DreamWorld Casino

Sofía Campos Oviedo, Fauricio Cespedes Gomez, Jairell Bonilla Ramírez

Universidad Fidélitas

Programación Básica (Introducción a la programación)

Prof. Alvaro Dionisio Camacho Mora

22 de agosto de 2023

Tabla de contenido

Introducción **4**

Objetivos **5**

Objetivo General: 5

Objetivos Específicos: 5

Descripción de los módulos implementados **6**

Avance 1 ..... **6**

Menu 6

User_Registration 7

Index 13

Avance 2 ..... **16**

Registro de Usuario 16

PIN 18

Depósito 19

Autenticación de Usuario 21

Avance 3 ..... **25**

Blackjack 25

Tragamonedas 31

Configuración avanzada 33

Estructuras usadas: **38**

Otros: **38**

Menú Principal: 39

Registro de Usuario Nuevo: 39

DreamWorld Casino:..... 39

Retirar Dinero:.....40

Depositar Dinero:..... 40

Ver Saldo Actual:.....40

Juegos en Línea:..... 40

Eliminar Usuario:..... 40

Configuración Avanzada:..... 40

Manual de Usuario..... 41

Registro de usuario nuevo:..... 41

DreamWorld Casino:..... 42

Retirar Dinero:.....42

Depositar Dinero:..... 42

Ver saldo actual:.....43

Juegos en línea:.....43

Eliminar usuario:..... 46

Salir:.....46

Configuración Avanzada..... 47

Eliminar usuario:..... 47

Modificar Valores del Sistema:.....47

Salir:.....48

Salir del programa:.....48

Conclusiones.....49

Bibliografía.....50

Introducción

En un mundo en constante evolución tecnológica y con la creciente demanda de entretenimiento en línea, la empresa Global Casinos Inc. se encuentra en la búsqueda de una solución que redefina la experiencia de un casino virtual. De aquí nace poder presentar el proyecto de desarrollo del software para mejorar la industria del entretenimiento en línea.

DreamWorld Casino, surge como una respuesta a la necesidad de proporcionar a los usuarios una plataforma de entretenimiento virtual que recree la sensación de un casino real de alta gama, pero a alcance mundial.

El propósito principal del proyecto es crear este casino en línea que capture la esencia y la emoción de los casinos tradicionales, brindando a los jugadores una experiencia auténtica y envolvente desde la comodidad de sus hogares o dispositivos móviles. DreamWorld Casino aspira a ser más que una plataforma de juegos; ya que busca reunir jugadores a nivel mundial

Durante el proyecto de desarrollo, se ha diseñado una plataforma de registro de usuarios que permitirá a los jugadores unirse a la comunidad de DreamWorld Casino de manera segura y sin complicaciones. El software ofrece dos de los juegos más clásicos en el mundo del casino y las apuestas, el “tragamonedas” y el “blackjack”, ambos diseñados para brindar una experiencia de juego realista y emocionante.

Objetivos

Objetivo General:

Desarrollar un programa de software en lenguaje Python que ponga en práctica el conocimiento adquirido durante el curso

Objetivos Específicos:

- Mostrar comprensión sólida de los módulos necesarios a través de los desafíos presentes en el proyecto, tomando en cuenta la estructura del software y algoritmo desarrollado.
- Desarrollar un programa que integre los submódulos vistos, estableciendo una secuencia lógica y cohesiva para satisfacer los requisitos de software.
- Presentar de manera efectiva la defensa del proyecto, demostrando que el enfoque adoptado resuelve de manera precisa el problema y a la vez justificando las diferentes partes del programa con técnicas de comunicación apropiadas.

Descripción de los módulos implementados

Este proyecto se basa en la creación de un casino online llamado “DreamWorld Casino”, que consta de un menú principal con diferentes funcionalidades, como el registro de un usuario, el cual toma información del usuario y posteriormente la guarda, el DreamWorld Casino, en el cual ya se puede jugar directamente, también el programa busca tener una configuración avanzada, donde se puedan cambiar datos internos directamente, como lo puede ser un usuario o el monto mínimo de las fichas.

Avance 1

Se crean tres archivos por aparte donde se almacenan distintas partes del programa:

- **menu**, donde se imprimen las opciones del menú principal y el usuario elige una de las opciones.
- **user_registration**, donde la persona crea su usuario
- **index**, donde según la opcion elegida por el usuario sigue uno de los procesos

Menú Principal

Menu

1. Se crea el método (**def**) **printMainMenu,** donde se imprimen las distintas opciones del menú principal
2. Se crea el método (**def**) **getMenuOption**
    a. Crear variable **menuOption**, donde se le pregunta al usuario cuál opción del menú
    b. Se devuelve **menuOption**

3. Se crea el método (**def**) **isValidMenuOption,** que espera el valor **menuOption**

    a. Se devuelve un boolean “true” si **menuOption** es mayor igual a 1 y menor igual a 4

4. Se crea el método (**def**) **validateMenuOption,** que espera un valor **menuOption**

    a. Se usa un if, que valida si **no** se cumple el método **isValidMenuOption**

        i. Se imprime un mensaje de error

        ii. Se vuelve a correr el menú principal desde el inicio

    b. Si se cumple el método devuelve **menuOption**

5. Se crea el método (**def**) **runMainMenu,** donde se llama a todos los métodos anteriores.

    a. Se llama el método antes creado **printMainMenu**

    b. Se crea la variable **menuOption,** donde se guarda el valor que devuelve **getMenuOption**

    c. Se devuelve la acción o valor de **validateMenuOption,** llevando la variable creada **menuOption**

