<!--PROYECTO FINAL-->

Generador de contraseñas {

<Por="Steven Guzmán"/>

}

Contenidos

Introducción {

Nombre: Generador Personalizado de Contraseñas y Passphrases

Lenguaje: Python 3

Objetivo: Crear contraseñas y passphrases seguras, con análisis de entropía, verificación opcional contra brechas de datos y exportación flexible.

Por qué es importante:

- Las contraseñas son la primera línea de defensa de la seguridad digital.
- Contraseñas débiles son explotadas constantemente en ataques de fuerza bruta o diccionario.
- Este proyecto integra conceptos de seguridad defensiva: entropía, verificación de exposición y generación aleatoria de alta calidad.

- Seguridad.
- Entriopia.
- Aleatoriedad.
- Passphrase.
- Pwned.
- Exportación.
- Python.

}

Generador de contraseñas {

```python
# ---------------------------
# Funciones de generación
# ---------------------------
def generar_contrasena(longitud: int = 12, minus: bool = True, mayus: bool = True,
                       digitos: bool = True, simbolos: bool = False) -> str:
    """
    Genera una contraseña aleatoria según los parámetros indicados.
    """
    pool = ''
    if minus:
        pool += string.ascii_lowercase
    if mayus:
        pool += string.ascii_uppercase
    if digitos:
        pool += string.digits
    if simbolos:
        pool += '!@#$%&*()-_=+[]{};:,.<>?/'
    if not pool:
        raise ValueError("Debes habilitar al menos un tipo de carácter.")
    return ''.join(secrets.choice(pool) for _ in range(longitud))
```

GENERACIÓN
DE CONTRASEÑA

Concepto de seguridad:

- Secrets garantiza que la contraseña no sea predecible ni reproducible por atacantes.

- la diversidad de tipos de caracteres aumenta la antriopia

}

Generador de passphrases {

```python
def generar_passphrase(palabras: int = 5, wordlist_path: Optional[str] = None) -> str:
    """
    Genera una passphrase de `palabras` palabras. Si se proporciona
    wordlist_path, la usa (una palabra por línea). Si no, usa la lista embebida.
    """
    words = []
    if wordlist_path:
        try:
            with open(wordlist_path, "r", encoding="utf-8") as f:
                words = [w.strip() for w in f if w.strip()]
        except Exception as e:
            raise RuntimeError(f"Error leyendo wordlist '{wordlist_path}': {e}")
    if not words:
        words = DEFAULT_EMBEDDED_WORDLIST
    return ' '.join(secrets.choice(words) for _ in range(palabras))
```

Concepto de seguridad:

- Passphrases largas con palabras aleatorias pueden superar la entropia de contraseñas cortas con símbolos, siendo mas memorables

GENERACIÓN
DE PASSPHRASES

}

Entriopia de Shannon {

```python
def shannon_entropy(s: str) -> float:
    """Calcula la entropía en bits (Shannon) de la cadena."""
    if not s:
        return 0.0
    freq = {}
    for ch in s:
        freq[ch] = freq.get(ch, 0) + 1
    entropy = 0.0
    length = len(s)
    for count in freq.values():
        p = count / length
        entropy -= p * math.log2(p)
    # entropía total en bits = entropy por símbolo * longitud
    return entropy * length
```

Formula

$$H = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

<28 bits

Muy débil

28-36 bits

Débil

36-60 bits

Aceptable

60-80 bits

Fuerte

>=80

Muy Fuerte

}

Consulta Pwned Passwords (K-anonymity){

```python
def pwned_count(password: str) -> Optional[int]:
    """
    Consulta la API de Have I Been Pwned (Pwned Passwords) usando k-anonymity.
    Devuelve el número de veces que aparece la contraseña en brechas (0 si no aparec
    Si requests no está disponible, devuelve None.
    """
    if not REQUESTS_AVAILABLE:
        return None
    sha1 = hashlib.sha1(password.encode('utf-8')).hexdigest().upper()
    prefix, suffix = sha1[:5], sha1[5:]
    url = f"https://api.pwnedpasswords.com/range/{prefix}"
    try:
        resp = requests.get(url, timeout=10)
    except Exception:
        return None
    if resp.status_code != 200:
        return None
    for line in resp.text.splitlines():
        if not line:
            continue
        try:
            suf, count = line.split(':')
        except ValueError:
            continue
        if suf == suffix:
            try:
                return int(count)
            except ValueError:
                return None
    return 0
```

