Diagramas de funcionalidad y repositorio (Generador de contraseñas seguras).

Autor:

- Vásquez Edwin

Docente:

- Iván Reyes

1 Dirección GIT

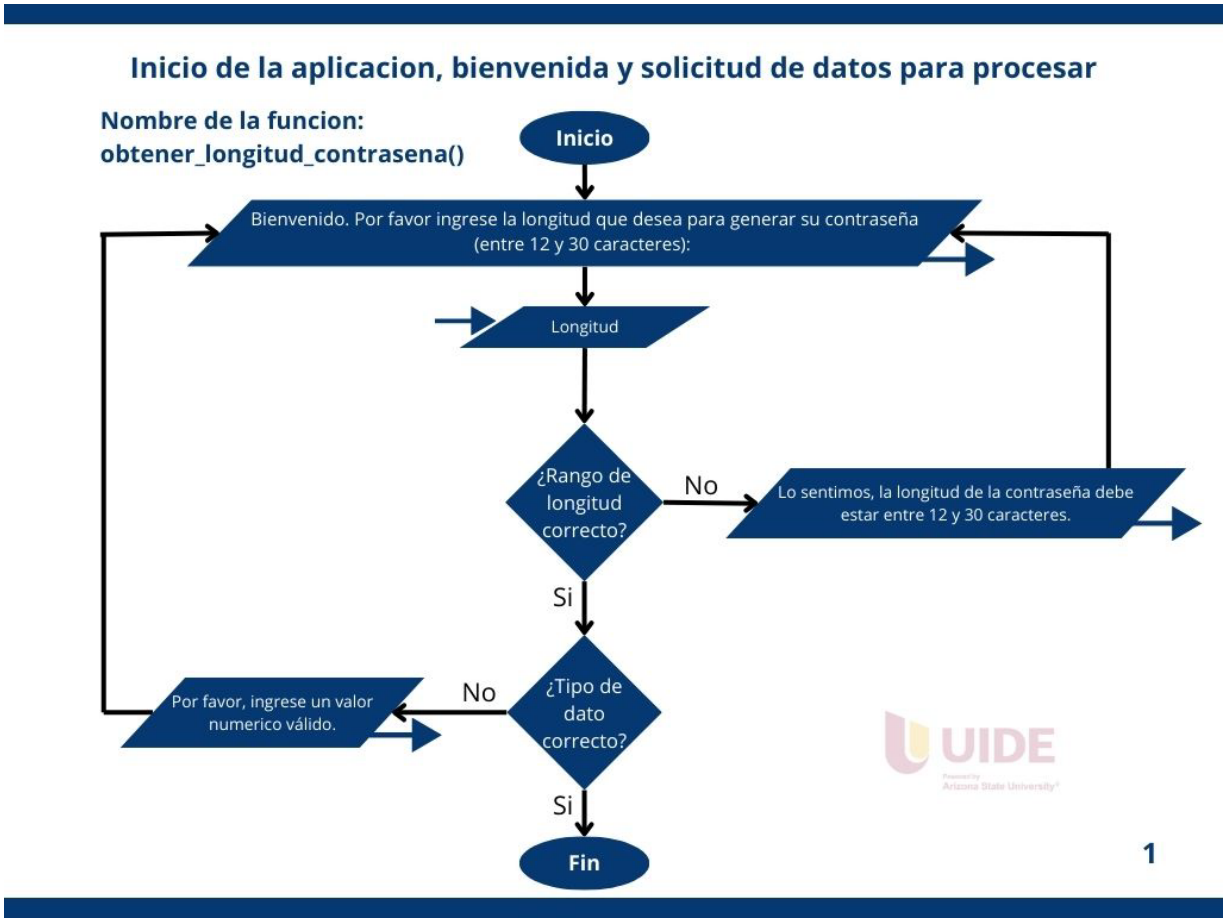
Los diagramas de flujo, documentos y el inicio de la codificación en Python se encuentran publicados en el siguiente enlace del repositorio GIT: https://github.com/Evasquez09/LogicaDeProgramacionAA2.git

2 Diagramas de flujo de las primeras funciones importantes

2.1 Función: (obtener_longitud_contraseña())

Esta función se encarga de dar la bienvenida y solicitar la longitud de contraseña que desea el usuario.