Registro de Usuario

User_Registration

1. Se importa la biblioteca **getpass**

2. Se crea la variable **userIdAttempts**, donde se van a ir guardando el número de intentos de ingreso de ID

3. Se crea el método (**def**) **validateUserIdAttempts,** donde se van a validar el número de intentos

    a. Se crea la variable **totalUserIdValidAttempts** y se le asigna 3

    b. Se hace global la variable **userIdAttempts,** para que se guarde por fuera del método. "**global**" se utiliza para declarar una variable dentro de una función como una variable global, lo que significa que la variable se puede utilizar tanto dentro de la función como fuera de ella, en todo el ámbito del programa. (Chaudhary, 2023)

    c. Para el caso de nuestro proyecto, estas son las justificaciones de su uso:

    d. Compartir información entre múltiples funciones

    e. Configuración global

    f. Mantener un estado global)

    g. Se le suma 1 a **userIdAttempts,** cuando se invoque a este método

    h. Se usa un **if**, para validar si **userIdAttempts** es igual a **totalUserIdValidAttempts**

        i. Se imprime un mensaje de que ha excedido los intentos de ingreso

        ii. Se importa el método **start** del archivo **index**

        iii. Se invoca el método **start**

    i. Si no se cumple, se llama al método **addRegistration**, el cual es creado al final de este archivo (**user_registration**)

4. Se crea el método (**def**) **isValidUserId,** donde se va a validar el mínimo del ID de usuario, que espera el valor **userId**

    a. Se crea la variable **minUserIdLenght**, el cual debe ser 5

    b. Devuelve un valor booleano, si el tamaño(**len**) de **userId** es mayor o igual a **minUserLenght**

5. Se crea el método (**def**) **validateUserId,** que espera el valor **userId**

    a. Se usa un **if**, para validar el booleano que devuelve **isValidUserId,** llevando el valor **userId**

        i. Si se cumple el mínimo de caracteres y por lo tanto, el método, devuelve el ID **userId**

        ii. Si no devuelve un mensaje de error de “ID de usuario invalido”

        iii. Se llama al método **validateUserIdAttempts**, antes creado para ir midiendo la cantidad de intentos de ingreso de ID

6. Se crea el método (**def**) **getUserId,** donde como tal se le pide al usuario que ingrese su ID.

    a. Se crea la variable **userId,** para guardar el ingreso del ID

    b. Devuelve el resultado del método **validateUserId**, llevando el ingreso del usuario (**userId**)

7. Se crea el método (**def**) **createPIN,** para solicitar al usuario que cree su PIN

    a. Se usa un **while**, para que se repita hasta que el usuario ingrese un PIN correcto y lo devuelva.

i. Se usa un **try,** para solicitar el PIN del usuario

1. Se crea la variable **userPin** que guarda el ingreso del PIN con la biblioteca **getpass** y lo convierte en **string**

2. Se usa un **if** para validar que el tamaño (**len**) del PIN sea mayor o igual a 6.

   a. Devuelve el PIN ingresado **userPin**

3. Con un **else,** si no se cumple se imprime un mensaje de que el PIN debe ser de al menos 6 dígitos

ii. Si el PIN que ingresa tiene algo más que números, entra en el **except**.

1. Muestra un mensaje de "ingresar solo números"

8. Se crea el método (**def**) **authenticatePin,** para confirmar el PIN creado por el usuario y espera el valor **userPin.**

a. Se usa un **while**, para que se repita hasta que el usuario ingrese el mismo PIN creado anteriormente.

i. Se usa un **try,** para solicitar la confirmación del PIN

1. Se crea la variable **confirmPin** que guarda el ingreso del PIN con la biblioteca **getpass** y lo convierte en **int**

2. Se usa un **if** para validar que el PIN creado (**userPin**) sea igual al PIN de confirmación.

   a. Se imprime un mensaje de confirmación de que se creó el PIN con éxito.

b. Devuelve el PIN ingresado **userPin**

3. Con un **else,** si no se cumple se imprime un mensaje de que el PIN no coincide con el creado anteriormente

ii. Si el PIN que ingresa tiene caracteres no númericos, entra en el **except**.

1. Muestra un mensaje de "ingresar solo números"

9. Se crea el método (**def) getUserPin,** donde se obtiene el PIN ingresado por el usuario.

a. Se crea la variable **userPin** que se le asigna lo que devuelve el método **createPIN**

b. Se invoca el método **authenticatePin,** donde se envía el valor de **userPin**

c. Se devuelve el PIN del usuario **userPin**

10. Se crea el método (**def) getDeposit,** donde se va a depositar el dinero inicial mínimo.

a. Se crea la variable global **minMoney**

b. A la variable **minMoney**¸ se le asigna 10000 (como mínimo temporal)

c. Se crea la variable **depositAttempts,** para el numero de intentos y se le asigna 0.

d. Se crea la variable **depositMoney,** para el ingreso de dinero del usuario

e. Se utiliza un **while** que se repite mientras el número de intentos sea diferente o igual a 3 **y** el dinero depositado sea menor al mínimo de dinero.

