



TECNOLÓGICO  
NACIONAL DE MÉXICO



# TECNOLÓGICO NACIONAL DE MÉXICO

## INSTITUTO TECNOLÓGICO DE TLAXIACO

### ASIGNATURA:

Seguridad y virtualización

### DOCENTE:

Ing. Edward Osorio Salinas

### INTEGRANTES DEL EQUIPO:

Luz Arleth López Bautista 21620036

Saúl López Bautista 21620073

### ACTIVIDAD:

**PRÁCTICA 1:** CONTRASEÑAS Y CERTIFICADOS

### CARRERA:

INGENIERIA EN SISTEMAS COMPUTACIONALES

### SEPTIMO SEMESTRE

### GRUPO:

7US

Heroica ciudad de Tlaxiaco. A 29 de agosto del 2024.

## INTRODUCCIÓN

En el entorno actual de desarrollo y administración de sistemas, la seguridad y la eficiencia son esenciales para proteger la integridad y el acceso a los recursos digitales. En este reporte, abordamos cuatro ejercicios prácticos que cubren aspectos cruciales de la seguridad informática y la gestión de certificados.

El primer ejercicio se centra en la creación de un programa en Java para validar contraseñas seguras. Dado que las contraseñas son una de las principales líneas de defensa contra el acceso no autorizado, es crucial que cumplan con ciertos criterios de seguridad. El programa implementado verifica que las contraseñas tengan una longitud mínima, incluyan caracteres diversos y no contengan espacios o caracteres repetitivos en exceso.

En el segundo ejercicio, desarrollamos un generador de contraseñas seguras. Generar contraseñas robustas automáticamente es una práctica recomendada para evitar contraseñas débiles que puedan ser vulnerables a ataques. Este generador cumple con los mismos criterios de seguridad establecidos en el primer ejercicio.

El tercer ejercicio aborda la configuración de un entorno de desarrollo seguro mediante la creación y el uso de certificados SSH. Los certificados SSH son fundamentales para autenticar y asegurar las conexiones entre el cliente y el servidor. En este caso, el proceso incluye la generación de claves SSH, su integración en GitHub y la clonación de un repositorio utilizando el protocolo SSH.

Finalmente, el cuarto ejercicio trata sobre la creación de un certificado SSL autofirmado con una validez de 365 días y su configuración en un servidor web local. Los certificados SSL son esenciales para cifrar la comunicación entre el servidor y el cliente, garantizando la confidencialidad de los datos transmitidos. Se realiza una petición GET al servidor para verificar que el certificado SSL está funcionando correctamente.

Estos ejercicios no solo proporcionan una comprensión práctica de la seguridad en el desarrollo y la administración de sistemas, sino que también ilustran la importancia de implementar buenas prácticas para proteger la información sensible.

## **Objetivos**

1. **Mejorar la Seguridad en la Programación:** Adquirir habilidades para implementar y validar contraseñas seguras en Java, asegurando que los programas sean resistentes a ataques comunes como fuerza bruta y suplantación de identidad.
2. **Generar Contraseñas Seguras:** Desarrollar herramientas para crear contraseñas robustas, promoviendo la seguridad y protección en sistemas y aplicaciones mediante la aplicación de buenas prácticas de seguridad.
3. **Implementar y Gestionar Claves SSH:** Aprender a generar y gestionar certificados SSH, integrando estos certificados con plataformas como GitHub para garantizar una comunicación segura y autenticada.
4. **Configurar Certificados SSL en Servidores Web:** Adquirir experiencia en la creación y configuración de certificados SSL autofirmados, y verificar su implementación en servidores web locales para asegurar la integridad y confidencialidad de las comunicaciones.

**Ejercicio 1. Crea un programa en Python que permita al usuario ingresar una contraseña y que valide si la contraseña es segura o no. Una contraseña segura debe cumplir con los siguientes criterios basados en las recomendaciones de Google:**

- Tener al menos 8 caracteres.
- Tener al menos una letra mayúscula (AZ).
- Tener al menos una letra minúscula (az).
- Tener al menos un número (0-9).
- Tener al menos un carácter especial ( ! , @ , # , \$ , % , ^ , & , \* , ( , ) , , , , , , , , , , - , \_ = + [ ] { } | \ ; : " ' , . < > / ? ~ `
- No debe contener espacios en blanco.
- No debe tener más de 2 caracteres iguales consecutivos.
- Si la contraseña cumple con los criterios, el programa deberá mostrar un mensaje indicando que la contraseña es segura, de lo contrario, deberá mostrar un mensaje indicando que la contraseña no es segura.

