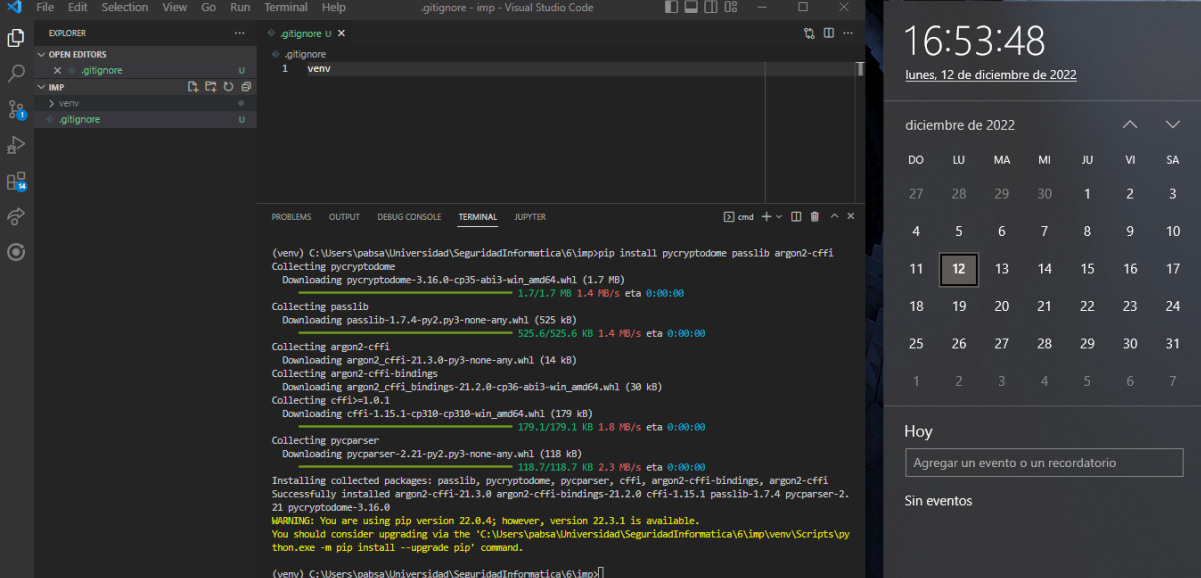


# Actividad 7

Pablo Sanchez Galdamez (21001135)