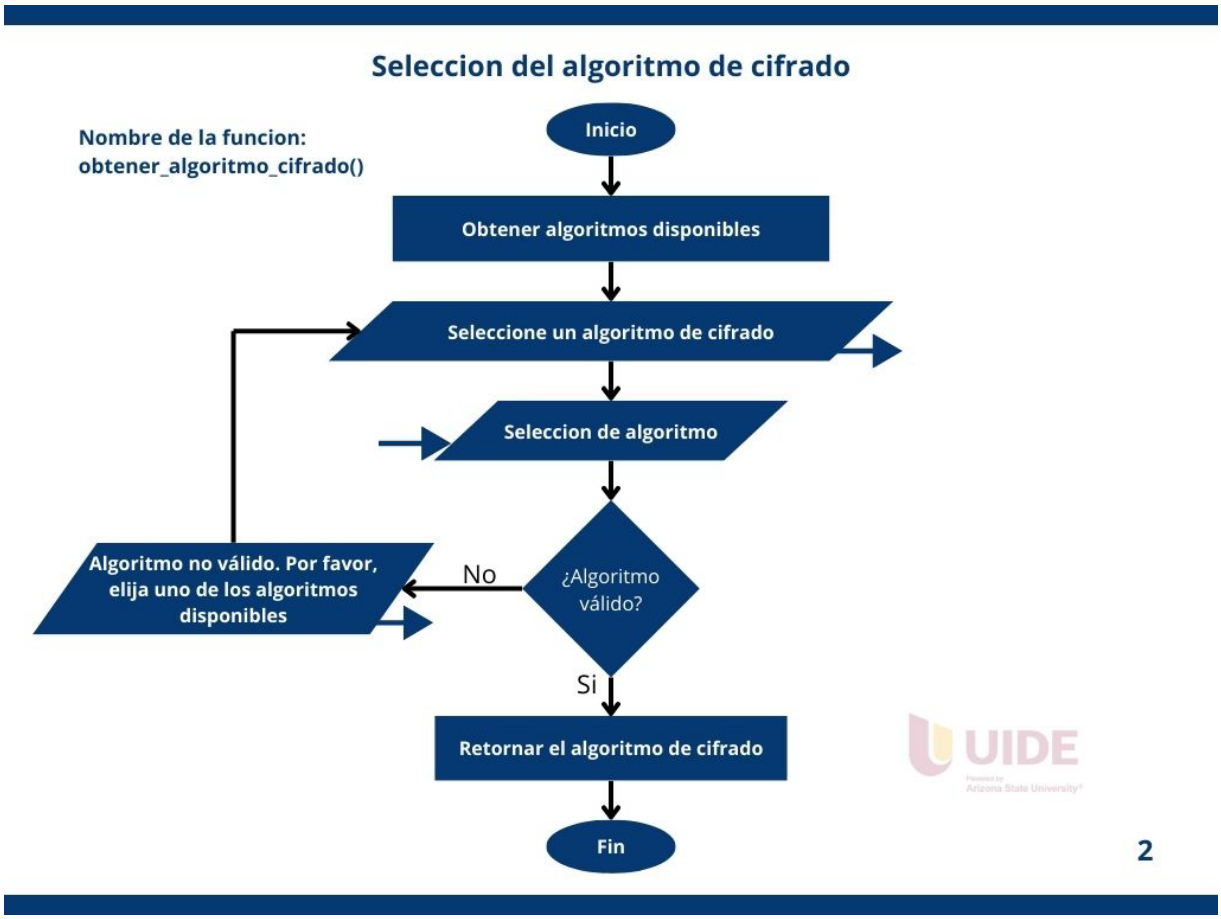
Esta longitud está comprendida entre 12 y 30 caracteres para poder obtener una contraseña que no sea demasiado larga ni corta.



