Memoria Práctica Criptografía
(UC3Bank)

Universidad Carlos III de Madrid

Información general

Grupo de clase	**81**
Grupo de prácticas	**01**
Autores del documento	**Carlos Saiz Blanco - 100451312** **Diego Del Pozo Fernández - 100451056**

Github: https://github.com/Carlossb02/G81_01_2022_100451312_100451056

Índice

Cuestiones a responder

- **¿Cuál es el propósito de su aplicación? :**

  Esta aplicación simula el funcionamiento de un pequeño banco online en el que tener tus ahorros y hacer transferencias a otros usuarios. Para ello utilizamos la interfaz de la consola de python, en donde recibiremos y enviaremos los mensajes simulando un cajero. ***(1)***

  Comenzando por el menú, le damos al usuario diversas opciones tras loguearse (seleccionar cuenta, crear cuenta) ***(2)*** y dentro de ellas operaciones ***(3)*** (ingresar, retirar, transferir)... ***(4)***

  Lo más destacable de esta aplicación es la utilización de distintos de funciones hash y cifrado simétrico tanto para proteger la privacidad de argumentos como el pin de la cuenta ***(5)***, el saldo de cuenta o la contraseña.
  Hemos tomado la decisión de no cifrar los números de cuenta ni los nombres de usuario, ya que en la vida real, son datos que se comparten y con los que no se puede realizar ningún tipo de operación salvo que tengas la contraseña o pin.

  La aplicación pedirá el pin de la cuenta correspondiente para iniciar sesión en ella, además, lo volverá a pedir en operaciones de retirada de dinero, transferencias o borrado de cuenta. Esto es debido a que en la realidad, en los bancos no te piden ningún tipo de clave para realizar un ingreso a una cuenta.

  La aplicación permite transferencias de dinero entre usuarios, las cuales son notificadas al usuario que las recibe cuando inicia sesión en la cuenta bancaria que ha recibido el dinero ***(6)***, que se añade a su saldo. Hemos decidido cifrar el saldo de una cuenta con un algoritmo de cifrado simétrico, ya que consideramos que es información que no debe ser pública por diversos motivos.

  La aplicación incluye comprobación de errores, para evitar introducir por consola un dato no válido ***(7)***. Las contraseñas deben tener mayúsculas, minúsculas y números y el pin de una cuenta bancaria debe estar compuesto de 4 dígitos exactamente ***(8)***.

  Por último, el número de cuenta de una cuenta bancaria, se genera de forma aleatoria en base a una expresión regular por motivos de seguridad.***(9)***

- **¿Para qué utiliza el cifrado simétrico? ¿Qué algoritmos ha utilizado y por qué? ¿Cómo gestiona las claves?**

  Hemos utilizado el cifrado simétrico para cifrar los saldos de las cuentas bancarias de los usuarios, debido a que lo consideramos una información sensible

  Para ello, hemos utilizado el algoritmo de Fernet, perteneciente a la librería cryptography nos permite cifrarlo bajo una key que se almacena y que posteriormente sirve para descifrarlo. Hemos utilizado este algoritmo ya que consideramos que es el que mejor se adapta a nuestras necesidades (https://cryptography.io/en/latest/fernet/#using-passwords-with-fernet-)

  El saldo cifrado se almacena en un .json asociado a la cuenta y la key para descifrarlo, se almacena en un .json en la carpeta Database/System.

  Importante: Esto no debe realizarse en condiciones normales, esas keys deben estar protegidas, ya que cualquiera con acceso a ellas, puede descifrar fácilmente el contenido de los mensajes.

  (10) (Ejemplo de las keys de los saldos de dos cuentas distintas en Database/System.symetric-keys.json)

- **¿Para qué utiliza las funciones hash o HMAC? ¿Qué algoritmos ha utilizado y por qué? En caso de HMAC, ¿cómo gestiona la clave/s?**

  Estas funciones hash las hemos utilizado para cifrar tanto la contraseña de la cuenta del usuario, como el pin de cada una de las cuentas.

