

# 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

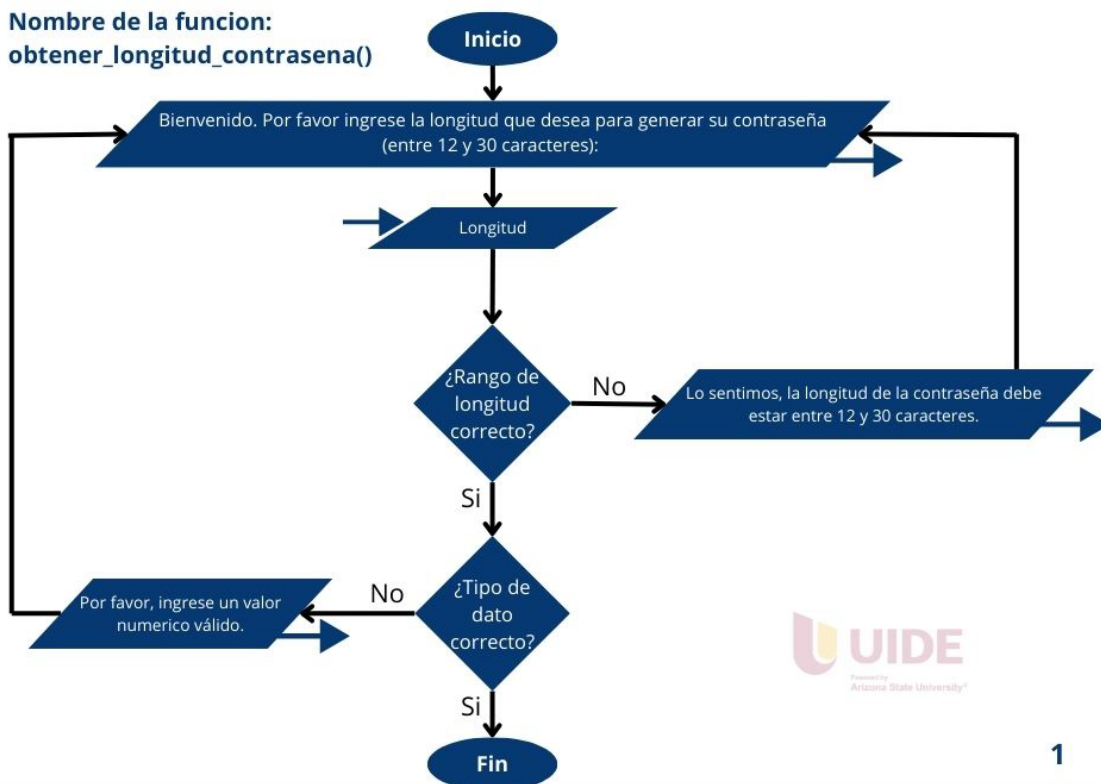
### 2.1 Función: (obtener\_longitud\_contraseña())

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.

#### Inicio de la aplicación, bienvenida y solicitud de datos para procesar

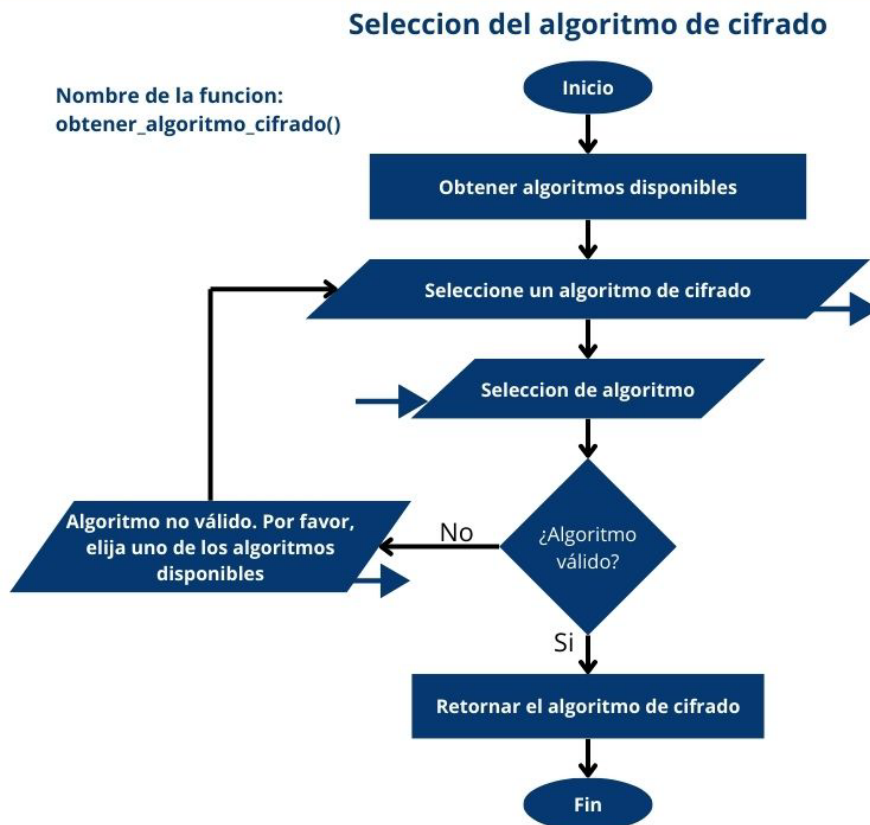
Nombre de la función:  
obtener\_longitud\_contraseña()