2.2 Función: obtener_algoritmo_cifrado()

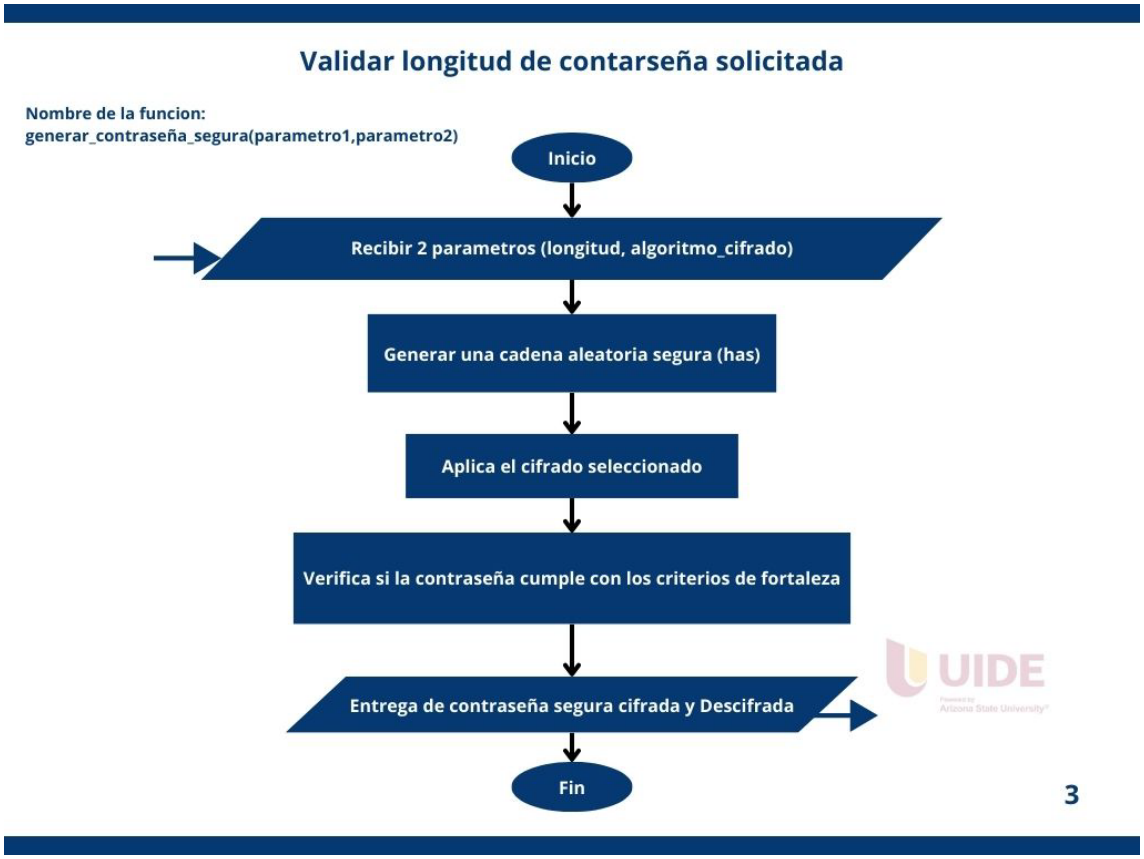
Con esta función el usuario puede elegir el tipo de cifrado que desea aplicar a su contraseña segura. Sus opciones son:

- 'shake_128'
- 'sha3_256'
- 'sha3_384'
- 'sha3_512'
- 'shake_256'
- 'blake2b'
- 'sha256'
- 'md5'
- 'sha384'
- 'sha512'
- 'blake2s'
- 'sha3_224'
- 'sha1'
- 'sha224'



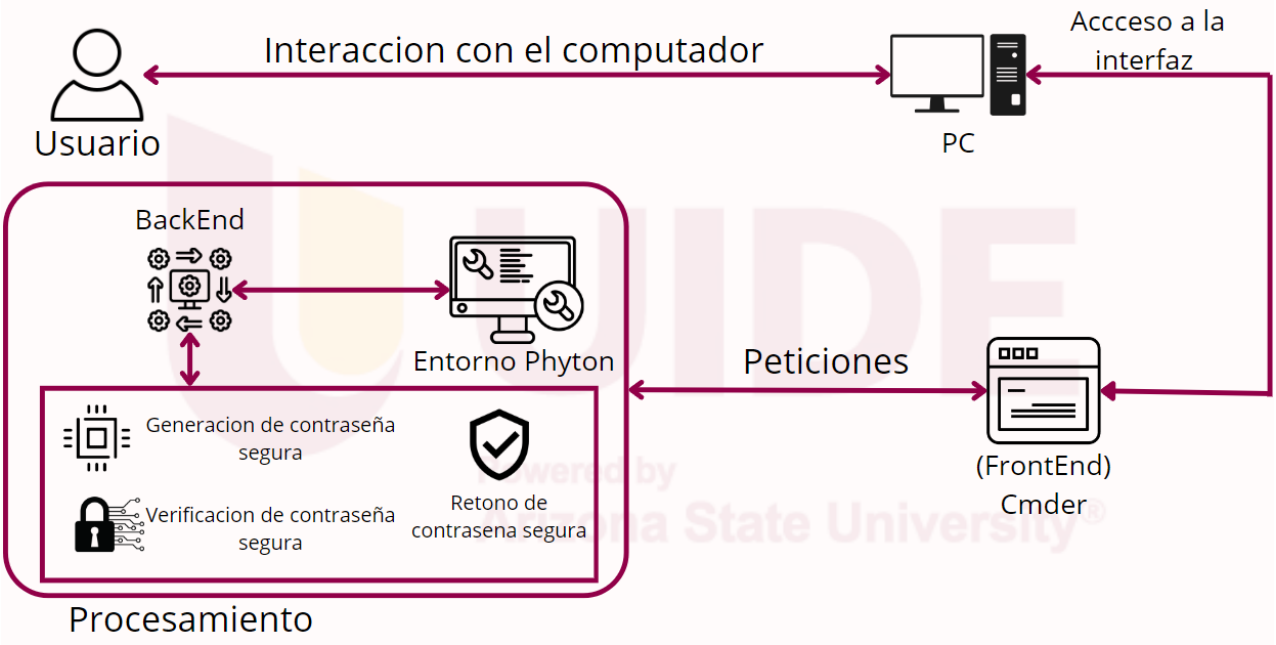
2.3 Función: generar_contraseña_segura(p1,p2)