### Paso 1. Paquete e importaciones:

Declaramos el paquete al que pertenece la clase. Los paquetes en Java se utilizan para organizar las clases en directorios.

Posteriormente Importamos la clase Scanner del paquete java.util, que se usa para la entrada de datos desde el teclado y por último importamos la clase Pattern del paquete java.util.regex, que se usa para trabajar con expresiones regulares.

```
1 package programcontraseña;
2
3 import java.util.Scanner;
4 import java.util.regex.Pattern;
5
```

### Ilustración 1 Nombre del paquete e importaciones

## Paso 2. Clase y Método de Validación de Contraseña.

Declaramos una clase pública llamada ProgramContraseña.

Una vez hecho eso definimos un método estático que toma una cadena (String) llamada "contrasena" y devuelve un valor booleano (boolean). Este método valida la contraseña según ciertos criterios:

- **if (contrasena.length() < 8):** Dentro del primer condicional if verificamos que la contraseña tenga al menos 8 caracteres. Si no, muestra un mensaje y retorna false.
- **if (contrasena.contains(" ")):** Verifica que la contraseña no contenga espacios. Si contiene espacios, muestra un mensaje y retorna false.
- **if (!Pattern.compile("[A-Z]").matcher(contrasena).find()):** Utiliza una expresión regular para verificar que la contraseña contenga al menos una letra mayúscula. Si no es así, muestra un mensaje y retorna false.
- **if (!Pattern.compile("(.)\\1\\1").matcher(contrasena).find()):** Verifica que no haya más de dos caracteres consecutivos iguales en la contraseña. Si se encuentran, muestra un mensaje y retorna false.
- **if (!Pattern.compile("[a-z]").matcher(contrasena).find()):** Utiliza una expresión regular para verificar que la contraseña contenga al menos una letra minúscula. Si no es así, muestra un mensaje y retorna false.
- **if (!Pattern.compile("[0-9]").matcher(contrasena).find()):** Verifica que la contraseña contenga al menos un dígito numérico. Si no lo hace, muestra un mensaje y retorna false.
- **if(!Pattern.compile("[!@#\$%+\*^&()\_\*=\*{}|;.:',<>~/]").matcher(contrasena).find()):** Utiliza una expresión regular para verificar que la contraseña contenga al menos un carácter especial. Si no se encuentra, muestra un mensaje y retorna false`.

Si todas las condiciones son satisfactorias, el método retorna true.

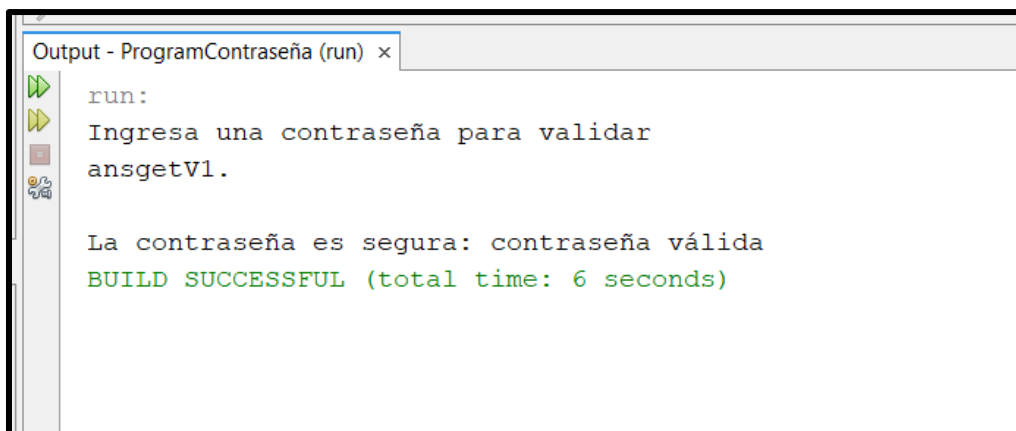


```

9
10 public static void main(String[] args) {
11     Scanner scanner = new Scanner(System.in);
12     System.out.println("Ingresa una contraseña para validar");
13     String contrasena = scanner.nextLine();
14     System.out.println();
15     if (ContrasenaSegura(contrasena)) {
16         System.out.println("La contraseña es segura:contraseña valida");
17     } else {
18         System.out.println("La contraseña no es segura");
19     }
20 }
21
22 }