## Implementación de la arquitectura

### Instalación de las librerías:

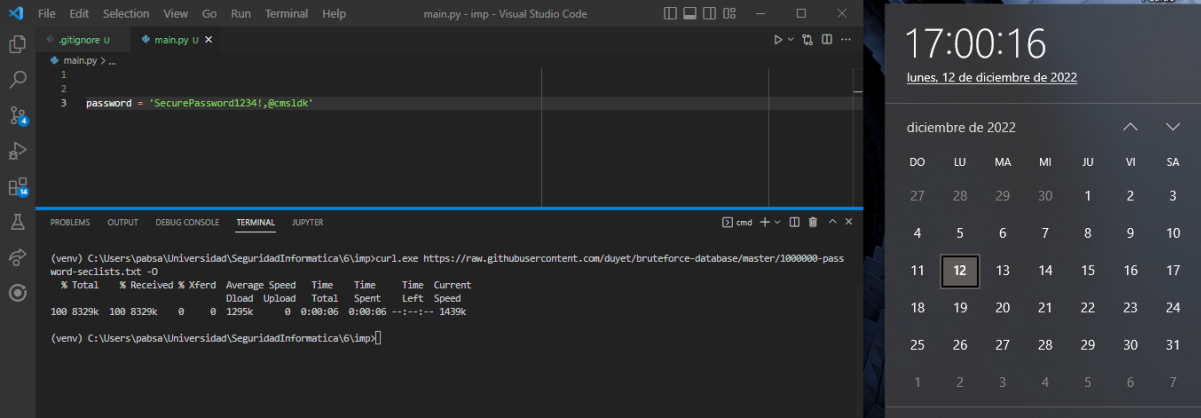


The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the command to install several Python libraries using pip. The output shows the progress of downloading and installing each package, including pycryptodome, passlib, argon2-cffi, and pycparser. A warning message indicates that the user is using pip version 22.0.4, while version 22.3.1 is available. The user is prompted to upgrade pip using the command 'python.exe -m pip install --upgrade pip'.

```
(venv) C:\Users\pabsa\Universidad\SeguridadInformatica\6\imp>pip install pycryptodome passlib argon2-cffi
Collecting pycryptodome
  Downloading pycryptodome-3.16.0-cp35-abi3-win_amd64.whl (1.7 MB)
    1.7/1.7 MB 1.4 MB/s eta 0:00:00
Collecting passlib
  Downloading passlib-1.7.4-py2.py3-none-any.whl (525 kB)
    525.6/525.6 KB 1.4 MB/s eta 0:00:00
Collecting argon2-cffi
  Downloading argon2_cffi-21.3.0-py3-none-any.whl (14 kB)
Collecting argon2-cffi-bindings
  Downloading argon2_cffi_bindings-21.2.0-cp36-abi3-win_amd64.whl (30 kB)
Collecting cffi>=1.0.1
  Downloading cffi-1.15.1-cp310-cp310-win_amd64.whl (179 kB)
    179.1/179.1 KB 1.8 MB/s eta 0:00:00
Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
    118.7/118.7 KB 2.3 MB/s eta 0:00:00
Installing collected packages: passlib, pycryptodome, pycparser, cffi, argon2-cffi-bindings, argon2-cffi
Successfully installed argon2-cffi-21.3.0 argon2-cffi-bindings-21.2.0 cffi-1.15.1 passlib-1.7.4 pycparser-2.21 pycryptodome-3.16.0
WARNING: You are using pip version 22.0.4; however, version 22.3.1 is available.
You should consider upgrading via the 'C:\Users\pabsa\Universidad\SeguridadInformatica\6\imp\venv\Scripts\python.exe -m pip install --upgrade pip' command.

(venv) C:\Users\pabsa\Universidad\SeguridadInformatica\6\imp>
```

