# CRIPTOGRAFÍA Y SEGURIDAD INFORMÁTICA

**Grupo 81**

**JUAN MIGUEL PULGAR ESQUIVEL 100451036**
100451036@alumnos.uc3m.es

**ALBA MARCO UGARTE 100451139**
100451036@alumnos.uc3m.es

**ID DEL GRUPO DE PRÁCTICAS: 19**

**SEPTIEMBRE DE 2022**

# GRADO EN INGENIERÍA INFORMÁTICA UC3M

**• ¿Cuál es el propósito de su aplicación?**
Nuestra aplicación es un casino virtual, donde los usuarios pueden jugar a la ruleta y al blackjack apostando dinero ( Waton Coins ).

Cuando el usuario entra en la aplicación se muestra un menú para registrarse o iniciar sesión si el usuario ya tiene cuenta.

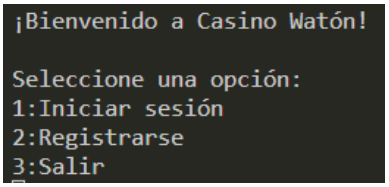



Imagen 1: Menú de inicio

Al registrarse, se pedirán los datos personales del usuario, donde se incluye el DNI, la tarjeta de crédito y una contraseña.

Si el usuario ya se registró anteriormente, podrá iniciar sesión. La aplicación comprobará que la contraseña es válida y se corresponde con la del usuario registrado.

Si el usuario ya ha iniciado sesión, le llevará a un menú donde puede retirar o ingresar dinero (WatonCoins) , jugar a la ruleta o al blackjack.

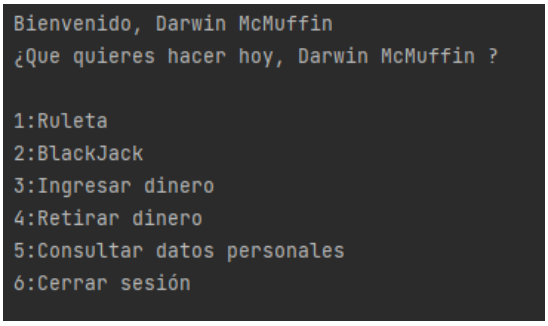



Imagen 2: Menú principal

Antes de empezar a jugar a la ruleta o al blackjack la aplicación preguntará al usuario que cantidad de dinero quiere apostar. Cuando se ha introducido la apuesta, comienza el juego. (La ruleta y el blackjack están simplificados).

En la ruleta tienes que apostar por un número del 0 al 36. Tu apuesta se duplicará si sale un número par y has apostado por uno par (o impar-impar) y tu apuesta se multiplicará por 14 si sale el 0 (y has apostado por el número 0).

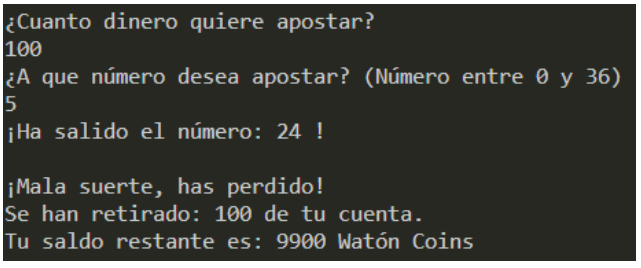



Imagen 3: Ejemplo de ruleta

El objetivo del blackjack es quedarse lo más cerca posible del número 21 pero sin pasarse.
Si algún participante se pasa del 21, habrá perdido. Gana el juego quién tenga un valor de cartas más cercano al 21 (sin pasarse) .
En el blackjack se comienza barajando. Después se entrega una carta al mesero y dos al participante, todas las cartas son visibles para los demás usuarios.
Ahora comienza el turno del participante, éste decidirá si quiere otra carta o no dependiendo del valor de sus cartas. Si supera los 21 puntos perderá y si no, será el turno del mesero.
El mesero pedirá cartas hasta que superé el valor del otro participante o se pase de 21 puntos.
A continuación se muestra una simulación de una partida de blackjack.

```
Barajeando...
Tus cartas son: [[4, 'PICAS'], [2, 'CORAZONES']]

El valor de tus cartas son: 6

Las cartas de la casa son: [['A', 'TREBOLES']]

El valor de las cartas de la casa son: 1


¿Pides otra carta? S/NS
Tus cartas son: [[4, 'PICAS'], [2, 'CORAZONES'], [9, 'CORAZONES']]

El valor de tus cartas son: 15

Las cartas de la casa son: [['A', 'TREBOLES']]

El valor de las cartas de la casa son: 1
```

```
¿Pides otra carta? S/NN
Tus cartas son: [[4, 'PICAS'], [2, 'CORAZONES'], [9, 'CORAZONES']]

El valor de tus cartas son: 15

Las cartas de la casa son: [['A', 'TREBOLES'], [10, 'PICAS']]

El valor de las cartas de la casa son: 11


Tus cartas son: [[4, 'PICAS'], [2, 'CORAZONES'], [9, 'CORAZONES']]

El valor de tus cartas son: 15

Las cartas de la casa son: [['A', 'TREBOLES'], [10, 'PICAS'], ['J', 'PICAS']]

El valor de las cartas de la casa son: 21


!Perdiste Watón! Otra vez será.
Se han retirado: 100 de tu cuenta.
Tu saldo restante es: 9800 Watón Coins
```