Flujo Visual

```
[Contraseña] -> SHA-1 -> Primeros 5 chars (prefix) -> API
                                                      |
                                                      v
                                          [Hashes con mismo prefix]
                                                      |
                                          Comparar sufijo local con hashes
                                                      |
                                                      v
                                          Número de veces aparecida
```

Concepto adicional:

- **Mas tipos de caracteres + mayor longitud → mas combinaciones posibles → mas seguridad**

}

Salidas: texto, JSON, CSV

```python
def salida_texto(password: str, entropy: float, label: str, pwned: Optional[int]) -> None:
    print(password)
    print(f"Entropía: {entropy:.2f} bits — {label}")
    if pwned is None:
        print("Pwned: (no disponible / requests no instalado o fallo en la consulta)")
    else:
        if pwned > 0:
            print(f"Pwned: APARECE en {pwned} brechas públicas (cambia esta contraseña).")
        else:
            print("Pwned: No aparece en la base de datos conocida.")
```

TEXTO

Para uso rápido en CLI.

JSON

Para integración con otros sistemas.

CSV

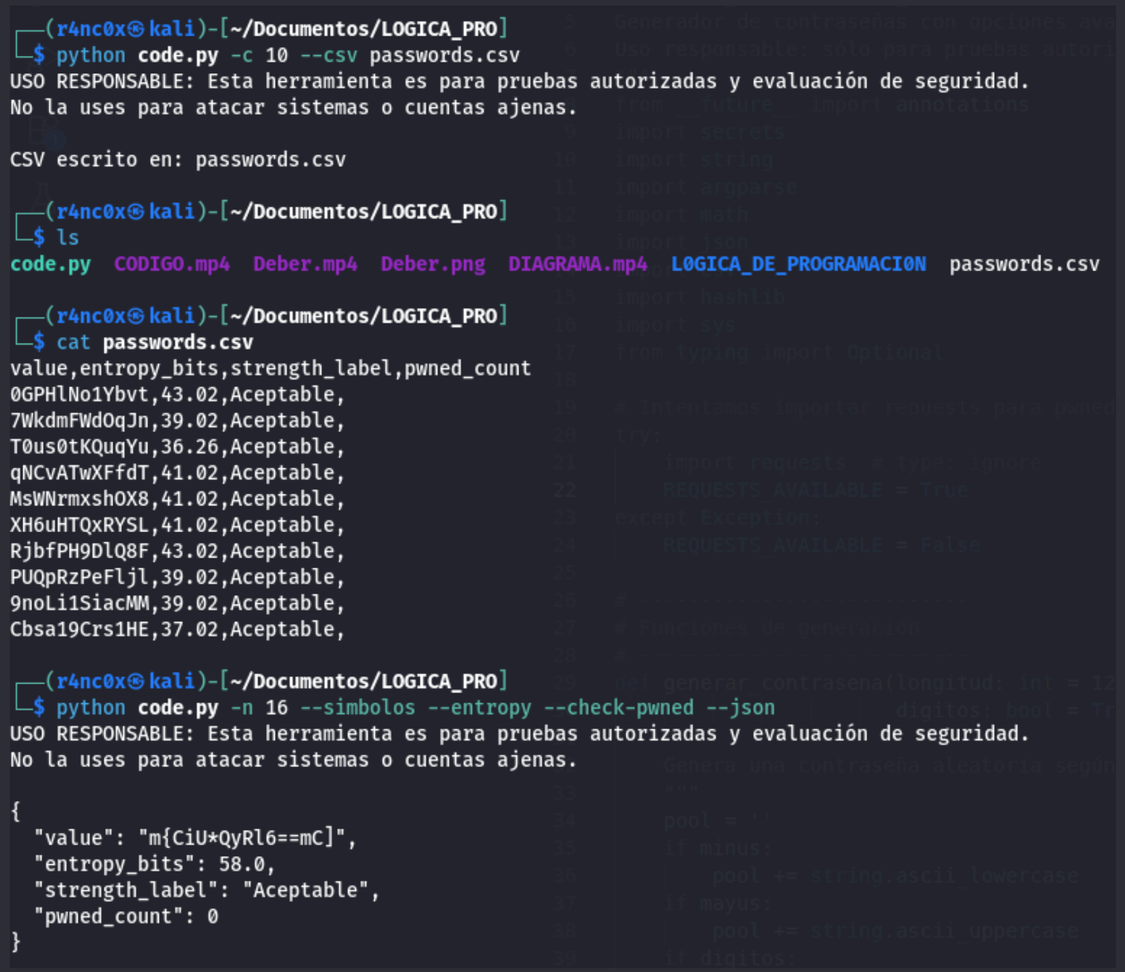
Para generar reportes de varias contraseñas.

