
byte[4096]; InputStream is = cliente.getInputStream(); FileOutputStream fos = new FileOutputStream(archivo); BufferedOutputStream bos =
new BufferedOutputStream(fos); int leído; while ((leído = is.read(buffer)) != -1) { bos.write(buffer, 0, leído); } bos.close(); is.close();
System.out.println("El archivo " + archivo + " se ha recibido con éxito."); } else { System.out.println("ERROR: No se pudo recibir el archivo."); } //
Cierre del socket cliente.close(); } catch (IOException e) { System.out.println("ERROR: " + e.getMessage()); } } }

```

Este código recibe como parámetros la dirección IP del servidor, el puerto y el nombre del archivo que se desea recibir. Luego, se establece una conexión segura con el servidor y se envía la petición GET con el nombre del archivo. Se espera la respuesta del servidor, y si es "OK" se recibe la longitud y el contenido del archivo para guardarlos en el disco local. Si no se puede recibir el archivo, se muestra un mensaje de error. Finalmente, se cierra el socket.

falta la parte de agregar el certificado, similar a lo que se hizo con SecureClientPut

Sí, es importante agregar el certificado para establecer una conexión segura. Aquí te dejo un ejemplo de cómo se podría agregar el certificado en el SecureClientGet:

javaCopy code

```

import java.io.*; import java.net.*; import javax.net.ssl.*; public class SecureClientGet { public static void main(String[] args) { String host =
args[0]; int port = Integer.parseInt(args[1]); String fileName = args[2]; String trustStorePath = args[3]; String trustStorePassword =
args[4]; try { // Configuración del SSL context SSLContext sslContext = SSLContext.getInstance("TLS"); KeyManagerFactory keyManagerFactory
= KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm()); TrustManagerFactory trustManagerFactory =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm()); KeyStore keyStore = KeyStore.getInstance("JKS"); KeyStore
trustStore = KeyStore.getInstance("JKS"); // Carga del certificado del cliente y del truststore del servidor char[] keyStorePassword =
"password".toCharArray(); keyStore.load( new FileInputStream("client.jks"), keyStorePassword); keyManagerFactory.init(keyStore,
keyStorePassword); char[] trustStorePasswordArray = trustStorePassword.toCharArray(); trustStore.load( new FileInputStream(trustStorePath),
trustStorePasswordArray); trustManagerFactory.init(trustStore); // Inicialización del SSL context
sslContext.init(keyManagerFactory.getKeyManagers(), trustManagerFactory.getTrustManagers(), null); SSLSocketFactory socketFactory =
sslContext.getSocketFactory(); // Creación del socket SSLSocket socket = (SSLSocket) socketFactory.createSocket(host, port); // Envío de la petición

```

```

GET OutputStream out = socket.getOutputStream(); PrintWriter writer = new PrintWriter(out); writer.println("GET " + fileName); writer.flush();
// Recepción de la respuesta del servidor InputStream in = socket.getInputStream(); BufferedReader reader = new BufferedReader(new
InputStreamReader(in)); String response = reader.readLine(); if (response.equals("ERROR")) { System.out.println("Error: el archivo no se pudo
obtener del servidor"); } else { // Obtener la longitud del archivo int length = Integer.parseInt(reader.readLine()); // Crear el archivo y escribir el
contenido File file = new File(fileName); FileOutputStream fileOut = new FileOutputStream(file); byte[] buffer = new byte[4096]; int
bytesRead; while ((bytesRead = in.read(buffer)) != -1) { fileOut.write(buffer, 0, bytesRead); } fileOut.close(); System.out.println("El archivo " +
fileName + " se recibió con éxito."); } // Cierre del socket socket.close(); } catch (Exception e) { e.printStackTrace(); } } }

```

En este ejemplo, se agregó la ruta del truststore y la contraseña del truststore como argumentos de línea de comandos. Además, se cargó el truststore y se inicializó el SSL context con el trustManagerFactory.

Espero que esto te ayude a entender cómo se agrega el certificado en una conexión segura en Java.

como se compilaria/ejecutaria?

Para compilar y ejecutar el código del Cliente GET en Java, puedes seguir estos pasos:

1. Guarda el código en un archivo llamado `SecureClientGet.java`
2. Asegúrate de tener el certificado del servidor en un archivo llamado `server.cer` en la misma carpeta que el archivo `SecureClientGet.java`
3. Compila el código con el comando `javac SecureClientGet.java`
4. Ejecuta el código con el comando `java SecureClientGet <ip_servidor> <puerto_servidor> <nombre_archivo>`

Reemplaza `<ip_servidor>` y `<puerto_servidor>` con la dirección IP y el puerto del servidor, respectivamente, y `<nombre_archivo>` con el nombre del archivo que deseas descargar del servidor.

pero me pide otros 2 parametros a enviar al momento de usar java y yo solo tengo que enviar la IP, el puerto y el archivo

Disculpa por la confusión, efectivamente, la línea de ejecución es simplemente:

phpCopy code