Imágenes 4 y 5: Ejemplo de blackjack

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

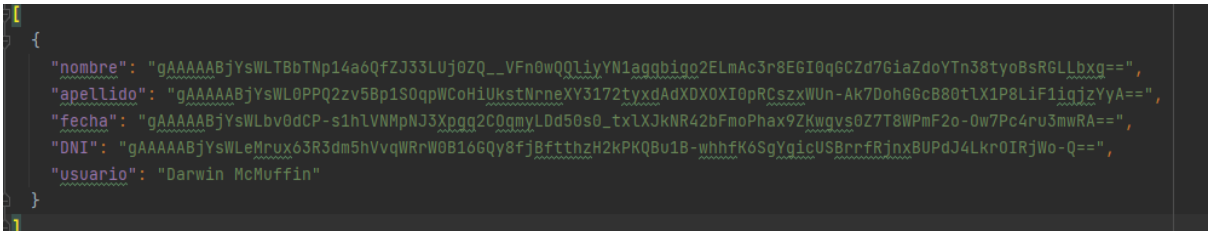
En el caso de nuestra aplicación, se utiliza el cifrado simétrico sobre un archivo JSON que contiene los datos personales de los usuarios, es decir, su nombre completo, fecha de nacimiento, DNI, tarjeta de crédito y nombre de usuario.

Para ello se ha decidido utilizar el algoritmo Fernet, principalmente debido a que creemos que es más sencillo de implementar que AES y parecido computacionalmente, además de ser igual o más seguro. Para implementar este algoritmo, utilizamos 2 claves que encriptan los datos de los usuarios y los datos bancarios de los usuarios. Hemos decidido utilizar claves diferentes ya que consideramos que utilizar varias claves aporta mayor seguridad al sistema, puesto que en caso de que se filtrase la clave de los datos de los usuarios el dinero de cada uno se mantendría oculto y viceversa.

```
[
  {
    "clave": "IuxZxetzFQRJZoCzBaDOoaEnuvhyl9j2GD7FdghHkW4="
  },
  {
    "clave": "ArLAnoyssBóz5rSxiwu5IzDuVDEHNpL2B9vSELdnsQQ="
  }
]
```