i. Se usa un **try** para empezar con el ingreso de dinero

1. En la variable **depositMoney,** se guarda el ingreso de dinero del usuario y lo convierte a **float**

2. Se usa un **if** para validar que el dinero depositado sea mayor o igual al mínimo

    a. Se imprime un mensaje de confirmación

    b. Se importa el método **start** del archivo **index**

    c. Invoca al método **start**, para que se devuelva al menú principal

3. Con un **else**, si no se cumple

    a. Se crea la variable **totalDepositValidAttempts** y se le asigna 3

    b. A la variable antes creada **depositAttempts** se le suma 1

    c. Se usa un **if,** donde se valida si el número de intentos hechos por el usuario (**depositAttempts**), es igual al total permitido (**totalDepositValidAttempts**)

        i. Imprime que se ha excedido los intentos

    d. Si no se cumple:

i. Se crea la variable de **attemptsLeft,** que se le asigna la resta de **totalDepositValidAttempts y depositAttempts**

ii. Se imprime que el monto no es válido y los intentos restantes que le quedan al usuario (**attemptsLeft**)

ii. Entra en el **except,** si el usuario ingresa caracteres no numéricos

11. Se crea el método (**def**) **addRegistration,** donde se obtienen los datos de registro del usuario

a. Se imprime un mensaje de Registro de Usuario

b. Se crea una variable global **userId**

c. Se crea una variable global **userName**

d. Se crea una variable global **userPin**

e. A **userId** se le asigna lo que devuelve el método **getUserId**

f. A **userName** se le asigna el ingreso del usuario del nombre

g. A **userPin** se le asigna lo que devuelve el método **getUserPin**

h. Se llama al método **getDeposit**

Index

Index

1. Se importa el método **runMainMenu** del archivo **menú**

2. Se importa el método **addRegistration** del archivo **user_registration**

3. Se crea el método (**def**) **start**, para correr el menú principal

    a. Se crea la variable **menuOption** y se le asigna el método **runMainMenu**

    b. Se usa un **if** para validar si la opción elegida en el menú principal es 1:

        i. Se llama **addRegistration**

    c. Se usa un **elif** validar si la opción elegida en el menú es 2:

        i. Se imprime un mensaje de que la opción no está disponible (temporal hasta que esté terminada la sección)

        ii. Se devuelve al menú principal

    d. Se usa otro **elif** validar si la opción elegida en el menú es 3:

        i. Se imprime un mensaje de que la opción no está disponible (temporal hasta que esté terminada la sección)

        ii. Se devuelve al menú principal

    e. Se usa otro **elif** validar si la opción elegida en el menú es 4:

        i. Se imprime un mensaje de “Saliendo de DreamWorld Casino…”

        ii. Se sale del programa

4. Se invoca al método **start** para que empiece a correr el algoritmo.

Estructuras usadas:

- if, elif, else

- for
- while

Otros:

- try except
- def
- print
- input
- getpass
- sys
- len()
- int()
- float()
- str()

Avance 2

Registro de Usuario

Función getUserInfo()

Retorna un Array con la información del usuario, el objetivo es englobar funcionalidades que se considera están bajo un mismo nivel de abstracción, debido a que recopila los datos del usuario. El orden de los datos retornados en el Array es el siguiente: [userId, userName, userPin]

Función getUsername()

Se encarga de solicitar y retornar el nombre al usuario, en formato de cadena de texto.

Función startUserRegistration()

Función de entrada al módulo, engloba los pilares para el registro de un usuario, compuesto por las funciones que obtienen la información del usuario, obtiene el depósito y añade finalmente el registro en la estructura de archivos, finalmente retorna al menú principal.

Función isUserExist(userId)

Función booleana, recibe por parámetro un Id, verifica si existe o no registro de alguna carpeta con el registro de este ID, de no ser así retorna falso, esto se logra gracias a la librería ‘os’ con su método exists, el cuál recibe la ruta del archivo por parámetro.

Función addRegistration()

Función que engloba otros métodos requeridos para el registro del usuario, como el guardado de la información del depósito y el guardado de información de sus credenciales.

Función setDepositMoney()

Obtiene la información del usuario, como el id, username, pin, posteriormente con la librería ‘os’ se crea el directorio con el identificador del usuario dentro de la carpeta ‘users/’ , para luego generar un archivo llamado ‘saldos.txt’ dentro y guardar ahí el depósito realizado por el usuario, ésto mediante la estructura ‘with’ la cual permite cerrar el archivo abierto pase lo que pase, y también se utiliza el método open() que recibe como primer parámetro la ruta del archivo y como segundo el modo de escritura.

Función setCredentials()

Obtiene la información del usuario, y abre mediante open() el archivo ‘usuarios_pines.txt’, este archivo es el encargado de almacenar las credenciales del usuario, siendo estas el ID y el Pin, al igual que la función setDepositMoney() se utiliza with para cerrar el archivo después de ser usado y write() para escribir dentro del archivo.

Estructuras nuevas

Estructura ‘with’, utilizada para cerrar los archivos abiertos después de su uso, pase lo que pase.

Librerías nuevas

Librería ‘os’, utilizada para crear carpetas, verificar si un archivo existe, en general permite hacer llamadas del sistema operativo.

PIN

Función createPin()

Esta función solicita al usuario que cree su PIN. El PIN debe contener al menos 6 dígitos. Si el usuario ingresa un valor con menos de 6 dígitos, se le pedirá que lo intente nuevamente.

La función entra en un bucle **while True** para asegurarse de que el usuario proporcione un valor válido. Utiliza **getpass.getpass()** para ocultar la entrada del PIN al usuario. (Schmidt, 2019)

Si el PIN ingresado es válido (tiene al menos 6 dígitos), la función devuelve el PIN como string.

Función authenticatePin(userPin)

Esta función autentica el PIN del usuario pidiéndole que lo confirme ingresándolo nuevamente. Recibe el siguiente parámetro:

- **userPin**: El PIN ingresado previamente por el usuario como string.

La función entra en un bucle **while True** para asegurarse de que el usuario proporcione un valor válido. Al igual que en **createPin()**, utiliza **getpass.getpass()**para ocultar la entrada del PIN al usuario.

Si el PIN ingresado coincide con el PIN previamente ingresado (**userPin**), se muestra un mensaje de éxito y se devuelve el **userPin**.

Función getUserPin()

Esta función es la principal y maneja el proceso completo de creación y autenticación del PIN del usuario. No recibe ningún parámetro.

La función llama a **createPin** para que el usuario cree un PIN. Luego, llama a **authenticatePin(userPin)** para autenticar el PIN ingresado.