```

Ilustración 3 Método main, entrada de datos del usuario y llamada al método para validar contraseña



```

Output - ProgramContraseña (run) x
run:
Ingresa una contraseña para validar
ansgetV1.

La contraseña es segura: contraseña válida
BUILD SUCCESSFUL (total time: 6 seconds)

```

Ilustración 4 Ejecución del programa

**Ejercicio 2. Crea un programa que me recomiende una contraseña segura. La contraseña debe cumplir con los criterios de la instrucción anterior.**

**Paso 1.** Utilizamos un bucle do-while para generar contraseñas repetidamente hasta encontrar una que pase la validación de seguridad.

Una vez que se encuentra una contraseña segura, se imprime en la consola.

```
public static void main(String[] args) {
    String contraseña;
    do {
        contraseña = generarContraseña();
    } while (!esSegura(contraseña));

    System.out.println("Contraseña generada: " + contraseña);
    System.out.println(x:"La contraseña es segura.");
}
```

Ilustración 5 Bucle do-while para generar contraseña

**Paso 2.** Definimos después unas cadenas de caracteres para cada tipo de carácter solicitado (mayúsculas, minúsculas, números y caracteres especiales).

- Aseguramos que la contraseña incluya al menos un carácter de cada tipo.
- Utilizamos un bucle para rellenar el resto de la contraseña hasta alcanzar una longitud entre 8 y 12 caracteres.

```
public static String generarContraseña() {
    String mayusculas = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    String minusculas = "abcdefghijklmnopqrstuvwxyz";
    String numeros = "0123456789";
    String caracteresEspeciales = "!@#$%^&*()-_+[]{}|;:',.<.>/?~";
    String todosCaracteres = mayusculas + minusculas + numeros + caracteresEspeciales;

    Random random = new Random();
    StringBuilder contraseña = new StringBuilder();

    // Asegurar que tenga al menos un carácter de cada tipo
    contraseña.append(mayusculas.charAt(random.nextInt(mayusculas.length())));
    contraseña.append(minusculas.charAt(random.nextInt(minusculas.length())));
    contraseña.append(numeros.charAt(random.nextInt(numeros.length())));
    contraseña.append(caracteresEspeciales.charAt(random.nextInt(caracteresEspeciales.length())));

    for (int i = 4; i < 12; i++) {
        contraseña.append(todosCaracteres.charAt(random.nextInt(todosCaracteres.length())));
    }

    return contraseña.toString();
}
```

Ilustración 6 Definición de cadenas de caracteres



**Paso 3.** Nos dirigimos en el método esSegura donde este apartado de este bloque de código verifica si la longitud de la contraseña es menor a 8 caracteres. Si es así, la contraseña se considera insegura y el método devuelve false.

```
public static boolean esSegura(String contraseña) {  
    if (contraseña.length() < 8) {  
        return false;  
    }  
}
```

Ilustración 7 Método para verificar que la contraseña sea segura

**Paso 4.** Después proseguimos a dirigirnos donde están las variables de control. Estas variables booleanas las utilizaremos para verificar si la contraseña contiene al menos una letra mayúscula, una minúscula, un número y un carácter especial.

```
boolean tieneMayuscula = false;  
boolean tieneMinuscula = false;  
boolean tieneNumero = false;  
boolean tieneEspecial = false;
```

Ilustración 8 Variables de control

**Paso 5.** Una vez teniendo las variables de control nos centramos en las **variables para controlar caracteres consecutivos**, las cuales tienen como nombres utilizadas, ultimoCaracter el cual nos almacena el último carácter que se procesó en la iteración anterior del bucle y el otro con el nombre de contadorConsecutivos cuenta cuántos caracteres iguales consecutivos han aparecido.

```
char ultimoCaracter = '\0';  
int contadorConsecutivos = 1;
```