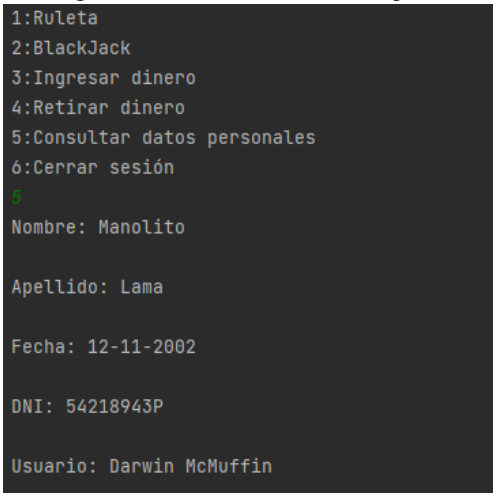
Imagen 6: Claves de usuario y dinero

Los datos del usuario son cifrados cuando se crea una cuenta y pueden ser descifrados siempre que el usuario quiera comprobar o modificar sus datos, ya que no sería posible visualizar los datos si están cifrados. A la hora de gestionar las claves, se ha decidido almacenar externamente el fichero JSON donde se almacenan las claves de encriptación (a la hora de entregar la práctica se entregará todo junto ya que no se podría comprobar el ejercicio de otra manera). Hemos decidido utilizar un pen-drive para almacenar las claves de forma que solo puedan ser accedidas de forma local. De esta manera prevenimos posibles accesos a la clave de encriptación no deseados que podría provocar que se filtren todos los datos personales de los usuarios que utilizan nuestra aplicación. Como se puede observar en las siguientes imágenes, los datos del usuario se cifran al crear la cuenta y se pueden descifrar siempre que el usuario quiera comprobarlos:



```
[
  {
    "nombre": "gAAAAABjYsWLTBbTNp14a6QfZJ33LUj0ZQ__VFn0wQQliyYN1aqqbigo2ELmAc3r8EGI0qGCZd7GiaZdoYTn38tyoBsRGLLbxg==",
    "apellido": "gAAAAABjYsWLOPPQ2zv5Bp1SOqpWCoHiUkstNrneXY3172tyxdAdXDXOXI0pRCszxWUn-Ak7DohGGcB80tlX1P8LiF1iqjzYyA==",
    "fecha": "gAAAAABjYsWLbv0dCP-s1hlVNMpNJ3Xpgq2COqmyLDd50s0_txlXJkNR42bFmoPhax9ZKwgvs0Z7T8WPmF2o-Ow7Pc4ru3mwRA==",
    "DNI": "gAAAAABjYsWLeMruxó3R3dm5hVvqWRrW0B16GQy8fjBftthzH2kPKQBu1B-whhfK6SgYgicUSBrrfRjnxBUPdJ4LkrOIRjWo-Q==",
    "usuario": "Darwin McMuffin"
  }
]
```


Imagen 7: Datos del usuario encriptados



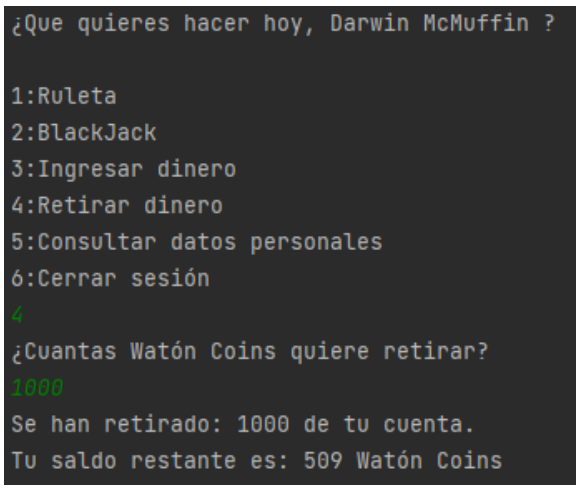
```
1:Ruleta
2:BlackJack
3:Ingresar dinero
4:Retirar dinero
5:Consultar datos personales
6:Cerrar sesión
5
Nombre: Manolito

Apellido: Lama

Fecha: 12-11-2002

DNI: 54218943P

Usuario: Darwin McMuffin
```

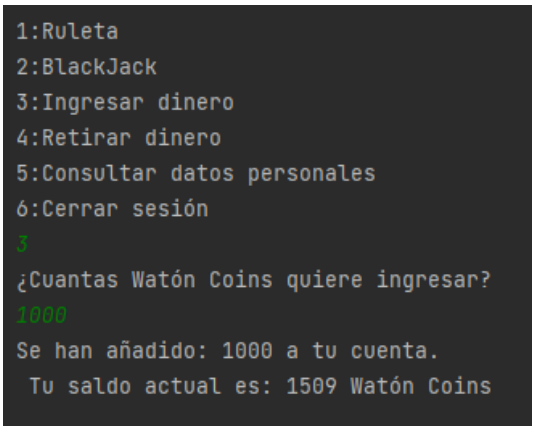

Imagen 8: Datos del usuario al desencriptar

Además, hemos decidido encriptar el dinero de los usuarios de forma que ningún usuario externo pueda comprobar el dinero de cada usuario. De esta forma, siempre que se necesite ingresar o retirar dinero, ya sea manualmente o mediante una apuesta, se desencripta el dinero y se realizan las operaciones necesarias para modificar el saldo del usuario. Esto se puede ver en las siguientes capturas, que corresponden al JSON donde aparece el dinero encriptado del usuario, a una operación de retirada y a una operación de ingreso, respectivamente.

```
[
  {
    "usuario": "Darwin McMuffin",
    "dinero": "gAAAAABjYsXfZKAcbwxCbXuP1uYN2UUDxD72jUMló6NC4jKWjKO8cuRqHWFrzhQwvKjxrXlsFn7IVó5rQD5RWCjWT7yEeEksj4Q=="
  }
]
```