Si el PIN es autenticado con éxito, se muestra un mensaje de éxito y se devuelve el **userPin**.

Depósito

Método printMenuMoneyType()

Esta función simplemente imprime un menú con tres opciones de depósito: Colones, Dólares y Bitcoin.

Función attemptsDeposit (depositMoney, depositAttempts, type)

Esta función gestiona los intentos de depósito y verifica si el monto ingresado es válido. Recibe los siguientes parámetros:

- **depositMoney**: El monto a depositar.
- **depositAttempts**: El número de intentos de depósito realizados.
- **type**: El tipo de moneda seleccionada para el depósito (1 para Colones, 2 para Dólares, 3 para Bitcoin).

La función primero determina el valor mínimo de depósito según el tipo de moneda seleccionado. Luego, verifica si el monto ingresado es mayor o igual al valor mínimo. Si es así, se considera el depósito como realizado con éxito y se devuelve **depositMoney**, **3** (número máximo de intentos permitidos) y **True** para **flagDeposit**.

Si el monto no alcanza el valor mínimo, la función aumenta el número de intentos **depositAttempts** en 1. Si el número de intentos alcanza el límite máximo (**totalDepositValidAttempts** es igual a 3), se muestra un mensaje y se regresa al menú principal. Si todavía hay intentos disponibles, se muestra un mensaje indicando el monto mínimo requerido y cuántos intentos quedan.

convertMoney: Esta función convierte el monto ingresado a una moneda específica, según el tipo de moneda seleccionado. Recibe los siguientes parámetros:

- **depositMoney**: El monto a depositar.
- **moneyType**: El tipo de moneda seleccionado para el depósito (1 para Colones, 2 para Dólares, 3 para Bitcoin).

La función obtiene los valores de cambio para el Dólar a Colones y el Dólar a Bitcoin a través de la función **helpers.TipoDeCambio**. Luego, realiza la conversión del monto de acuerdo con la moneda seleccionada.

Función getDeposit(depositMoney, depositAttempts)

Esta función es la principal y maneja el proceso de depósito. Recibe los siguientes parámetros:

- **depositMoney**: El monto a depositar.
- **depositAttempts**: El número de intentos de depósito realizados.

La función inicia un bucle que se repite hasta que el número de intentos sea igual a 3. En cada iteración, muestra el menú de opciones de depósito y solicita al usuario que ingrese el número correspondiente al tipo de moneda que desea depositar.

Según la opción seleccionada, se solicita al usuario que ingrese el monto a depositar. Luego, se convierte el monto a dólares utilizando la función **convertMoney**. Después, se llama a la función **depositAttempts** para verificar si el monto ingresado es válido.

Si el depósito se realiza con éxito (**flagDeposit** es **True**), se sale del bucle y se devuelve el **depositMoney** y **True**. Si no, se sigue solicitando al usuario que ingrese un monto válido hasta que se alcancen los tres intentos o se realice un depósito válido

Autenticación de Usuario

Se crean 2 archivos por aparte donde se almacenan 2 nuevas partes del programa:

- **helpers**, donde se van a ir añadiendo los métodos y funciones que usemos repetidamente durante el proyecto.
- **user_authentication**, donde la persona se logea e ingresa a su cuenta
- Se crea un nuevo file llamado “user_authentication”
    - Se importa el archivo “helpers”
    - Se importa la biblioteca “getpass”
    - Se importa el archivo “user_registration”
    - Se crea una variable global “userNameTxt”, para poder usarla y cambiarla en distintas funciones
    - Se crea el metodo “login” para hacer la autenticación del usuario y que pueda ingresar a su cuenta
        - Variable “userIdAttempts” inicializada en 0, para ir guardando los intentos de ingreso de ID

- Variable “userPinAttempts” inicializada en 0, para ir guardando los intentos de ingreso de PIN
- Variable “totalValidAttempts” inicializada en 3, para guardar el número total de intentos
- Variable “userIdTxt” inicializada en “jai”, que guarda un ID temporal para validar el usuario. **En un futuro se va a guardar el ID traido del txt**
- Variable “userNameTxt” inicializada en “Jairell”, que guarda un nombre temporal. **En un futuro se va a guardar el nombre del usuario traido del txt.**
- Variable “userMoneyTxt ” inicializada en “10000”, que guarda una cantidad de deposito temporal. **En un futuro se va a guardar el depósito traido del txt.**
- Variable “userPinTxt” inicializada en “123456”, que guarda un nombre temporal. **En un futuro se va a guardar el nombre del usuario traido del txt.**
- Se implementa un while que se repita mientras que los intentos de ingreso de ID sea diferente a 3
    - Se crea variable “userId” para pedirle al usuario que ingrese su ID

- Se usa un if para validar que el “userId” ingresado, sea igual “userIDTxt” que vendra desde el txt en un futuro (por el momento debe ser igual a “jai”)
    - Se implementa otro while para que se repita mientras que los intentos de PIN sea diferente a 3
        - Se crea la variable “userPin” para que el usuario digite su PIN.
        - Se usa otro if para validar que el pin ingresado sea igual al del txt:
        - Si se cumple se llama al metodo “**menuCasino”**
        - Se devuelve el el userId, userPin, userNameTxt, userMoneyTxt, para guardarlos luego
        - Si no se cumple, else:
        - Se le suma y asigna 1 a “userPinAttempts”
        - Se usa otro if para validar la cantidad de intentos, si los intentos de PIN son iguales al total de intentos antes planteados (3)
            - Se imprime un mensaje que muestra que se han excedido los intentos de ingreso de PIN

- Se llama a la funcion de helpers, “returnToMainMenu”, para volver al menu principal

- Si no se cumple, else

- Se crea la variable “attemptsLeft” para tener la cantidad entre la resta de 3 y el numero de intentos de PIN

