TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

UNIDAD 1 INTRODUCCIÓN A LA SEGURID AD- CONTRASEÑAS Y CERTIFICADOS

YAEL DE JESÚS SANTIAGO ORTIZ

JAVIER NOE CRUZ ESPAÑA

KEVIN SÁNCHEZ HERNÁNDEZ

INGENIERIA EN SISTEMAS COMPUTACIONALES

7MO SEMESTRE 7US

Índice

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 (A-Z).
    - Tener al menos una letra minúscula (a-z).
    - 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.

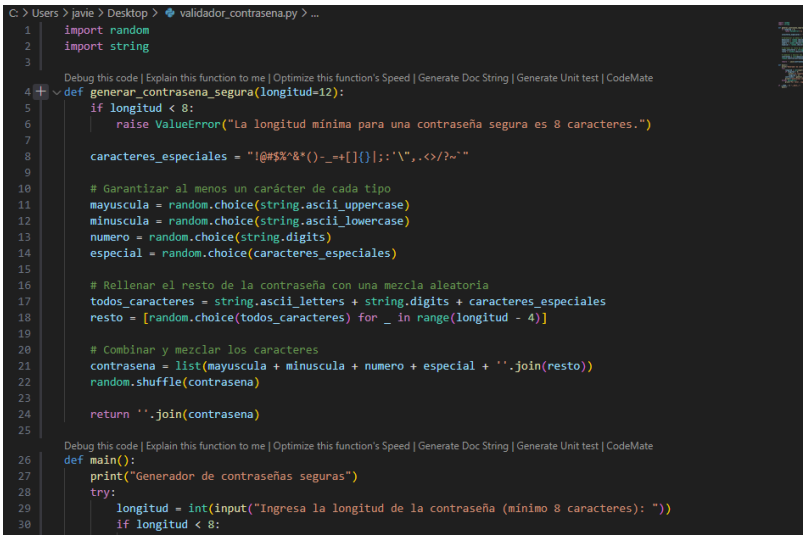
```python
import re

def es_contrasena_segura(contrasena):
    # Verificar longitud mínima
    if len(contrasena) < 8:
        return "La contraseña debe tener al menos 8 caracteres."

    # Verificar al menos una letra mayúscula
    if not any(char.isupper() for char in contrasena):
        return "La contraseña debe tener al menos una letra mayúscula (A-Z)."

    # Verificar al menos una letra minúscula
    if not any(char.islower() for char in contrasena):
        return "La contraseña debe tener al menos una letra minúscula (a-z)."

    # Verificar al menos un número
    if not any(char.isdigit() for char in contrasena):
        return "La contraseña debe tener al menos un número (0-9)."

    # Verificar al menos un carácter especial
    if not any(char in "!@#$%^&*()-_=+[]{}|;:'\",.<>/?~`" for char in contrasena):
        return "La contraseña debe tener al menos un carácter especial (!, @, #, etc.)."

    # Verificar que no contenga espacios en blanco
    if any(char.isspace() for char in contrasena):
        return "La contraseña no debe contener espacios en blanco."

    # Verificar que no haya más de 2 caracteres iguales consecutivos
    if re.search(r'(.)\1{2,}', contrasena):
        return "La contraseña no debe tener más de 2 caracteres iguales consecutivos."

    return "La contraseña es segura."

def main():
    print("Validador de contraseñas seguras")
    contrasena = input("Ingresa tu contraseña: ")
    resultado = es_contrasena_segura(contrasena)
    print(resultado)

if __name__ == "__main__":
    main()
```

```
Validador de contraseñas seguras
Ingresa tu contraseña: noecruz16798
La contraseña debe tener al menos una letra mayúscula (A-Z).
```