## 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'
- 'blake2b'
- 'blake2s'
- 'sha3\_256'
- 'sha256'
- 'sha3\_224'
- 'sha3\_384'
- 'md5'
- 'sha1'
- 'sha3\_512'
- 'sha384'
- 'sha224'
- 'shake\_256'
- 'sha512'



## 2.3 Función: generar\_contraseña\_segura(p1,p2)

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

### Validar longitud de contraseña solicitada

Nombre de la función:  
 generar\_contraseña\_segura(parametro1,parametro2)

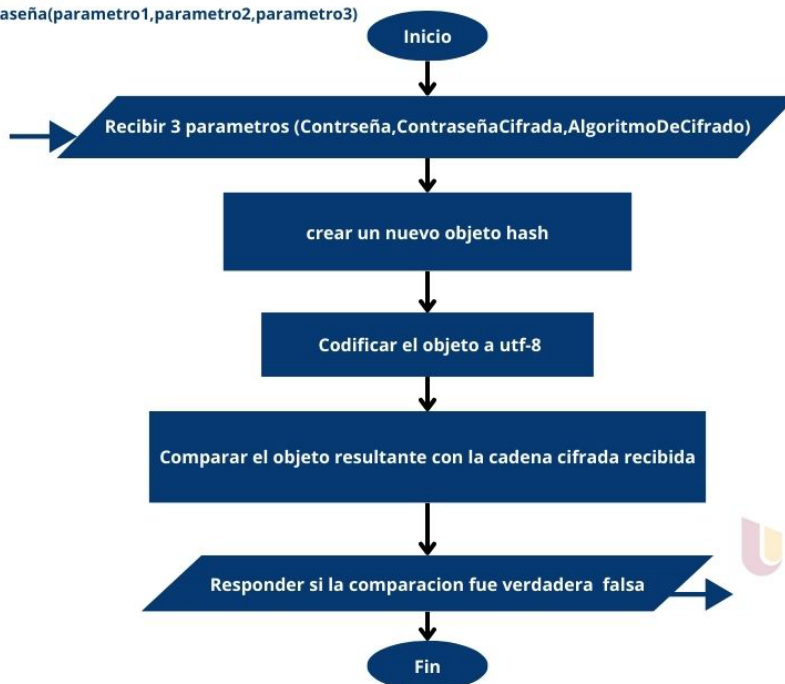


## 2.4 Funcion: verificar\_contraseña(P1,P2,P3)

Esta función se encarga de verificar que la contraseña puede ser descifrada.

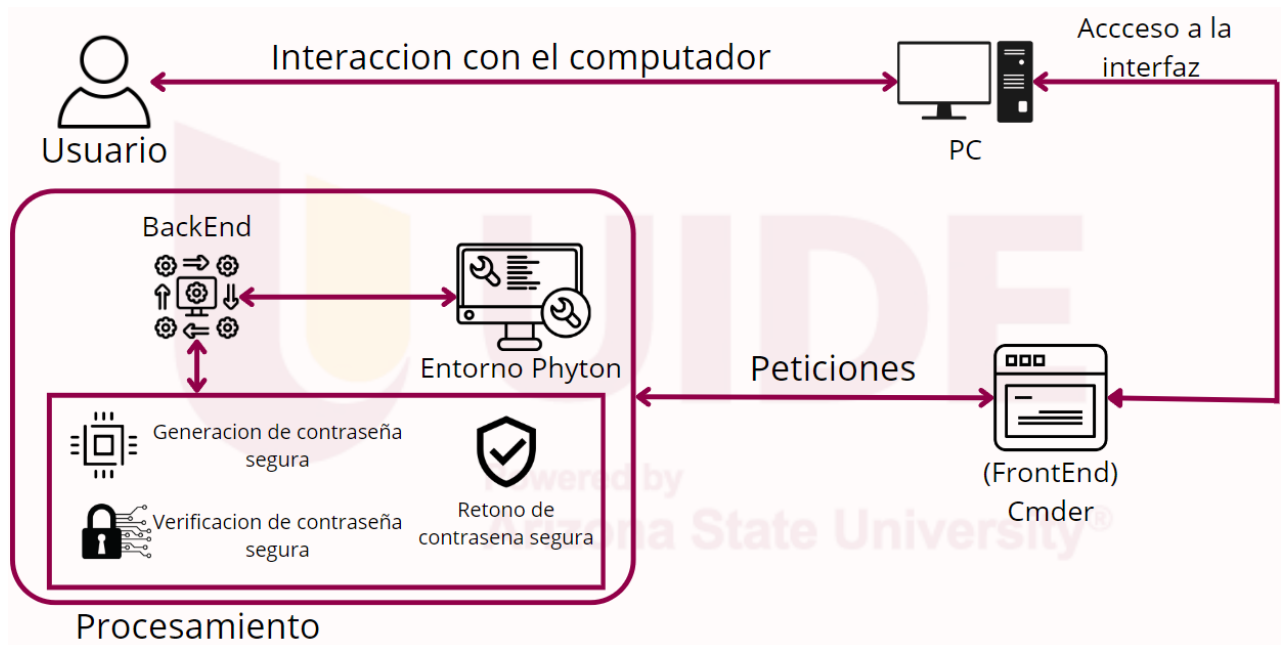
### verificar el descifrado de la contraseña