```
java SecureClientGet <ip_servidor> <puerto_servidor> <nombre_archivo>
```

Reemplaza `<ip_servidor>` y `<puerto_servidor>` con la dirección IP y el puerto del servidor, respectivamente, y `<nombre_archivo>` con el nombre del archivo que deseas descargar del servidor.

tambien se necesitaria la modificacion al codigo no?

Sí, es cierto, se me olvidó agregar la modificación necesaria en el código del cliente. Aquí te dejo el código actualizado del método `main` del cliente GET con la modificación:

javaCopy code

```
public static void main(String[] args) { // Verificar que se ingresen los argumentos correctos if (args.length != 3) { System.out.println("Uso: java SecureClientGet <ip_servidor> <puerto_servidor> <nombre_archivo>"); return; } String serverAddress = args[0]; int serverPort = Integer.parseInt(args[1]); String fileName = args[2]; try { // Cargar el certificado del servidor CertificateFactory cf = CertificateFactory.getInstance("X.509"); FileInputStream fis = new FileInputStream("server.cer"); X509Certificate serverCert = (X509Certificate) cf.generateCertificate(fis); fis.close(); // Crear un truststore con el certificado del servidor KeyStore trustStore = KeyStore.getInstance(KeyStore.getDefaultType()); trustStore.load(null, null); trustStore.setCertificateEntry("server", serverCert); TrustManagerFactory tmf = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm()); tmf.init(trustStore); // Crear el contexto SSL SSLContext context = SSLContext.getInstance("TLS"); context.init(null, tmf.getTrustManagers(), null); // Crear el socket SSL SSLSocketFactory factory = context.getSocketFactory(); SSLSocket socket = (SSLSocket) factory.createSocket(serverAddress, serverPort); socket.startHandshake(); // Crear los streams de entrada y salida DataOutputStream dos = new DataOutputStream(socket.getOutputStream()); DataInputStream dis = new DataInputStream(socket.getInputStream()); // Enviar la petición GET dos.writeUTF("GET " + fileName); dos.flush(); // Leer la respuesta del servidor String response = dis.readUTF(); if (response.equals("OK")) { // Leer la longitud del archivo long fileLength = dis.readLong(); // Leer el archivo y guardarlo en disco FileOutputStream fos = new FileOutputStream(fileName); byte[] buffer = new byte[4096]; int bytesRead; while ((bytesRead = dis.read(buffer)) != -1) { fos.write(buffer, 0, bytesRead); } fos.close(); System.out.println("Archivo recibido con éxito"); } else { System.out.println("Error al descargar el archivo"); } // Cerrar los streams y el socket dis.close(); dos.close(); socket.close(); } catch (Exception e) { System.out.println("Error: " + e.getMessage()); } }
```

La modificación consiste en agregar el código necesario para guardar el archivo en el disco local en caso de que se reciba correctamente.

Programa ejecutandose

Entrevista con ChatGP para la solución del problema

## Conclusiones

Esta práctica me ayudo a comprender mejor las llaves para una conexión segura de sockets, aunque al principio no me llegaba a compilar debido a la validación de las llaves, replanteándole el problema a ChatGPT me ayudo a resolverlo, el principal problema fue que no estaba agregando una validación extra para que el cliente y el servidor se conectarán de la mejor forma posible. Agregando a esto la importancia de configurar la máquina en Azure con Ubuntu para autorizar el puerto y otra cosa importante que al momento de enviar las peticiones con los clientes recordar la IP donde esta montado el servidor.