Imagen 9: Dinero del usuario encriptado




Imágenes 10 y 11: Dinero del usuario al ser desencriptado para modificarse

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

En nuestra aplicación utilizamos las funciones hash para cifrar las contraseñas de los usuarios de manera que en ningún caso deban descifrarse para realizar comprobaciones. De esta forma, nos aseguramos de que las contraseñas no puedan ser accedidas por ningún usuario externo pero cada usuario pueda iniciar sesión si lo desea. Para ello se ha utilizado el algoritmo Hashlib SHA256, que genera un hash de 256 caracteres a partir de la contraseña de los usuarios. Así, cuando un usuario intente iniciar sesión, se transformará la contraseña que escribe el usuario a un hash y se comprobará si coincide con el original, que equivale a la contraseña que puso el usuario en el registro. Si coinciden, significa que la contraseña introducida es correcta y se iniciará sesión. Para futuras entregas se ha pensado en ampliar el sistema de forma que se puedan modificar los datos de la cuenta. De esta forma se podrá utilizar el hash para modificar la contraseña, ya que se comprobará con el mismo método si la nueva contraseña coincide con la anterior, lo que sería inválido puesto que la nueva contraseña siempre debería ser distinta a la anterior.

Se ha decidido utilizar el algoritmo Hashlib SHA256, ya que ha sido especialmente optimizado para Python, que es el lenguaje de programación que se está utilizando. Además, las funciones hash tienen un coste computacional muy bajo, por lo que comprobar la contraseña de los usuarios consume muy poco tiempo. Además, resulta muy fácil de utilizar ya que las funciones Hash tienen un tamaño fijo, por lo que la información queda más organizada que con cadenas de tamaño variable, que al cifrar información de gran tamaño puede dar como resultado cadenas de tamaño muy grande.

Como se puede observar en las siguientes imágenes, al realizar un registro se nos guarda un archivo JSON con el usuario y la contraseña transformada con el algoritmo. Al iniciar sesión, cuando el usuario introduce la contraseña, esta se transforma en un hash y se comprueba si es igual al hash original, en cuyo caso iniciará sesión.

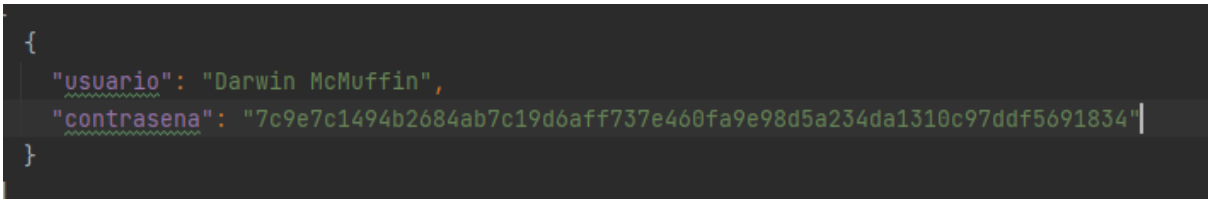



Imagen 12: Hash de la contraseña del usuario

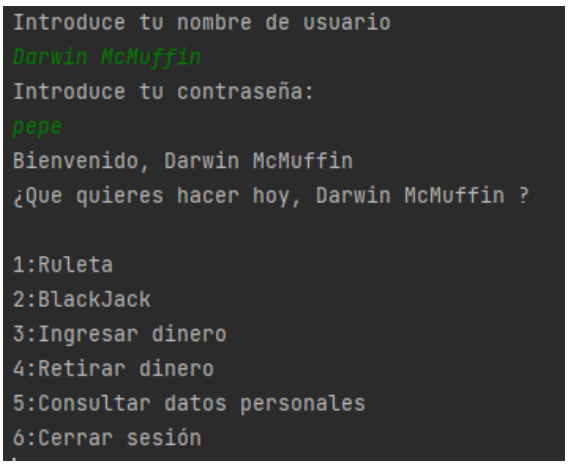


Imagen 13: Inicio sesión del usuario comprobando que los hash sean iguales