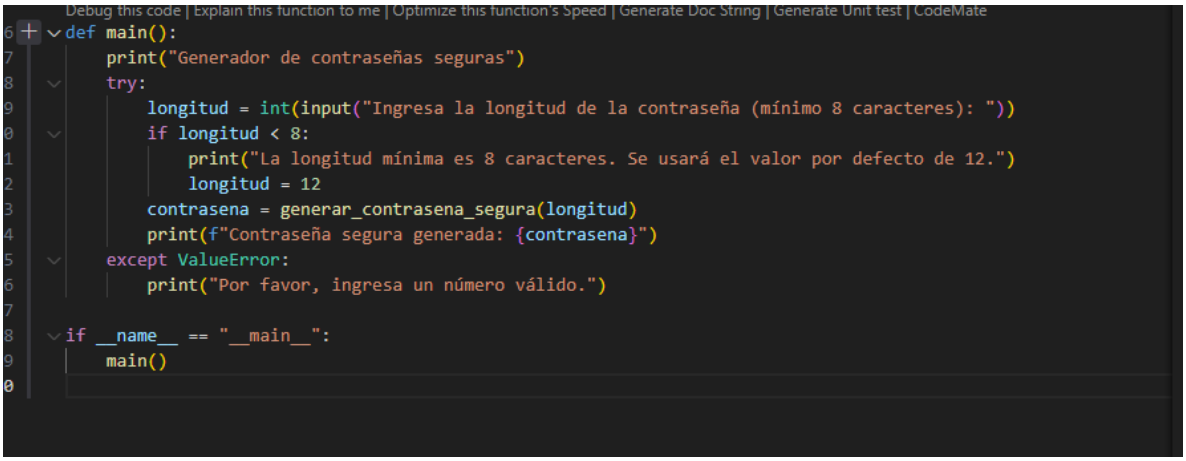
```
PS C:\Users\javie\Desktop> python validador_contrasena.py
Validador de contraseñas seguras
Ingresa tu contraseña: Noecruz16798
La contraseña debe tener al menos un carácter especial (!, @,
#, etc.).
```

```
PS C:\Users\javie\Desktop> python validador_contrasena.py
Validador de contraseñas seguras
Ingresa tu contraseña: Noecruz16798!
La contraseña es segura.
```

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



```python
import random
import string

def generar_contrasena_segura(longitud=12):
    if longitud < 8:
        raise ValueError("La longitud mínima para una contraseña segura es 8 caracteres.")

    caracteres_especiales = "!@#$%^&*()-_=+[]{}|;:'\",.<>/?~`"

    # Garantizar al menos un carácter de cada tipo
    mayuscula = random.choice(string.ascii_uppercase)
    minuscula = random.choice(string.ascii_lowercase)
    numero = random.choice(string.digits)
    especial = random.choice(caracteres_especiales)

    # Rellenar el resto de la contraseña con una mezcla aleatoria
    todos_caracteres = string.ascii_letters + string.digits + caracteres_especiales
    resto = [random.choice(todos_caracteres) for _ in range(longitud - 4)]

    # Combinar y mezclar los caracteres
    contrasena = list(mayuscula + minuscula + numero + especial + ''.join(resto))
    random.shuffle(contrasena)

    return ''.join(contrasena)

def main():
    print("Generador de contraseñas seguras")
    try:
        longitud = int(input("Ingresa la longitud de la contraseña (mínimo 8 caracteres): "))
        if longitud < 8:
```

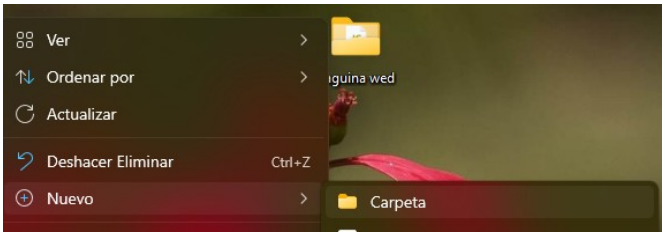


```python
def main():
    print("Generador de contraseñas seguras")
    try:
        longitud = int(input("Ingresa la longitud de la contraseña (mínimo 8 caracteres): "))
        if longitud < 8:
            print("La longitud mínima es 8 caracteres. Se usará el valor por defecto de 12.")
            longitud = 12
        contrasena = generar_contrasena_segura(longitud)
        print(f"Contraseña segura generada: {contrasena}")
    except ValueError:
        print("Por favor, ingresa un número válido.")

if __name__ == "__main__":
    main()
```