Ilustración 9 Variables para controlar caracteres consecutivos

**Paso 6.** Creamos un ciclo for que nos permitirá recorrer cada carácter de la contraseña uno por uno.

```
for (int i = 0; i < contraseña.length(); i++) {
    char c = contraseña.charAt(i);
```

Ilustración 10 Ciclo for para recorrer la contraseña

**Paso 7.** Agregamos condiciones a nuestro programa que determinara si el carácter actual (c) es una letra mayúscula, una minúscula, un número o un carácter especial. Dependiendo del tipo de carácter, se establece la variable booleana correspondiente en true.

```
if (Character.isUpperCase(c)) {
    tieneMayuscula = true;
} else if (Character.isLowerCase(c)) {
    tieneMinuscula = true;
} else if (Character.isDigit(c)) {
    tieneNumero = true;
} else if ("!@#$%^&*()-_+=[]{}|;:',.<>/?~\".indexOf(c) != -1) {
    tieneEspecial = true;
}
```

Ilustración 11 Condicionales para determinar que la contraseña cumpla con lo que se pide

**Paso 8.** Al final de cada iteración del bucle, se actualiza ultimoCaracter para que sea el carácter actual (c). Esto asegura que en la siguiente iteración se pueda comparar el nuevo carácter con el actual.

```
ultimoCaracter = c;
```

```
}
```

Ilustración 12 Actualización de valor de la variable ultimoCaracter

**Paso 9.** Por ultimo después de recorrer todos los caracteres de la contraseña, este retorno evalúa si la contraseña contiene al menos una letra mayúscula, una minúscula, un número y un carácter especial.

```
return tieneMayuscula && tieneMinuscula && tieneNumero && tieneEspecial;
```

Ilustración 13 Evaluación de contraseña

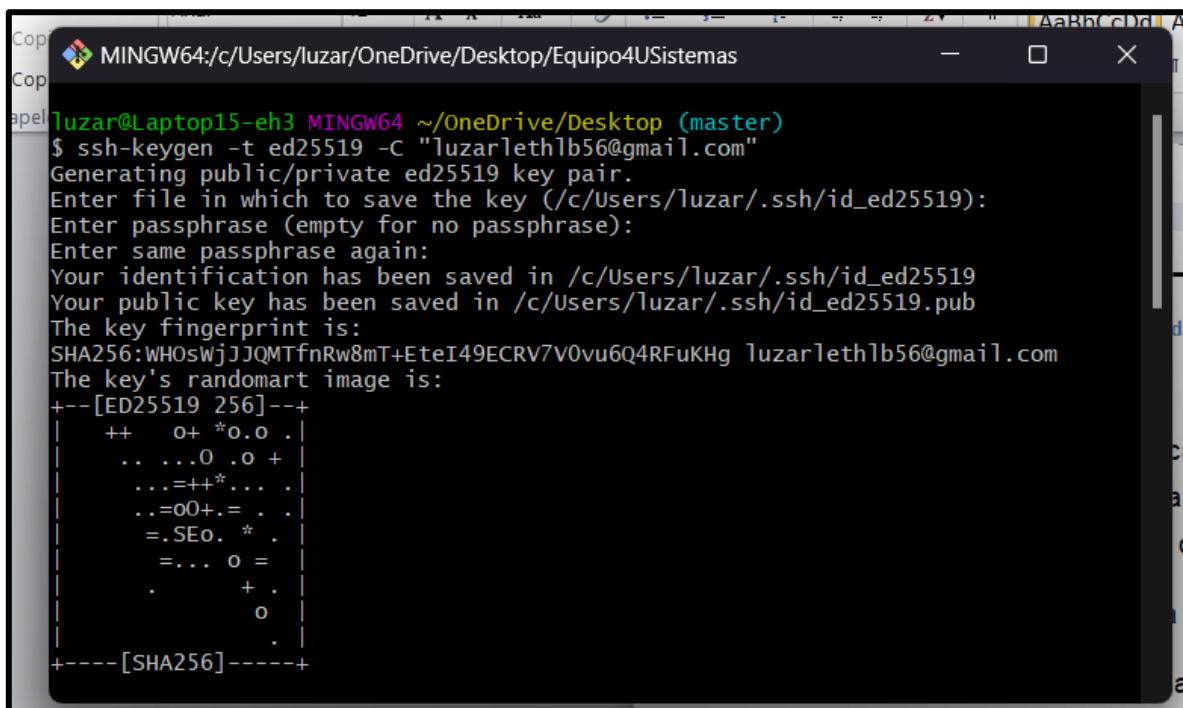
**Ejercicio 3. Crea un certificado SSH, clave pública y clave privada, agrega el certificado SSH a tu cuenta de GitHub y realiza un git clone de un repositorio nuevo utilizando la ruta SSH del repositorio.**