}

CLI Principal {

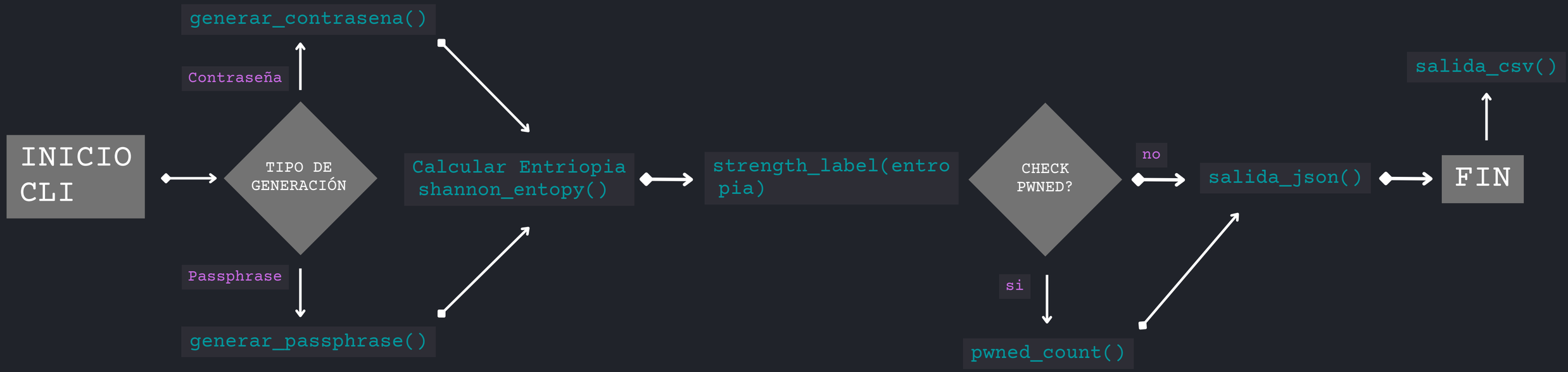
El CLI se implementa mediante el módulo estándar argparse, que facilita la definición de argumentos opcionales y posicionales, proporcionando ayuda contextual y validación automática de parámetros.

De esta forma, el programa puede ejecutarse con distintas combinaciones de opciones, adaptándose a las necesidades del usuario.



}

Estructura {


generar_contrasena()
Contraseña
INICIO CLI
TIPO DE GENERACIÓN
Passphrase
generar_passphrase()
Calcular Entriopia shannon_entopy()
strength_label(entropia)
CHECK PWNED?
no
si
salida_json()
pwned_count()
salida_csv()
FIN

}

<!--Proyecto Final-->

Gracias {

<Por="Steven Guzmán"/>

}