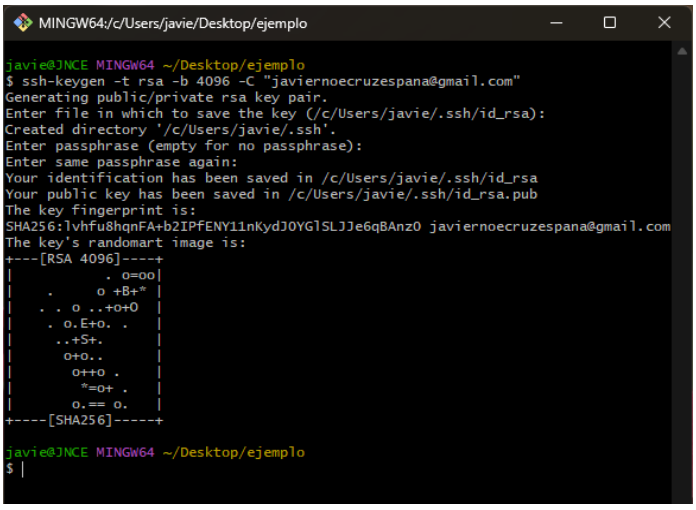
```
Generador de contraseñas seguras
Ingresa la longitud de la contraseña (mínimo 8 caracteres): 8
Contraseña segura generada: CX8Mh!y1
PS C:\Users\javie\Desktop>
```

```
PS C:\Users\javie\Desktop> python validador_contrasena.py
Generador de contraseñas seguras
Ingresa la longitud de la contraseña (mínimo 8 caracteres): 5
La longitud mínima es 8 caracteres. Se usará el valor por defecto de 12.
Contraseña segura generada: pK7X`pA9Ax;T
PS C:\Users\javie\Desktop> python validador_contrasena.py
Generador de contraseñas seguras
Ingresa la longitud de la contraseña (mínimo 8 caracteres): 8
Contraseña segura generada: >2WogO1F
PS C:\Users\javie\Desktop>
```

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



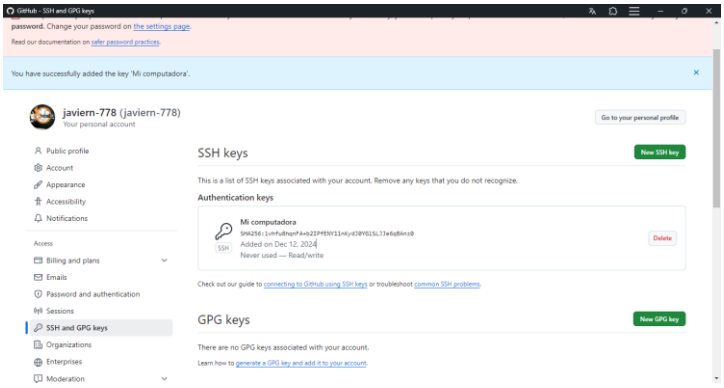
Generar un certificado SSH (par de claves)