- Se imprime un mensaje de que el dato ingresado no es valido y el numero de intentos restantes (attemptsLeft)

- Si no se cumple la igualdad entre los intentos de ID y el número total de intentos (3):

- Se le suma y asigna 1 a los intentos del usuario de ID

- Se implementa un if que valide la cantidad de intentos del usuario de ID y la cantidad total

- Se imprime un mensaje de que ha excedido los intentos de ID

- Se llama a la funcion de helpers, “returnToMainMenu”, para volver al menu principal

- Si no se cumple, else

        - Se crea la variable “attemptsLeft” para tener la cantidad entre la resta de 3 y el número de intentos de ID
        - Se imprime un mensaje de que el dato ingresado no es valido y el número de intentos restantes (attemptsLeft)

    - Se crea el método “menuCasino”, donde se van a presentar las opciones del DreamWorld Casino
        - Se imprime un mensaje de bienvenida
        - Se implementa un while
            - Se imprime la primera opcion del menú (“Retirar Dinero”)
            - Se imprime la segunda opcion del menú (“Depositar Dinero”)
            - Se imprime la tercera opcion del menú (“Ver saldo actual”)
            - Se imprime la cuarta opcion del menú (“Juegos en línea”)
            - Se imprime la quinta opcion del menú (“Eliminar usuario”)

Avance 3

Blackjack

Función start(id, pin, name)

Es la función de arranque del blackjack, recibe tres parámetros los cuales son los datos del usuario, esta función se encarga de ejecutar los procesos principales del juego, así como el reseteo de variables, la muestra de instrucciones, la apuesta y finalmente el juego.

Función resetGlobalValues(id, pin, name)

Requerida debido a que cuando el usuario quiera volver a jugar todos los valores utilizados en la partida anterior deben estar inicializados en sus valores por defecto para evitar errores.

Función showInstructions()

Muestra en pantalla las instrucciones del Blackjack

Función amountToBetHandle()

Manja la cantidad de dinero que el usuario va a apostar, llama para recopilar la cantidad y llama para setearla en el sistema.

Función getAmountToBet()

Retorna el monto indicado por el cliente para ser apostado.

Función getValidAmountFormat()

Verifica que el monto ingresado por cliente tenga un formato correcto, es decir que el usuario no ingresa letras ni símbolos no válidos para una cantidad

Función checkAmountToBet(amountToBet)

Verifica lo siguiente:

1. Que el saldo del usuario sea mayor al monto mínimo de apuesta.
2. Que el usuario tenga el dinero para poder apostar lo que dice apostar.
3. Que lo que va a apostar sea mayor al monto mínimo de apuesta.

Función hasPlayerEnoughMoney(amountToBet, playerMoney)

Función booleana que verifica que el usuario tiene en su saldo el monto que dice querer apostar.

Función play()

Función principal del juego, hace un llamado para que el crupier reparta las cartas.

Función assignCards()

El crupier reparte las cartas, con la secuencia de que la primera es para el usuario, la siguiente para el crupier, la siguiente el usuario, y la última el crupier, donde la primer carta del crupier la muestra como oculta.

Función getRandomCard(user)

Genera una carta aleatoria y la retorna, por ejemplo “3 de Corazones Rojo”.

Función getAsCardValue()

Cuando a un usuario le sale de carta un As automáticamente se le pregunta qué valor desea darle a ese As, si un 1 o un 11.

Función isValidAsCardOption(asCardOption)

Valida que la opción que haya ingresado sea “1” u “2”, debido a que hay que decir 1 para que valga 1 y 2 para que valga 11.

Función arePlayerCardsEqual(playerCards)

Función booleana que valida si las cartas recibidas por el usuario son iguales. (Para posteriormente preguntarle si desea dividir en manos separadas en caso de ser verdadero).

Función askToDobuleBet()

Le pregunta al usuario si desea doblar su apuesta.

Función doubleBet()

Actualiza la apuesta del usuario con su apuesta inicial multiplicada por dos.

Función askToDivideCards()

Si obtuvo dos cartas iguales, el sistema le pregunta al usuario si desea dividir sus cartas.

Función isValidDivideCardsOption(option)

Valida si la opción ingresada en la función anterior es “1” para sí o “2” para no.

Función divideCards()

Divide las cartas en dos manos diferentes, (Ahora playerCards es una matriz donde cada “fila” representa una mano)

Función checkScore(playerHand)

En caso de que la suma de las cartas del usuario sea mayor o igual a 21 ejecuta un llamado a la función que revisa si es o no es un blackjack.

Función checkBlackjack(playerHand)

Esta es una de las funciones principales, valida 4 escenarios:

1. Que el usuario se pase de 21, en este caso pierde su apuesta, o su mano si es que dividió previamente.
2. El crupier saca cartas hasta obtener una puntuación igual o mayor al jugador, para validar los siguientes 3 escenarios:
3. Empate en puntuaciones: el usuario no pierde ni gana dinero.
4. El crupier se pasó de 21: en este caso el jugador gana su apuesta multiplicada por dos.

5. El crupier sacó más que el jugador pero menos o igual a 21: el jugador pierde su apuesta.

Función getCardValues(cards)

Retorna el valor de todas las cartas del usuario, por ejemplo si tiene una J le va a devolver que vale 10.

Función removeHand(playerHand)

Si el usuario dividió cartas y perdió o empató con esa mano, entonces esa mano será eliminada y no podrá volver a jugar con ella.

Función stayPlayingOrReturn()

Pregunta al usuario si quiere volver a jugar blackjack o si desea volver al menu del casino.