### Descarga de la base de datos de las contraseñas:

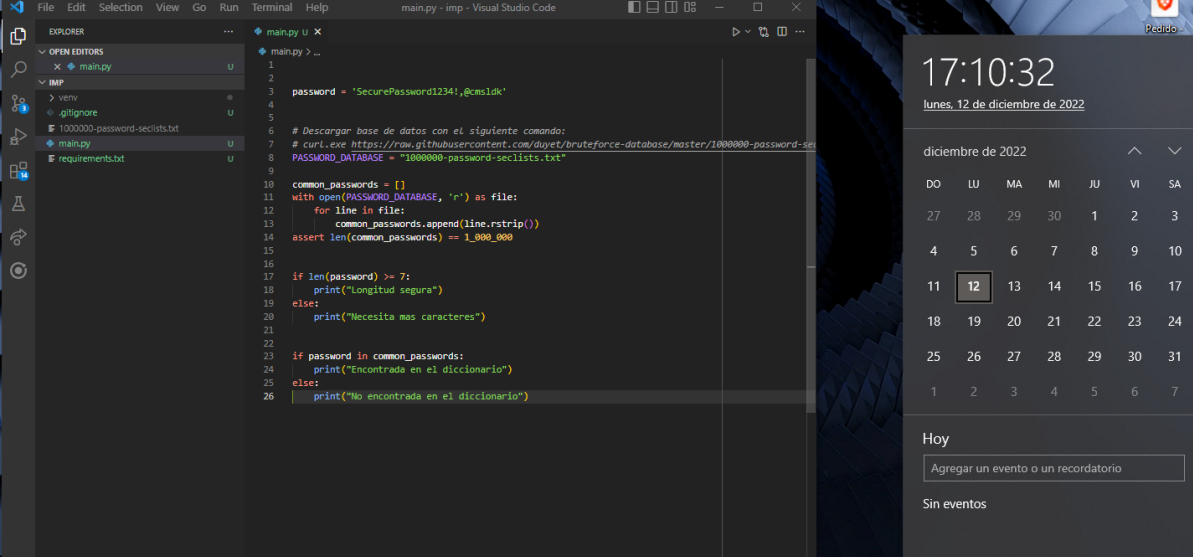


The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the command to download a password database from a GitHub repository using curl. The output shows the progress of the download, including the total size of the file (100 KB) and the current download speed (1439 KB/s).

```
(venv) C:\Users\pabsa\Universidad\SeguridadInformatica\6\imp>curl.exe https://raw.githubusercontent.com/duyet/bruteforce-database/master/1000000-passwords.txt
% Total % Received % Xferd Average Speed Time Time Current
Dload Upload Total Spent Left Speed
100 8329k 100 8329k 0 0 1295k 0 0:00:06 0:00:06 --:--:-- 1439k

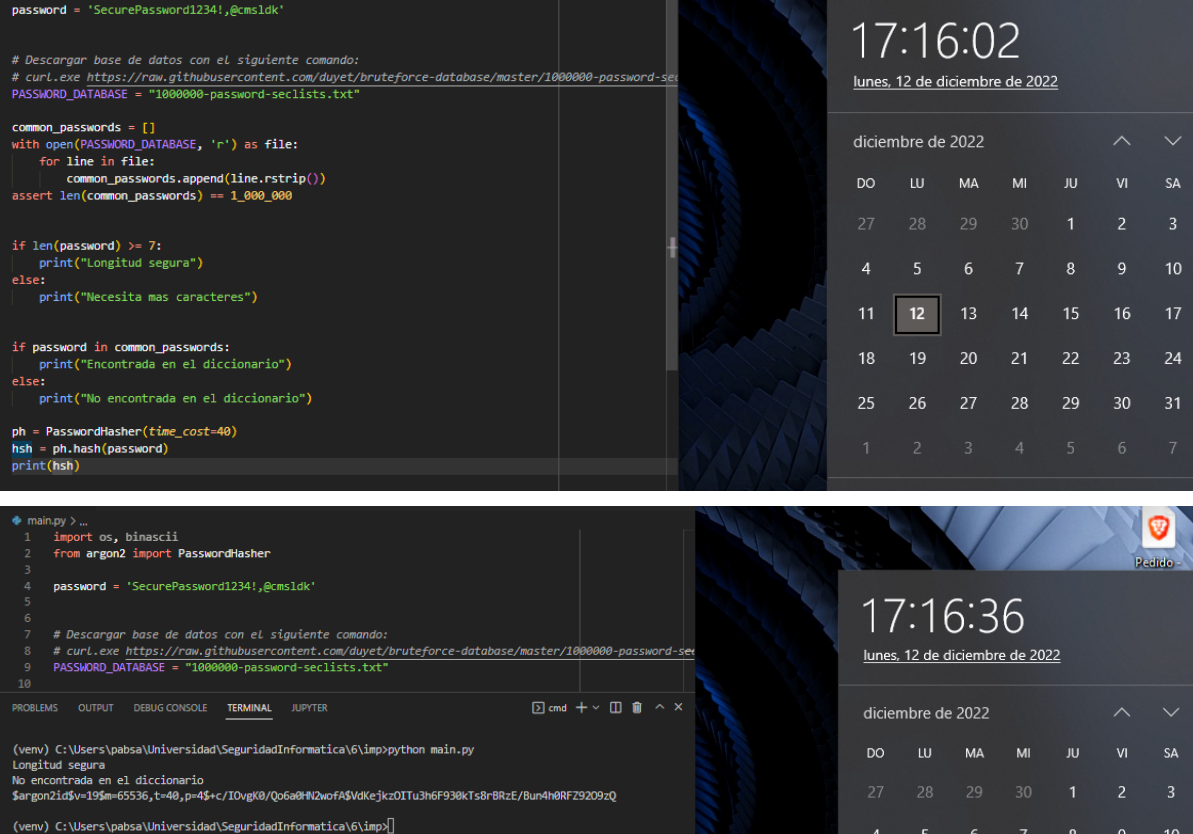
(venv) C:\Users\pabsa\Universidad\SeguridadInformatica\6\imp>
```

Vemos que la contraseña tenga una longitud mayor o igual a 7, y verificamos de que no aparezca en la base de datos:



```
1 password = 'SecurePassword1234!@cmsldk'
2
3
4
5
6 # Descargar base de datos con el siguiente comando:
7 # curl.exe https://raw.githubusercontent.com/duyet/bruteforce-database/master/1000000-password-se
8 PASSWORD_DATABASE = "1000000-password-seclists.txt"
9
10 common_passwords = []
11 with open(PASSWORD_DATABASE, 'r') as file:
12     for line in file:
13         common_passwords.append(line.rstrip())
14 assert len(common_passwords) == 1_000_000
15
16
17 if len(password) >= 7:
18     print("Longitud segura")
19 else:
20     print("Necesita mas caracteres")
21
22
23 if password in common_passwords:
24     print("Encontrada en el diccionario")
25 else:
26     print("No encontrada en el diccionario")
```