**Paso 1. Generar una nueva clave SSH.**

Ejecutamos el siguiente comando en Git Bash para generar un par de claves SSH:

```
ssh-keygen -t ed25519 -C "luzarlethlb56@gmail.com"
```

Posteriormente guardamos la clave en la ubicación predeterminada por lo que pulsamos enter. Una vez hecho esto nos pedirá añadir una passphrase, por lo que si la deseamos añadir para mayor seguridad la ingresamos y presionamos Enter, caso contrario sino queremos añadirla solo presionamos Enter.

A screenshot of a terminal window titled "MINGW64: c:/Users/luzar/OneDrive/Desktop/Equipo4USistemas". The prompt is "luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)". The command entered is "\$ ssh-keygen -t ed25519 -C 'luzarlethlb56@gmail.com'". The output shows the generation of an ed25519 key pair, saving it to /c/Users/luzar/.ssh/id\_ed25519, and displaying the public key fingerprint and a randomart image.

```
luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)
$ ssh-keygen -t ed25519 -C "luzarlethlb56@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/luzar/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/luzar/.ssh/id_ed25519
Your public key has been saved in /c/Users/luzar/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:WH0sWjJJQMTfnRw8mT+EteI49ECRV7V0vu6Q4RFuKHg luzarlethlb56@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|  ++  o+ *o.o . |
| .. ...O .o + |
| ...=++*... . |
| ..=00+=. = . |
| =.SEo. * . |
| =... o = |
| . + . |
| . o |
| . |
+-----[SHA256]-----+
```

Ilustración 14 Generación de clave SSH

## Paso 2. Añadir la Nueva Clave a GitHub

- Una vez que hayamos generado la nueva clave SSH obtenemos la clave pública, por lo que ejecutamos el siguiente comando para copiar la clave pública al portapapeles:

```
luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)
$ cat ~/.ssh/id_ed25519.pub | clip
luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)
```

Ilustración 15 Comando para copiar la clave

- Nos dirigimos a github e iniciamos sesión en nuestra cuenta, navegamos a “Settings (Configuración)”, luego a “SSH and GPG keys” y pulsamos el botón de “nueva clave SSH.”
- Le damos un título a la clave, en nuestro caso "Mi PC" y pegamos la clave pública en el campo correspondiente.