Función menu()

Maneja todo el funcionameinto del menu, como mostrarlo en pantalla, administrar cada opción ingresada etc…

Función printMenu()

Muestra en pantalla las opciones del menú del blackjack, como ver mis cartas, pedir carta, ver carta del crupier o detenerse.

Función getMenuOption()

Solicita una opción del menú al usuario y la retorna.

Función isValidMenuOption(option)

Valida si la opción se encuentra en el menú, es decir si es “1”, “2”, “3” o “4”, retorna True o False.

Función handleMenuOption(option)

Recibe una opción del menú y en caso de ser “1” ejecuta la función requestNewCard(), en caso de ser “2” ejecuta la función stand(), en caso de ser “3” ejecuta la función printPlayerCards() y en caso de ser “4” ejecuta la función printCrupierCards()

Función requestNewCard()

Se valida si el usuario tiene cartas divididas, de ser así se le pregunta a cuál mano desea pedir una nueva carta, de no ser así simplemente se le agrega una carta a su única mano mediante la función addNewCardToHand(), finalmente para ambos procesos se ejecuta checkScore()

Función getHand()

Esta función sólo se ejecuta en caso de que el usuario haya dividido sus cartas, donde se le consulta cuál mano desea usar y se retorna esa mano.

Función getHandOption()

Valida que la opción ingresada por el usuario sea válida, es decir “1” para la Mano 1 y “2” para la mano 2, finalmente retorna la opción.

Función isValidHandOption(option)

Función boolena que retorna True si la opción ingresada es “1” o “2”.

Función addNewCardToHand(playerHand)

Llama a la función getRandomCard(), se agrega esa carta a la mano del usuario y se le muestra al usuario en pantalla cuál carta sacó.

Función stand()

Si el usuario decide detenerse se ejecuta esta función, donde si tiene cartas divididas se valida la primer mano ejecutando el checkBlackjack() y si no sólo se ejecuta checkBlackjack() con la única mano del usuario.

Función printPlayerCards()

Valida primero si el usuario tiene manos divididas, de ser así ejecuta printTwoHands() y si no printTheOnlyHand().

Función printTheOnlyHand()

Mediante un for se recorre el array playerCards, y se le van mostrando las cartas al usuario, finalmente se le muestra la sumatoria de sus cartas.

Función printTwoHands()

Mediante dos for (anidados) se recorre la matriz donde cada fila corresponde a una mano del usuario, se muestra en pantalla cada carta de cada mano y para cada mano también se muestra la sumatoria de sus cartas.

Función printCrupierCards()

Mediante un for se recorre el array crupierCards, donde se muestran las cartas del crupier a exepción de la primera, ya que la primera carta que se reparte el crupier está oculta.

Tragamonedas

1. *Importación de módulos y funciones:*

    a. El código importa varios módulos y funciones utilizando las declaraciones **import**. Los módulos incluyen las bibliotecas **helpers**,

getpass, **os**, **random**, **time** y los archivos **user_authentication** y **menu_casino**.

2. *Definición de funciones:*

    a. Función start()

        i. Esta función toma tres argumentos: id, pin y name. Esta función es el punto de entrada para el juego de tragamonedas.

        ii. Muestra las instrucciones básicas del juego de tragamonedas al usuario utilizando la función print(). Estas instrucciones describen los pasos para jugar el juego y las reglas asociadas.

        iii. La función llama a la función **getMoney(userId)** del módulo **menu_casino** para obtener el saldo del usuario. Si el saldo es inferior a la apuesta mínima (**minBet**), muestra un mensaje de saldo insuficiente y redirige al usuario de vuelta al menú principal.

        iv. Si el saldo es suficiente para realizar una apuesta, la función llama a la función **game()** para iniciar el juego.

    b. Función game():

        i. Inicia un bucle **while** con la variable **flag** para permitir que el jugador juegue repetidamente hasta que decida dejar de jugar.

        ii. El jugador puede introducir la cantidad de dinero que desea apostar. Además, se verifica si el jugador tiene suficiente saldo para realizar la apuesta.

iii. Después de establecer la apuesta, el jugador presiona "Enter" para tirar de la palanca y ver los resultados.

iv. Dependiendo del número de intentos que lleve el usuario (**totalAttemptsSlots**), se muestra una secuencia específica de símbolos en los carretes o una combinación aleatoria con la biblioteca **random**. (Lozano, 2023)

v. Se evalúa si los símbolos en los carretes forman una combinación ganadora. Si es así, se actualiza el saldo del usuario según la tabla de pagos y se muestra un mensaje correspondiente.

vi. Se da la opción al jugador de jugar nuevamente o salir del juego.

vii. En el bucle **while** de la función **game()**, si el jugador elige no continuar jugando, la variable **flag** se establece en **False** y la función **user_authentication.menuCasino** es devuelta

Configuración avanzada

Importación de módulos:

- **os:** Este módulo proporciona funciones para interactuar con el sistema operativo, como manipular archivos y directorios.
- **getpass:** Este módulo permite ocultar la entrada de contraseñas y otros datos sensibles.
- **helpers:** Nuestra biblioteca helpers donde se encuentran funciones utilizadas frecuentemente.

Definición de constantes y variables:

- **config_file:** Cadena que almacena el nombre del archivo "configuracion_avanzada.txt".
- **arrayConfAvanzada:** Se llama a la función confAvanzada() de la clase helpers para obtener la lista de configuraciones avanzadas.

Función deleteUser(id):

1. Esta función toma el id del usuario.
2. Primero se construye la ruta a la carpeta del usuario basada en el id.
3. Se verifica si existe la carpeta del usuario y si existe, se procede a eliminar sus archivos y subcarpetas.
4. Luego se elimina la carpeta principal del usuario.
5. Se abre el archivo "usuarios_pines.txt" y se eliminan las líneas correspondientes al usuario que se está eliminando.
6. Finalmente, se imprime un mensaje de éxito o fracaso.