Hash de la contraseña:



```
password = 'SecurePassword1234!@cmsldk'

# Descargar base de datos con el siguiente comando:
# curl.exe https://raw.githubusercontent.com/duyet/bruteforce-database/master/1000000-password-se
PASSWORD_DATABASE = "1000000-password-seclists.txt"

common_passwords = []
with open(PASSWORD_DATABASE, 'r') as file:
    for line in file:
        common_passwords.append(line.rstrip())
assert len(common_passwords) == 1_000_000

if len(password) >= 7:
    print("Longitud segura")
else:
    print("Necesita mas caracteres")

if password in common_passwords:
    print("Encontrada en el diccionario")
else:
    print("No encontrada en el diccionario")

ph = PasswordHasher(time_cost=40)
hsh = ph.hash(password)
print(hsh)
```

```
1 import os, binascii
2 from argon2 import PasswordHasher
3
4 password = 'SecurePassword1234!@cmsldk'
5
6
7 # Descargar base de datos con el siguiente comando:
8 # curl.exe https://raw.githubusercontent.com/duyet/bruteforce-database/master/1000000-password-se
9 PASSWORD_DATABASE = "1000000-password-seclists.txt"
10
```

(venv) C:\Users\pabsa\Universidad\SeguridadInformatica\6\imp>python main.py

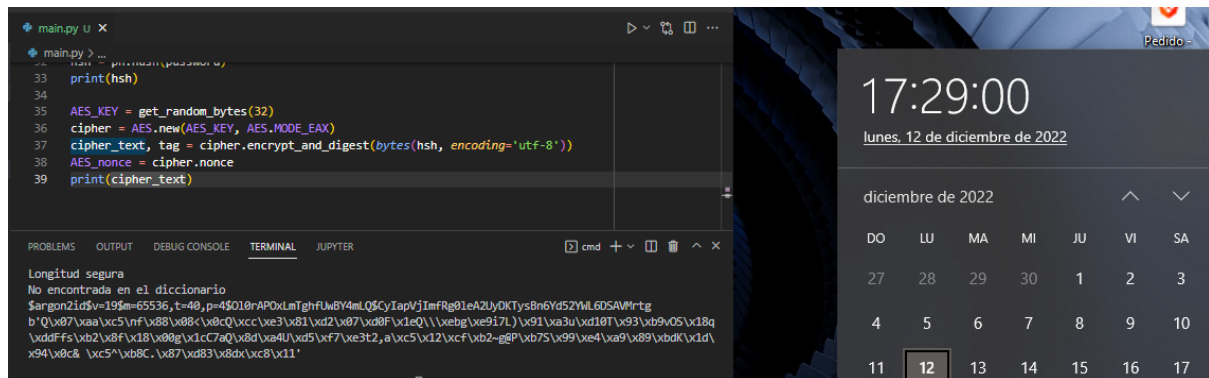
Longitud segura

No encontrada en el diccionario

\$argon2id\$v=19\$m=65536,t=40,p=4\$c/I0vgk8/Qo6a8N2wofA\$VdKejkz0ITU3h6F930kTs8rBRzE/Bun4h0RFZ9209zQ

(venv) C:\Users\pabsa\Universidad\SeguridadInformatica\6\imp>

Pepper:



```
main.py > ...
33 print(hsh)
34
35 AES_KEY = get_random_bytes(32)
36 cipher = AES.new(AES_KEY, AES.MODE_EAX)
37 cipher_text, tag = cipher.encrypt_and_digest(bytes(hsh, encoding='utf-8'))
38 AES_nonce = cipher.nonce
39 print(cipher_text)
```

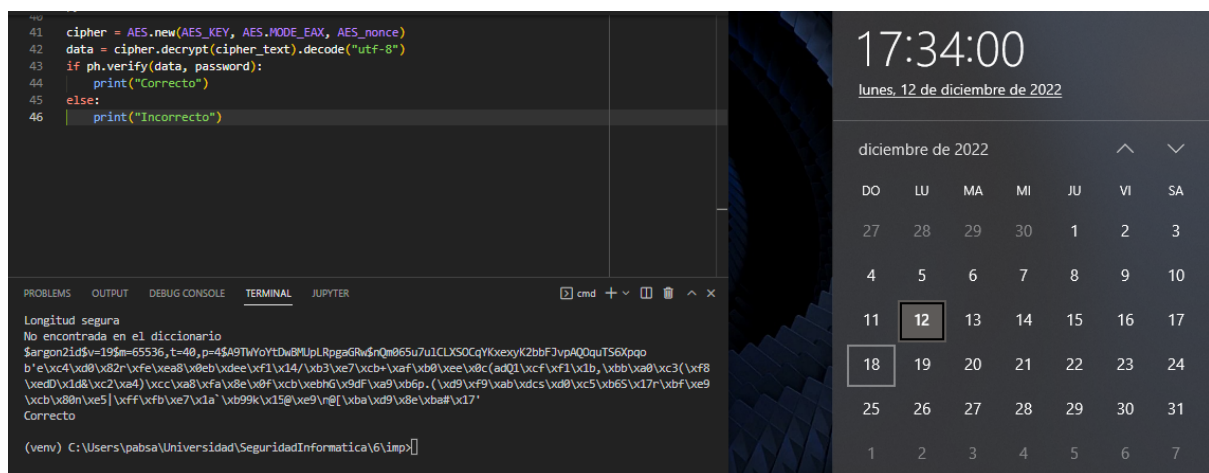
Longitud segura  
No encontrada en el diccionario  
\$argon2id\$v=19\$m=65536,t=40,p=4\$010rAPOxLnTghFLwB4mLQ5CyIapVjImFRg01eAZUyOKTysBn6Yd5ZYWL6DSAMrTg  
b'Qix87\xaa\x55\nf\x88\x08<\x0c\xcc\x03\x81\x02\x07\x0dF\x1eQ\\ \xebg\x09i7L) \x91\x03u\x010T\x93\x09v05\x18q  
\xddF\x02\x08f\x18\x00g\x1c7aQ\x0d\x04u\x05\x07\x0e3t2,a\x05\x12\x0cF\x02-gP\x075\x09\x04\x09\x09\x0dK\x1d  
\x94\x0c& \x05\x08C, \x07\x0d83\x0dx\x0c8\x11'