Copiar la clave pública

```
javie@JNCE MINGW64 ~/Desktop/ejemplo
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCpLcwLarIOuGw7oJTFCwCCqbG36ofWdC9uufIbANHv/
4ooIn1xTkuDXK5dSPXxHvsRflyTVcs7JdvW6hzLJBwY1rarjSw4+ex8luYToWE2QOASRDuANbKT3AaZHj
MlWvOWeVp8ONYYD3F+/V7iLgFthe6DAKNN9GVwgE4NhWRaezkktbgvbcmWeP/2PzjP186gvYOcBsjT83Q
U1qdsk6LHp8O19meTh7a6U3Mxp1evduU2yaZcPe5aTAaRSJ7YWzJOi4KGvLDYl4luJATFW2pKOO9iAQyp
JTHJpuQUnJOQtjyeZ1/gkygzh71JR7FFGONRFoN7hTrWCsMhYFgsVpx+LOxMCfqXGdERzoWWUfPcZ75E7
HyubnpKIdMmHcu42U/KasP1YOLZWUzoqJo43bd4tHVpWbdDRv7DQFxkr0MO39SSuOZeJvPiG9NhgBOe/O
KzZZOdZmxJMpB+9+6O19qCK19uVyWkojVaHE+dxmtWGMKOXnNF7UFV3OA8NOH/WUV2+qVsRp6KRLGSvrv
YTAfCIh9WRIlXDckdoltBbyowwKvwKOClsQ3zu4CvOLw2Bwkum4LX2KMS5vfWhJEUYDcOgk9JpNoRKTGW
KVVsSauqM1u29xenhN7GSuFTJIzezC8fZOaceeMRVRqY+Lr9fpQ1z5kFIhyQpqH7eqAnFQVEkQ== javi
ernoecruzespana@gmail.com

javie@JNCE MINGW64 ~/Desktop/ejemplo
$
```

Añadir la clave pública a GitHub



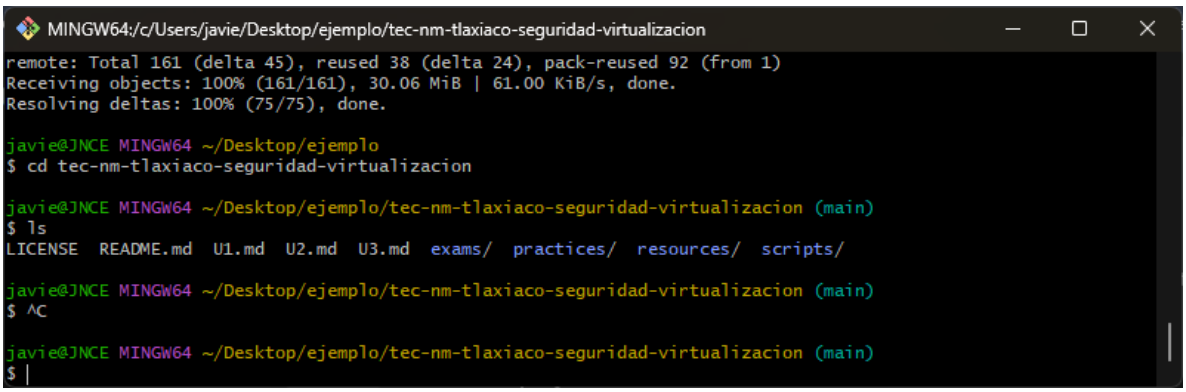
Probar la conexión SSH

```
MINGW64:/c/Users/javie/Desktop/ejemplo

javie@JNCE MINGW64 ~/Desktop/ejemplo
$ ssh -T git@github.com
Hi javiern-778! You've successfully authenticated, but GitHub does not provide shell access.

javie@JNCE MINGW64 ~/Desktop/ejemplo
$
```

Clonar el repositorio usando SSH

```
MINGW64:/c/Users/javie/Desktop/ejemplo
javie@JNCE MINGW64 ~/Desktop/ejemplo
$ ssh -T git@github.com
Hi javiern-778! You've successfully authenticated, but GitHub does not provide shell access.

javie@JNCE MINGW64 ~/Desktop/ejemplo
$ git clone git@github.com:javiern-778/tec-nm-tlaxiaco-seguridad-virtualizacion.git
Cloning into 'tec-nm-tlaxiaco-seguridad-virtualizacion'...
remote: Enumerating objects: 161, done.
remote: Counting objects: 100% (69/69), done.
remote: Compressing objects: 100% (45/45), done.
remote: Total 161 (delta 45), reused 38 (delta 24), pack-reused 92 (from 1)
Receiving objects: 100% (161/161), 30.06 MiB | 61.00 KiB/s, done.
Resolving deltas: 100% (75/75), done.

javie@JNCE MINGW64 ~/Desktop/ejemplo
$
```