  Hemos utilizado el algoritmo KDF Scrypt de la librería cryptography para derivar la contraseña del usuario, debido a que es uno de los algoritmos recomendados para este propósito.
  (https://cryptography.io/en/latest/hazmat/primitives/key-derivation-functions/#cryptography.hazmat.primitives.kdf.scrypt.Scrypt)

  También hemos utilizado el algoritmo PBKDF2HMAC de la librería cryptography para proteger el pin de la cuenta bancaria. Hemos decidido utilizar este algoritmo para tener dos algoritmos distintos protegiendo las cuentas, ya que si no se tienen las dos claves (la contraseña y el pin), no se puede operar con las cuentas del usuario.
  (https://cryptography.io/en/latest/hazmat/primitives/key-derivation-functions/#cryptography.hazmat.primitives.kdf.pbkdf2.PBKDF2HMAC)

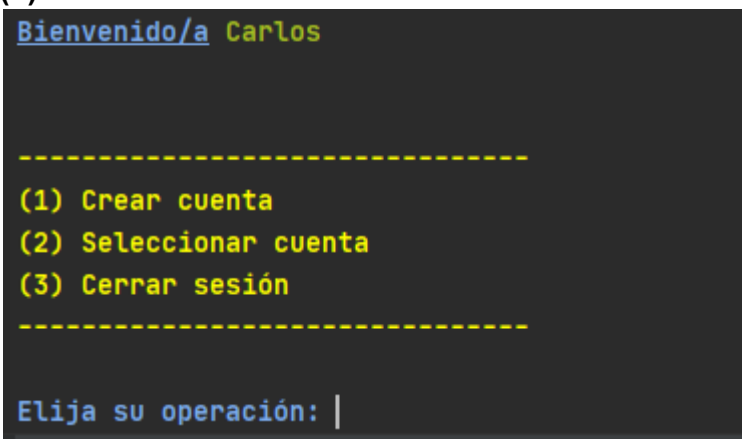
  El token generado de la contraseña junto al salt, se almacenan en un .json junto con el nombre del usuario y las cuentas bancarias que posee el usuario.

  El token generado del pin se almacena junto a su salt en un .json junto al resto de información de la cuenta bancaria.

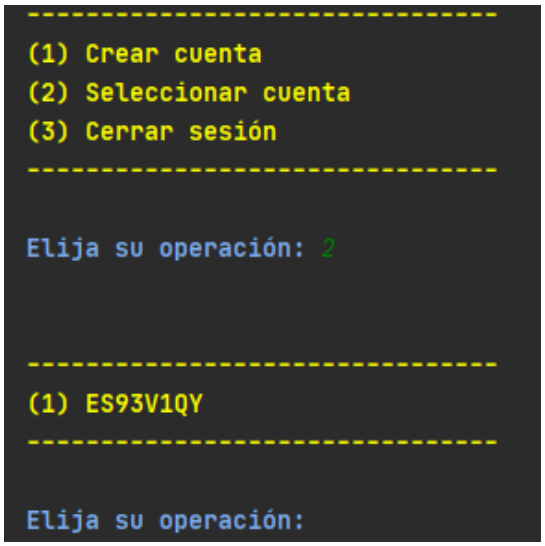
Imágenes

(1)

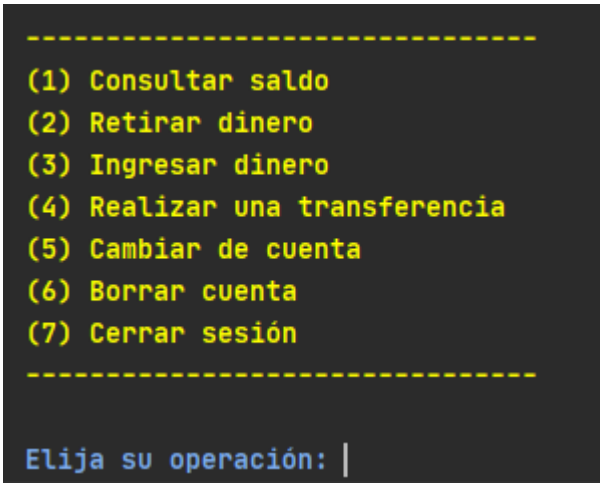
(2)



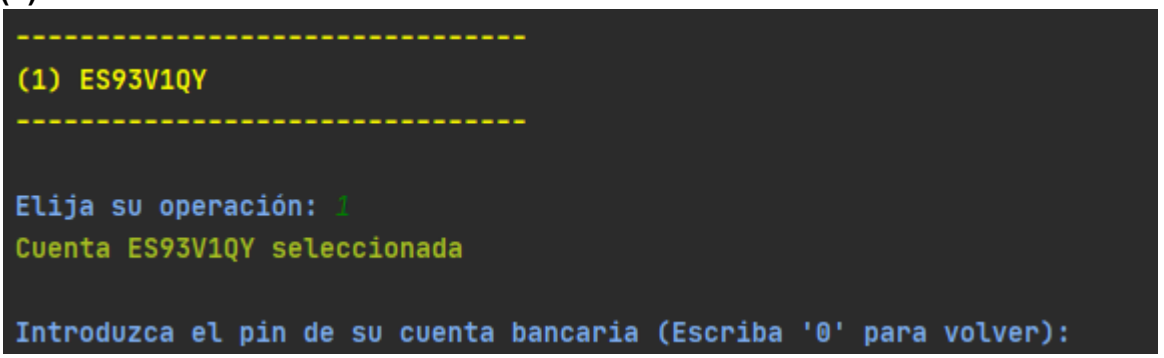
(3)