Función modifySystemValues():

1. Esta función imprime los valores actuales de las configuraciones avanzadas y luego permite al usuario seleccionar un valor para modificar.
2. Si la selección es válida, se solicita al usuario que ingrese el nuevo valor.
3. Luego, se actualiza el archivo "configuracion_avanzada.txt" con los nuevos valores.

Función advancedSettings():

1. Esta función controla el acceso a las configuraciones avanzadas del programa.
2. Lee un PIN especial del archivo "usuarios_pines.txt".
3. Solicita al usuario ingresar un PIN y compara con el PIN leído.

4. Si el PIN es correcto, muestra un menú de opciones relacionadas con las configuraciones avanzadas.
5. Si elige la opción 1, solicita el ID de un usuario y llama a la función deleteUser() para eliminar al usuario.
6. Si elige la opción 2, muestra un submenú para modificar diferentes valores del sistema.
7. Actualiza los valores y los guarda en el archivo "configuracion_avanzada.txt".
8. Si elige la opción 3, sale de la configuración avanzada y vuelve al menú principal.

Retirar Dinero:

1. La función withdrawMoney(id) es llamada.
2. Se inicializa el contador depositAttempts para rastrear el número de intentos de retiro.
3. Se obtiene el saldo actual del usuario con el id proporcionado utilizando la función getMoney(id) y se almacena en la variable balance.
4. Se muestra el saldo actual al usuario.
5. Comienza un bucle while que permite al usuario intentar retirar dinero hasta tres veces (o menos si se realiza un retiro exitoso).
6. En cada iteración del bucle, se solicita al usuario que ingrese la cantidad que desea retirar.
7. Se verifica si la cantidad ingresada es menor o igual al saldo disponible.
    a. Si es así, se calcula el nuevo saldo restando la cantidad ingresada al saldo actual, se actualiza el saldo llamando a updateMoney(id, newBalance), y se muestra un mensaje de retiro exitoso con el nuevo saldo.
    b. Si no es así, se incrementa depositAttempts y se muestra un mensaje de fondos insuficientes, indicando cuántos intentos quedan antes de llegar a tres.
    c. En ambos casos de intento de retiro (éxito o fallo), se utiliza la instrucción return para salir de la función.

8. Si el usuario excede los tres intentos, se muestra un mensaje indicando que se ha superado el límite y se llama a la función helpers.returnToMainMenu() para volver al menú principal.

Depositar Dinero:

1. La función depositMoney(id) es llamada.
2. Se inicializa el contador moneyTypeAttempts para rastrear el número de intentos de selección de tipo de moneda.
3. Se muestra el saldo actual del usuario.
4. Comienza un bucle while que permite al usuario intentar seleccionar el tipo de moneda hasta tres veces (o menos si se realiza un depósito exitoso).
5. En cada iteración del bucle, se muestra un menú de opciones de tipos de moneda mediante printMenuMoneyType(id) y se solicita al usuario que ingrese el número de opción correspondiente.
6. Se verifica si la opción ingresada es válida (1, 2 o 3).
    a. Si es válida, se llama a la función processDeposit(id, moneyType) para continuar con el proceso de depósito.
    b. Si no es válida, se incrementa moneyTypeAttempts, se muestra un mensaje indicando cuántos intentos quedan antes de llegar a tres.
    c. En ambos casos de intento de selección de moneda (éxito o fallo), se utiliza la instrucción return para salir de la función.
7. Si el usuario excede los tres intentos, se muestra un mensaje indicando que se ha superado el límite y se llama a la función helpers.returnToMainMenu() para volver al menú principal.

Eliminar Usuario:

1. La función deleteUser(id, pin, name) es llamada con el id del usuario, su pin y su nombre.
2. Se solicita al usuario que ingrese su PIN para confirmar la eliminación de la cuenta.
3. El PIN ingresado por el usuario se compara con el PIN almacenado en la variable pin.
    a. Si coincide, se verifica si el saldo del usuario es igual a cero.
        i. Si el saldo es cero, se solicita una confirmación adicional para eliminar la cuenta.
            1. Si el usuario confirma, se llama a la función deleteFilesAndFolder(id) para eliminar los archivos y la carpeta del usuario, y se devuelve True.
    b. Si el saldo no es cero, se muestra un mensaje indicando que no se puede eliminar la cuenta con saldo disponible.
4. Si el PIN no coincide, se muestra un mensaje indicando que la operación de eliminación se cancela debido a un PIN incorrecto, y se devuelve False.

Estructuras usadas:

- if, elif, else
- for
- while

Otros:

- try except
- def
- print
- input
- getpass
- os
- len()
- int()
- float()
- str()
- global

Requerimientos del Sistema

Menú Principal:

- Presenta cuatro opciones para que el usuario elija: Registro de usuario nuevo, DreamWorld Casino, Configuración Avanzada y Salir. Cada opción corresponde a funciones independientes dentro del programa.

Registro de Usuario Nuevo:

- Creación de ID de usuario (mínimo 5 caracteres alfanuméricos).
- El ID no debe estar registrado previamente en los archivos.
- Se permiten hasta tres intentos para ingresar un ID válido.
- Solicitud de nombre, para guardarlo junto a su ID.
- Escogencia y validación del PIN del usuario, el cual usaría para iniciar sesión en su cuenta (máximo 6 dígitos).
- Depósito obligatorio en moneda dólar, colón o bitcoin.
- Guardado de la información y creación de archivos asociados al usuario.

DreamWorld Casino:

- Carga de información de usuarios desde el archivo de texto antes creado con todos los datos.
- Autenticación de usuarios mediante su ID y PIN.
- Submenú con opciones para retirar dinero, depositar dinero, ver saldo, jugar juegos en línea y eliminar usuarios.