17:29:00  
lunes, 12 de diciembre de 2022

diciembre de 2022

DO	LU	MA	MI	JU	VI	SA
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

Autenticación:



```
41 cipher = AES.new(AES_KEY, AES.MODE_EAX, AES_nonce)
42 data = cipher.decrypt(cipher_text).decode("utf-8")
43 if ph.verify(data, password):
44     print("Correcto")
45 else:
46     print("Incorrecto")
```

Longitud segura  
No encontrada en el diccionario  
\$argon2id\$v=19\$m=65536,t=40,p=4\$A9TWYoYtDw8MUpLRpGaGRw4nQm065U7u1CLXS0CqYK0xyK2bbfJvpAQ0quTS6Xpgo  
b'e\x04\x0b\x02r\xfe\x08\x0eb\x0de\x0f1\x04/\x03\x07\x0cb+\xaf\x0b\x0ee\x0c(adQ1\x0f\x0f1\x0b, \xb0\x0a0\x03(\xf8  
\x0d\x01d8\x02\x04)\xcc\x08\xfa\x0e\x0f\x0cb\x0bhG\x0dF\x09\x0bP, (\x09\x09\x0ab\x0dcs\x0d\x05\x0b65\x017r\x0bf\x0e9  
\x0b\x08n\x0e5] \xff\x0b\x0e7\x01a' \xb09k\x015@x09\n@(\xba\x09\x0e\x0ba#\x17\*  
Correcto

17:34:00  
lunes, 12 de diciembre de 2022

diciembre de 2022

DO	LU	MA	MI	JU	VI	SA
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

## Justificación de porqué se considera segura esta arquitectura

Hay varias cosas que le hacen la vida prácticamente imposible a un atacante.

Primero el password se convierte en un hash. Esto nos permite a nosotros saber si la contraseña que nos están dando es la correcta, pero le impide a un atacante que obtenga el hash saber cual es la contraseña original.

Segundo, no solo guardamos el hash como texto plano en la base de datos, sino que lo encriptamos. De modo que si se filtra la base de datos, el atacante tendrá que obtener la llave AES primero (Que de por sí es ridículamente costoso), y luego tendrá que encontrar qué contraseña es la que da el hash.

Tercero, revisamos que las contraseñas no esten entre el millón más usado. Así que obligamos al atacante a hacer un costoso trabajo de usar la fuerza bruta, en lugar de un diccionario.

Y cuarto, todas las contraseñas reciben un salt antes de el hash. Esto impide que el atacante tenga una base de datos de contraseñas y sus hashes contra la que comparar a la

nuestra. O si llega a intentar un ataque de fuerza bruta, le impedirá formar una caché con la que pueda probar con otras contraseñas luego.

## ¿Aplicar esta arquitectura sustituye el utilizar doble factor de autenticación (2FA)?

En mi opinión no. Por más complicado e imposible que le hagamos al atacante un ataque de fuerza bruta, esa no es la única forma de obtener las contraseñas de los usuarios.

La mayoría de los hackeos no son por un ataque de fuerza bruta, o por un error muy profundo en la implementación. Hay lugares más fáciles de atacar, y el factor humano siempre será el más sencillo.

Los usuarios siempre pueden caer en phishing, o en ingeniería social, así que no hay nada como preguntarle directamente al usuario: “Está usted iniciando sesión aquí?”