Verifica el contenido del repositorio



Conceptos de Seguridad en Redes y Criptografía

1. Contraseña

Una **contraseña** es una combinación secreta de caracteres (letras, números y símbolos) utilizada para verificar la identidad de un usuario y proteger el acceso a datos y recursos. Las contraseñas robustas deben ser difíciles de adivinar y contener una mezcla de caracteres para evitar ataques como el **fuerza bruta** y el **diccionario**.

2. Certificado Digital

Un **certificado digital** es un archivo electrónico que autentica la identidad de una entidad (usuario, servidor o dispositivo) y facilita la comunicación segura en redes. Los certificados utilizan la **infraestructura de clave pública (PKI)** y contienen:

- **Clave pública**
- **Información del propietario**
- **Autoridad certificadora (CA)**
- **Fecha de expiración**

Se emplean principalmente en HTTPS y otras tecnologías de encriptación para garantizar la confidencialidad y autenticidad.

3. Firma Digital

La **firma digital** es un mecanismo criptográfico que garantiza:

- La **autenticidad** del origen del mensaje.
- La **integridad**, asegurando que el mensaje no se haya modificado.

La firma digital utiliza una clave privada del remitente y una clave pública del receptor para verificar estos atributos.

4. Cifrado Simétrico

El **cifrado simétrico** utiliza una única clave tanto para **encriptar** como para **desencriptar** la información. Es más rápido y eficiente pero presenta desafíos, como la gestión segura de claves. Ejemplos: **AES** (Advanced Encryption Standard).

Ventajas:

- Rápido y eficiente.
- Uso menos costoso en recursos.

Desventajas:

- La distribución segura de claves es crítica y difícil.

5. Cifrado Asimétrico

El **cifrado asimétrico** utiliza dos claves diferentes: una clave **privada** y una clave **pública**. La clave pública encripta y la clave privada desencripta.
Ejemplo: **RSA** (Rivest-Shamir-Adleman).

Ventajas:

- Mejor gestión de claves.
- Mayor seguridad en la comunicación a larga distancia.

Desventajas:

- Más lento y consume más recursos.

6. Hash

El **hash** es una función criptográfica que toma una entrada y devuelve una cadena de longitud fija (valor hash). Es **irreversible**, lo que significa que no se puede obtener la entrada original desde el valor hash.
Ejemplos: **SHA-256**.

Uso:
Se emplea para verificar la integridad de archivos y contraseñas.

7. Encriptación

La **encriptación** es el proceso de transformar información legible (**texto plano**) en un formato codificado (**texto cifrado**) para protegerla contra accesos no autorizados. Se basa en algoritmos de cifrado y claves.

Algoritmos de Cifrado

1. AES (Advanced Encryption Standard)

- **Tipo:** Cifrado simétrico.
- **Uso:** Estándar de cifrado ampliamente utilizado para proteger datos.
- **Características:**
    - Soporte para claves de **128, 192 y 256 bits**.
    - Alta velocidad y eficiencia.
    - Usado en **Wi-Fi, VPN y HTTPS**.

2. RSA (Rivest-Shamir-Adleman)

- **Tipo:** Cifrado asimétrico.
- **Uso:** Seguridad en la transferencia de datos y firma digital.
- **Características:**
    - Usa pares de claves (pública y privada).
    - Comúnmente empleado en **HTTPS, firmas digitales y certificados**.
    - Basado en problemas matemáticos complejos de **factorización de números grandes**.