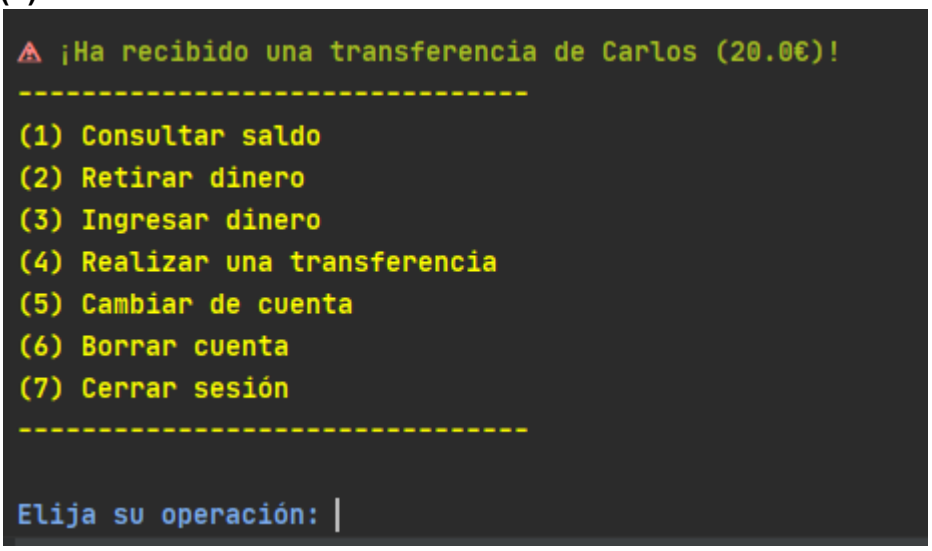
(4)


(1) Consultar saldo
(2) Retirar dinero
(3) Ingresar dinero
(4) Realizar una transferencia
(5) Cambiar de cuenta
(6) Borrar cuenta
(7) Cerrar sesión
Elija su operación:

(5)


(1) ES93V1QY
Elija su operación: 1
Cuenta ES93V1QY seleccionada
Introduzca el pin de su cuenta bancaria (Escriba '0' para volver):

(6)


¡Ha recibido una transferencia de Carlos (20.0€)!
(1) Consultar saldo
(2) Retirar dinero
(3) Ingresar dinero
(4) Realizar una transferencia
(5) Cambiar de cuenta
(6) Borrar cuenta
(7) Cerrar sesión
Elija su operación:

(7)

```
    print("\033[93m--------------------------------")
    print("(1) Consultar saldo")
    print("(2) Retirar dinero")
    print("(3) Ingresar dinero")
    print("(4) Realizar una transferencia")
    print("(5) Cambiar de cuenta")
    print("(6) Borrar cuenta")
    print("(7) Cerrar sesión")
    print("\033[93m--------------------------------")
    sel=input("\n\033[94mElija su operación: ")

    if sel=="1":
        dinero_saldo(usuario, cuenta)
    elif sel=="2":
        dinero_retirar(usuario, cuenta)
    elif sel=="3":
        dinero_ingresar(usuario, cuenta)
    elif sel=="4":
        dinero_transferencia(usuario, cuenta)
    elif sel=="5":
        del(cuenta)
        cuentas(usuario)
    elif sel=="6":
        borrar_cuenta(usuario, cuenta)
    elif sel=="7":
        del(cuenta)
        del(usuario)
        print("\033[92mSe ha cerrado la sesión, hasta pronto...\n")
        home()
    else:
        print("\033[91mError: Esa opción es incorrecta\n")
        operaciones(usuario, cuenta)
```

(8)

```
password_pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{6,}$"
pin_pattern="^(\d{4})$"
```

(9)

```
numero_cuenta=exrex.getone(r'ES[A-Z0-9]{6}')
```

(10)


{"ES93V1QY": "WnzVUVE2KNLxECzvsXxPxVbAVycVW-6gJtDKPNJse9c=", "ES1EWN88": "nREqrXaFEZCFnScz3EeVX7Z8tenorVlg4Nn0mfhYkYI="}