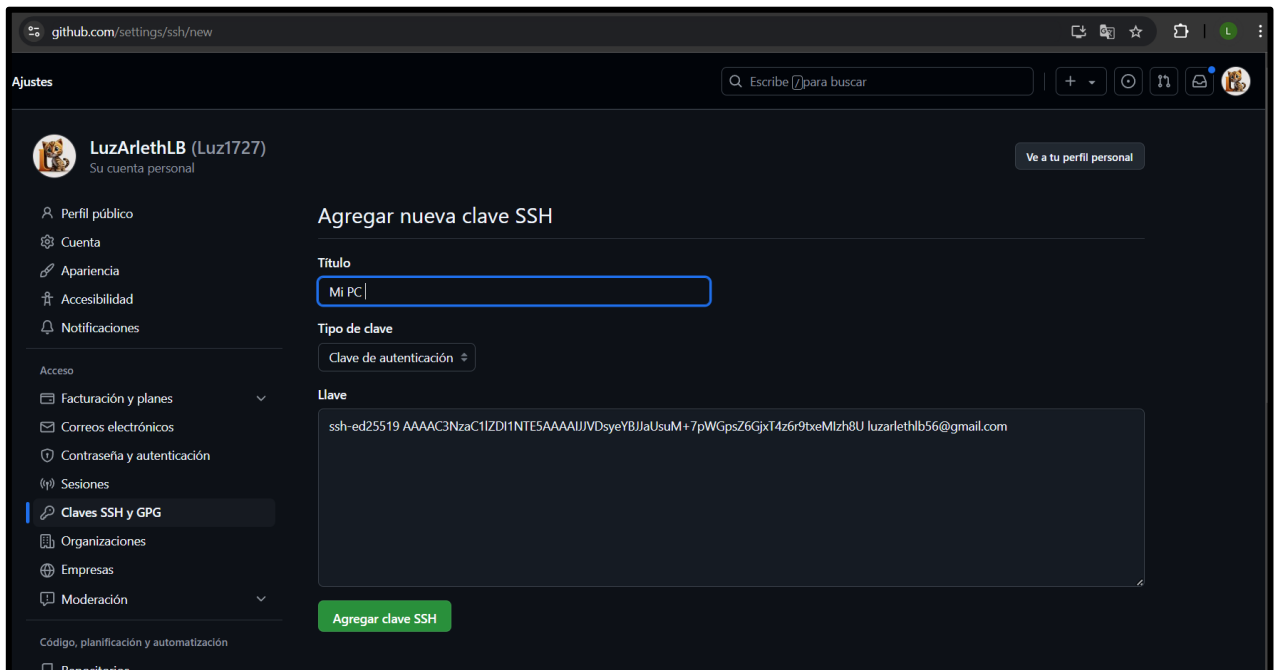


Ilustración 16 Clave SSH en github

- Hacemos clic en Add SSH key para guardar.

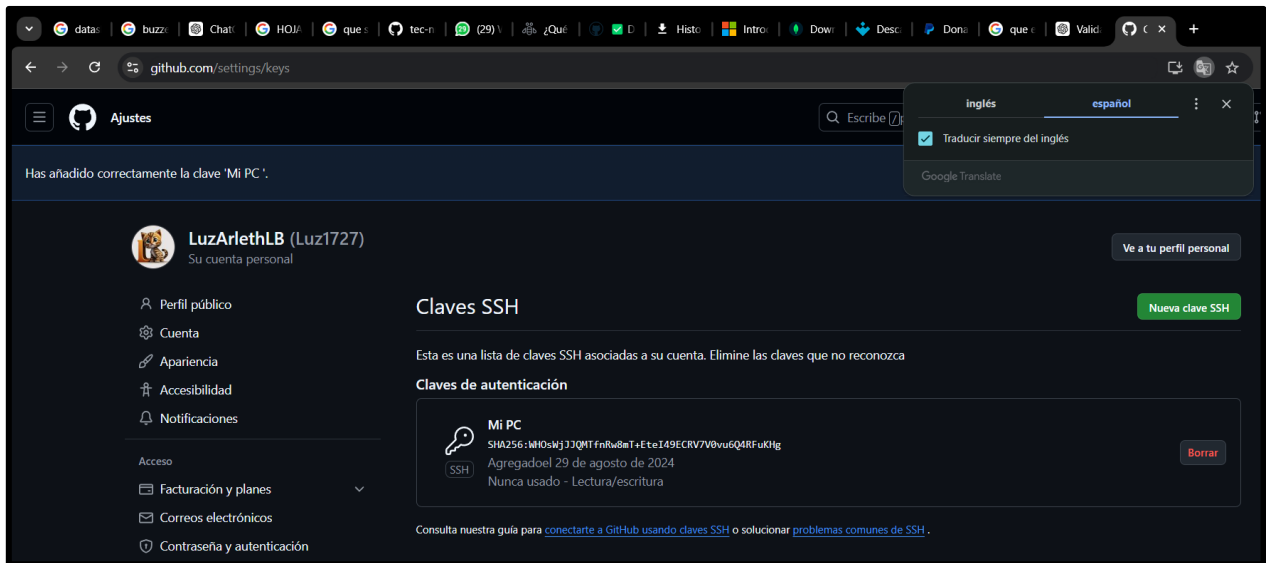


Ilustración 17 Clave añadida correctamente

### 3. Probar la Conexión SSH

Para asegurarnos de que la clave SSH está configurada correctamente y que podemos comunicarnos con GitHub, realizamos una prueba de conexión donde ejecutamos el siguiente comando:

```
luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)
$ ssh -T git@github.com
Hi Luz1727! You've successfully authenticated, but GitHub does not provide shell
access.
```

Ilustración 18 Clave configurada correctamente

Por lo que podemos ver que indica que hemos configurado correctamente la clave SSH y que estamos autenticados con GitHub.