3. SHA-256 (Secure Hash Algorithm 256 bits)

- **Tipo:** Algoritmo de hash.
- **Uso:** Verificación de la integridad de datos y contraseñas.
- **Características:**
    - Produce un hash de **256 bits**.
    - Muy robusto y difícil de romper.
    - Usado en sistemas blockchain y protocolos de seguridad como **TLS y SSL**.

Estándares de Cifrado

1 . SSL (Secure Sockets Layer)

- **Descripción:** Protocolo criptográfico que proporciona **seguridad y autenticidad** en la comunicación entre clientes y servidores en Internet.
- **Uso:** Ha sido reemplazado en gran medida por TLS, pero todavía se usa en algunos sistemas heredados.
- Garantiza:
    - **Cifrado**,
    - **Integridad**,
    - **Autenticación**.

2. TLS (Transport Layer Security)

- **Descripción:** Es la versión más actualizada del protocolo SSL y proporciona **seguridad robusta para conexiones en redes**.
- **Uso:** Protocolo utilizado en **HTTPS, correo electrónico y VoIP**.
- Proporciona:
    - **Cifrado de datos, autenticación y protección de integridad**.

Protocolos de Seguridad

1. HTTPS (HyperText Transfer Protocol Secure)

- **Uso:** Protocolo para la comunicación segura entre navegadores y servidores web.
- **Características:**
    - Combina **HTTP con SSL/TLS**.
    - Asegura el cifrado y la autenticidad en la transferencia de datos entre el cliente y el servidor.
    - Garantiza la **confidencialidad y la integridad**.

2. SFTP (Secure File Transfer Protocol)

- **Uso:** Protocolo seguro para la transferencia de archivos entre servidores y clientes.
- **Características:**
    - Usa el protocolo **SSH** para cifrar la comunicación y transferir datos de forma segura.
    - Proporciona funcionalidades de **autenticación y control de acceso**.

3. SSH (Secure Shell)

- **Uso:** Protocolo utilizado para acceder a sistemas remotos y ejecutar comandos de manera segura.
- **Características:**
    - Proporciona comunicación **cifrada y autenticación robusta**.
    - Usado en la administración remota de servidores y dispositivos en redes.

Conclusión

Has completado el proceso de configurar tu clave SSH, conectar tu computadora con GitHub y clonar tu repositorio localmente. Todo esto te ha permitido acceder y trabajar con tu proyecto de manera eficiente y segura. Gracias a SSH, ahora puedes realizar operaciones de Git sin necesidad de usar tu contraseña cada vez, lo que facilita y agiliza tu flujo de trabajo.

Este proceso te proporciona las bases necesarias para:

- Colaborar en proyectos de desarrollo utilizando Git y GitHub.
- Gestionar tu código, realizar cambios y mantener tu historial de commits.
- Organizar y trabajar con archivos y recursos del repositorio, lo cual es crucial para el éxito del proyecto.

Bibliografía

1. **Libro:**
    - Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice* (8ª ed.). Pearson.
        - Referencia completa sobre conceptos y algoritmos de cifrado, estándares y protocolos de seguridad.
2. **Artículos y documentación técnica:**
    - **National Institute of Standards and Technology (NIST)**:
        - Publicaciones oficiales y documentación técnica sobre algoritmos de cifrado y estándares.
        - Sitio web: https://csrc.nist.gov
3. **Internet y Enciclopedias en línea:**
    - **Wikipedia**
        - Referencias rápidas y explicaciones sobre conceptos criptográficos y algoritmos.
        - Sitio web: https://www.wikipedia.org
4. **Documentación de Protocolos de Seguridad:**
    - **RFC (Request for Comments)**:
        - Documentación técnica oficial sobre estándares y protocolos de Internet, como **SSL, TLS y SSH**.
        - Sitio web: https://www.rfc-editor.org
5. **Sitios educativos y recursos online:**
    - **Cybrary** y **Coursera**:
        - Ofrecen cursos y materiales sobre conceptos de seguridad informática, cifrado y criptografía.