Nombre de la funcion:  
 verificar\_contraseña(parametro1,parametro2,parametro3)



## 2.5 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 está incluido en el repositorio GIT:

```
inicio.py X
inicio.py > ...
1 import secrets #Esta libreria nos proporciona números y textos aleatorios de manera segura.
2 import hashlib #Esta libreria nos proporciona diversos hash, es utilizada en este código para cifrar contraseñas
3
4 #rango de longitudes
5 LONGITUD_MINIMA = 12
6 LONGITUD_MAXIMA = 30
7
8 #Funcion para generar contraseña segura
9 def generar_contraseña_segura(longitud, algoritmo_cifrado):
10     #Este bucle lo utilizo para asegurarme que el cifrado cumpla con los requisitos de seguridad
11     while True:
12
13         # Genera una cadena aleatoria segura, la funcion le pertenece a la libreria secrets
14         contraseña = secrets.token_urlsafe(longitud)
15         # Aplica el cifrado seleccionado
16         cifrado = hashlib.new(algoritmo_cifrado)
17         cifrado.update(contraseña.encode('utf-8'))
18         contraseña_cifrada = cifrado.hexdigest()
19         return contraseña, contraseña_cifrada
20
21 #Funcion para obtener diferentes algoritmos de cifrado. Para dar a elegir al usuario el que desea
22 def obtener_algoritmos_cifrado_disponibles():
23     return hashlib.algorithms_guaranteed
24
25 #Funcion para validar si la longitud de contraseña requerida esta entre 12 y 30 para que no sea demasiado larga ni demasiado corta
26 def obtener_longitud_contraseña():
27     #Ciclo para solicitar la contraseña nuevamente hasta que se cumpla con los parámetros requeridos
28     while True:
29         try:
30             longitud_contraseña = int(input("Bienvenido. Por favor ingrese la longitud que desea para generar su contraseña (entre 12 y 30 caracteres): "))
31             if LONGITUD_MINIMA <= longitud_contraseña <= LONGITUD_MAXIMA:
32                 return longitud_contraseña
33             else:
34                 print("Lo sentimos, la longitud de la contraseña debe estar entre 12 y 30 caracteres.")
35         except ValueError:
36             print("Por favor, ingrese un valor numerico válido.")
37
38 #Funcion para solicitar diferentes algoritmos de cifrado.
39 def obtener_algoritmo_cifrado():
40     algoritmos_disponibles = obtener_algoritmos_cifrado_disponibles()
41
42     #Ciclo para reintentar solicitar la contraseña nuevamente hasta que se cumpla con los parámetros requeridos
43     while True:
44         algoritmo_cifrado = input(f"Seleccione un algoritmo de cifrado ({', '.join(algoritmos_disponibles)}): ").lower()
45         if algoritmo_cifrado in algoritmos_disponibles:
46             return algoritmo_cifrado
47         else:
48             print("Algoritmo no válido. Por favor, elija uno de los algoritmos disponibles.")
49
50 # Solicitar al usuario la longitud y el algoritmo de cifrado
51 longitud_contraseña = obtener_longitud_contraseña()
52 algoritmo_cifrado = obtener_algoritmo_cifrado()
53
54 # Generar la contraseña segura
55 contraseña, contraseña_cifrada = generar_contraseña_segura(longitud_contraseña, algoritmo_cifrado)
56
57 # Mostrar la contraseña cifrada
58 print("Esta es su contraseña cifrada:", contraseña_cifrada)
59 print("Esta es su contraseña segura en claro:", contraseña)
```