Esta función se encarga de aplicar el cifrado elegido por el usuario



2.4 Diagrama de arquitectura

De la misma manera se presenta el diagrama de arquitectura, el cual muestra características técnicas del funcionamiento del software que se desarrollara en Python.



3 Código de aplicación

Este código esta incluido en el repositorio GIT:

inicio.py

```python
import secrets #Esta libreria nos proporciona números y textos aleatorios de manera segura.
import hashlib #Esta libreria nos proporciona dibersos hash, es utilizada en este código para cifrar contrasenas

#rango de lonjitudes
LONGITUD_MINIMA = 12
LONGITUD_MAXIMA = 30

#Funcion para generar contrasena segura
def generar_contrasena_segura(longitud, algoritmo_cifrado):
    #Este bucle lo utilizo para asegurarme que el cifrado cumpla con los requisitos de seguridad
    while True:

        # Genera una cadena aleatoria segura, la funcion le pertenese a la libreria secrets
        contrasena = secrets.token_urlsafe(longitud)
        # Aplica el cifrado seleccionado
        cifrado = hashlib.new(algoritmo_cifrado)
        cifrado.update(contrasena.encode('utf-8'))
        contrasena_cifrada = cifrado.hexdigest()
        return contrasena, contrasena_cifrada

#Funcion para obtener diferentes algoritmos de cifrado. Para dar a elejir al usuario el que desea
def obtener_algoritmos_cifrado_disponibles():
    return hashlib.algorithms_guaranteed

#Funcion para validar si la lonjitud de contrasena requerida esta entre 12 y 30 para que no sea demasiado larga ni demasiado corta
def obtener_longitud_contrasena():
    #Ciclo para solicitar la contarseña nuevamente hasta que se cumpla con los paraetros requeridos
    while True:
        try:
            longitud_contrasena = int(input("Bienvenido. Por favor ingrese la longitud que desea para generar su contraseña (entre 12 y 30 caracteres): "))
            if LONGITUD_MINIMA <= longitud_contrasena <= LONGITUD_MAXIMA:
                return longitud_contrasena
            else:
                print("Lo sentimos, la longitud de la contraseña debe estar entre 12 y 30 caracteres.")
        except ValueError:
            print("Por favor, ingrese un valor numerico válido.")

#Funcion para solicitar diferentes algoritmos de cifrado.
def obtener_algoritmo_cifrado():
    algoritmos_disponibles = obtener_algoritmos_cifrado_disponibles()

    #Ciclo para reintentarsolicitar la contarseña nuevamente hasta que se cumpla con los paraetros requeridos
    while True:
        algoritmo_cifrado = input(f"Seleccione un algoritmo de cifrado ({', '.join(algoritmos_disponibles)}): ").lower()
        if algoritmo_cifrado in algoritmos_disponibles:
            return algoritmo_cifrado
        else:
            print("Algoritmo no válido. Por favor, elija uno de los algoritmos disponibles.")

# Solicitar al usuario la longitud y el algoritmo de cifrado
longitud_contrasena = obtener_longitud_contrasena()
algoritmo_cifrado = obtener_algoritmo_cifrado()

# Generar la contrasena segura
contrasena, contrasena_cifrada = generar_contrasena_segura(longitud_contrasena, algoritmo_cifrado)

# Mostrar la contrasena cifrada
print("Esta es su contraseña cifrada:", contrasena_cifrada)
print("Esta es su contraseña segura en claro:", contrasena)
```