#### Paso 4. Clonar un Repositorio usando SSH.

Nos dirigimos a uno de los repositorios que teníamos ya creado anteriormente, por lo que copiamos la URL SSH del repositorio y lo clonamos con “git clone”.

```
luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)
$ git clone git@github.com:Luz1727/Equipo4USistemas.git
Cloning into 'Equipo4USistemas'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.

luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)
```

Ilustración 19 Repositorio clonado

## Paso 5. Navegar al Repositorio.

- Podemos cambiar el directorio ejecutando el siguiente comando:

```
luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop (master)
$ cd Equipo4USistemas

luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop/Equipo4USistemas (main)
```

Ilustración 20 Repositorio Equipo4USistemas

- Así como también podemos verificar archivos.

```
luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop/Equipo4USistemas (main)
$ ls
Codigo.txt  CodigoFuenteEquipo601.txt

luzar@Laptop15-eh3 MINGW64 ~/OneDrive/Desktop/Equipo4USistemas (main)
```

Ilustración 21 Verificación de archivos

**Ejercicio 4. Crea un certificado SSL autofirmado con una validez de 365 días y añádelo a un servidor web local. Realice una petición GET al servidor web local utilizando curl y muestra el certificado SSL.**

**Paso 1.** Abrimos la terminal de Openssl navegamos hasta la unidad.

Cd"C:/Users/lopez/Documents/Septimosemestre/Seguridadyvirtualizacion/CUART  
OEJERCICIO/"

```
C:\Users\lopez>cd "C:/Users/lopez/Documents/Septimo semestre/Seguridad y virtualizacion/CUARTOEJERCICIO/"
```

Ilustración 22 Terminal Openssl

**Paso 2.** Generamos el Certificado y la clave.

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout myserver.key -out  
myserver.crt

```
C:\Users\lopez\Documents\Septimo semestre\Seguridad y virtualizacion\CUARTOEJERCICIO>openssl req -x509 -nodes -days 365  
-newkey rsa:2048 -keyout myserver.key -out myserver.crt
```

Ilustración 23 Certificado y clave

**Paso 3.** Comenzamos a rellenar varios campos que nos solicitaba para identificar a la entidad que solicita el certificado. Estos campos incluyen el código de país, el nombre del estado o provincia, el nombre de la localidad o ciudad, el nombre de la organización o empresa, la unidad organizativa dentro de la organización, el nombre común, y una dirección de correo electrónico de contacto.

```
Country Name (2 letter code) [AU]:MX  
State or Province Name (full name) [Some-State]:Oaxaca  
Locality Name (eg, city) []:Tlaxiaco  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Tecnologia  
Organizational Unit Name (eg, section) []:Sack junior  
Common Name (e.g. server FQDN or YOUR name) []:Saul  
Email Address []:lopezbautistasaul12@gmail.com
```

Ilustración 24 Identidad del certificado



**Paso 4.** Creamos un nuevo script en python para Verificar la Existencia de Archivos

Si los archivos están en el mismo directorio que el script.

```
from http.server import HTTPServer, SimpleHTTPRequestHandler
import ssl
import os

# Rutas a los archivos de certificado y clave
certfile = 'myserver.crt'
keyfile = 'myserver.key'

# Verificar si los archivos existen
if not os.path.isfile(certfile):
    raise FileNotFoundError(f"Certificado no encontrado: {certfile}")
if not os.path.isfile(keyfile):
    raise FileNotFoundError(f"Clave no encontrada: {keyfile}")

# Configuración del servidor
server_address = ('localhost', 4443)
httpd = HTTPServer(server_address, SimpleHTTPRequestHandler)

# Crear el contexto SSL
context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain(certfile=certfile, keyfile=keyfile)

# Envolver el socket del servidor en el contexto SSL
httpd.socket = context.wrap_socket(httpd.socket, server_side=True)

print("Servidor HTTPS corriendo en https://localhost:4443")
httpd.serve_forever()
```

Ilustración 25 Script de python

**Paso 5.** Ejecutamos dicho Script, como podemos no nos marcó ningún error por lo que quiere decir que los archivos se crearon de forma exitosa.

**Servidor HTTPS corriendo en https://localhost:4443**

Ilustración 26 Ejecución de script

**Paso 6.** Ingresamos al enlace que nos brindó y podemos apreciar dicha interfaz

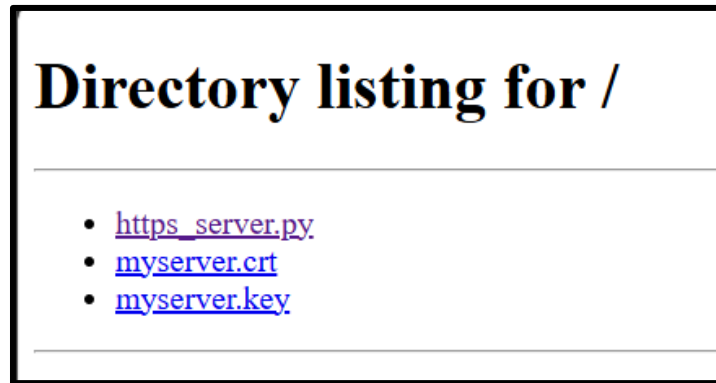


Ilustración 27 Directorio

**Paso 7.** Como último paso utilizamos el siguiente comando

`curl -v -k https://localhost:4443` para obtener respuesta de la verificación de nuestro certificado

```
C:\Users\lopez>curl -v -k https://localhost:4443
Host localhost:4443 was resolved.
IPv6: ::1
IPv4: 127.0.0.1
Trying [::1]:4443...
Trying 127.0.0.1:4443...
Connected to localhost (127.0.0.1) port 4443
schannel: disabled automatic use of client certificate
ALPN: curl offers http/1.1
ALPN: server did not agree on a protocol. Uses default.
using HTTP/1.x
GET / HTTP/1.1
Host: localhost:4443
User-Agent: curl/8.8.0
Accept: */*
```

Ilustración 28 verificación de certificado

```
C:\Users\lopez\Documents\Septimo semestre\Seguridad y virtualizacion\CUARTOEJERCICIO>openssl s_client -connect localhost:4443 -showcerts
Connecting to 127.0.0.1
CONNECTED(00000200)
Can't use SSL_get_servername
depth=0 C=MX, ST=Oaxaca, L=Tlaxiaco, O=Tecnologia, OU=Sack junior, CN=Saul, emailAddress=lopezbautistasaul12@gmail.com
verify error:num=18:self-signed certificate
verify return:1
depth=0 C=MX, ST=Oaxaca, L=Tlaxiaco, O=Tecnologia, OU=Sack junior, CN=Saul, emailAddress=lopezbautistasaul12@gmail.com
verify return:1
---
Certificate chain
0 s:C=MX, ST=Oaxaca, L=Tlaxiaco, O=Tecnologia, OU=Sack junior, CN=Saul, emailAddress=lopezbautistasaul12@gmail.com
i:C=MX, ST=Oaxaca, L=Tlaxiaco, O=Tecnologia, OU=Sack junior, CN=Saul, emailAddress=lopezbautistasaul12@gmail.com
a:PKCS7: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
v:NotBefore: Aug 30 02:57:23 2024 GMT; NotAfter: Aug 30 02:57:23 2025 GMT
```

## **CONCLUSION**

A través de la realización de los ejercicios propuestos, se ha logrado una comprensión integral y práctica de aspectos cruciales en la seguridad informática. El desarrollo de programas en Java para validar contraseñas y generar recomendaciones de contraseñas seguras ha demostrado ser fundamental para la implementación de prácticas robustas en el manejo de credenciales. Estos conocimientos permiten garantizar la protección de sistemas frente a accesos no autorizados y ataques cibernéticos.

La creación y gestión de certificados SSH y SSL, así como su integración en plataformas como GitHub y servidores web locales, han proporcionado habilidades prácticas en la implementación de mecanismos de seguridad esenciales para la comunicación en entornos de red. La experiencia adquirida en la configuración y verificación de certificados asegura que las comunicaciones sean seguras y que la integridad de los datos esté protegida.

En resumen, estos ejercicios no solo han fortalecido la comprensión de conceptos teóricos relacionados con la seguridad, sino que también han proporcionado habilidades prácticas aplicables a escenarios del mundo real, contribuyendo así a una mayor preparación en el campo de la seguridad informática.