Retirar Dinero:

- Verificación del saldo actual del usuario y retiro de dinero.
- Control de intentos y mensajes en caso de errores.

Depositar Dinero:

- Depósito en diferentes monedas (conversión si es necesario dependiendo de cada moneda). Verificación de saldo mínimo y control de intentos.

Ver Saldo Actual:

Muestra el saldo actual del usuario en dólares.

Juegos en Línea:

- Elección de juego (Blackjack o Tragamonedas).
- Instrucciones y reglas del juego.
- Apostar y jugar.
- Control de intentos y mensajes en caso de errores.
- Actualización de saldos dentro de los archivos de texto y posibilidad de jugar nuevamente.

Eliminar Usuario:

Eliminación de usuario, validando el usuario con su respectivo PIN.

Configuración Avanzada:

- Acceso con PIN especial de administrador, que solo el equipo conoce.
- Opciones para eliminar usuarios y modificar valores del sistema.

Manual de Usuario

Al correr el código te encontrarás con un menú que te brinda las principales opciones del casino:

1) **Registro de usuario nuevo**
2) **DreamWorld Casino**
3) **Configuración Avanzada**
4) **Salir**

Debes ingresar un número dependiendo a la opción que desees realizar.

Registro de usuario nuevo:

Dentro del módulo de **registro de usuario nuevo** te va a pedir que ingreses tu ID de usuario (es importante tomar en cuenta que debe contener mínimo cinco caracteres).

Luego, te va a pedir que ingreses tu nombre (sin mínimo o límite de caracteres).

También debes ingresar tu número de PIN, el cual usarás como contraseña para iniciar sesión. De igual forma, tienes que volver a ingresar el mismo PIN para confirmar que sea el correcto (ambos pines no van a salir en pantalla, para proteger tu seguridad).

Ahora que ya creaste tus credenciales, debes realizar un depósito mínimo, donde primero tienes que elegir un número que determina el tipo de moneda con la que deseas depositar. Se te va a pedir que ingreses el monto a depositar en tu cuenta (el mínimo a depositar se te desplegará en pantalla con el tipo de cambio correcto).

Una vez ingreses el depósito tu usuario será registrado con éxito.

DreamWorld Casino:

Dentro del módulo de **DreamWorld Casino** te va a pedir que ingreses tu ID de usuario anteriormente creado en el **registro de usuario nuevo,** el cual debe coincidir. A continuación se te va a pedir tu PIN para poder iniciar sesión en tu cuenta.

Se te desplegarán distintas opciones, donde debes ingresar un número para especificar la opción que quieres:

1) Retirar Dinero
2) Depositar Dinero
3) Ver saldo actual
4) Juegos en línea
5) Eliminar usuario
6) Salir

Retirar Dinero:

Se le mostrará la cantidad de dinero (**en dólares**), que posee actualmente en su cuenta. Luego verá una cajita donde debe ingresar el dinero que desea retirar. **Por favor tome en cuenta que este monto a dinero debe ser ingresado de igual forma en dólares.**

Depositar Dinero:

Se le mostrará la cantidad de dinero (**en dólares**), que posee actualmente en su cuenta. Debe elegir el tipo de moneda con el que desea depositar, ingresando el número que corresponde a cada una.

Luego, se le pedirá el monto a depositar en el tipo de moneda elegido. **En este caso no hay un monto mínimo que debas depositar.**

Ver saldo actual:

Simplemente te desplegará el monto actual de tu cuenta **en dólares.**

Juegos en línea:

Se le desplegará la lista de juegos disponibles, donde deberá ingresar el número de juego que desee jugar:

1) Blackjack

2) Tragamonedas

Blackjack.

Una vez elija la opción para jugar "blackjack" se muestra en pantalla las instrucciones del juego, seguidamente deberá tomar en cuenta lo siguiente:

1. **Introducir apuesta:** Digita el monto que desea apostar, debe cumplir con el monto mínimo y debe tener el dinero suficiente.

2. **Repartición de cartas:** El crupier empieza a repartir las cartas, la primera para el jugador, la segunda para él pero es oculta, la tercera para el jugador y la última para él.

3. **Doblar apuesta:** Se le consulta al usuario si desea doblar su apuesta, es decir apostar el doble de lo que apostó inicialmente.

4. **Menú del Blackjack:** Ahora se muestra en pantalla el menú del blackjack con cuatro opciones: 1. Pedir carta, 2. Deseo parar, 3. Consultar mis cartas, 4. Consultar carta del Crupier.

5. **Pedir Carta:** Si seleccionó la opción de pedir carta el sistema le agregará una nueva carta a su mano, si la sumatoria de sus cartas excede 21 usted perderá su apuesta, si es 21 se validará con las cartas del crupier si se trata de una victoria o un empate. Volverá al menú del blackjack y podrá pedir cuantas cartas quiera sin exceder 21.

6. **Consultar mis cartas:** Se le mostrará en pantalla todas sus cartas y la sumatoria de las mismas. Volverá al menú del blackjack.

7. **Consultar carta del crupier:** Se mostrará sólo la segunda carta del crupier ya que la primera está oculta. Volverá al menú del blackjack.

8. **Deseo parar:** Se muestran los resultados de la partida, el crupier empieza a tomar cartas hasta igualar o superar la puntuación del usuario, si se excede de 21 el usuario gana y gana el doble de lo que apostó, existe el empate donde no se pierde ni se gana nada, y la derrota donde el usuario pierde lo que apostó. Aquí termina el juego.

9. **Seguir jugando:** Se le consulta al usuario si desea volver a jugar blackjack o si desea volver al menú del casino.

Tragamonedas.

Una vez elija el juego de “tragamonedas”, se le mostrarán todas las instrucciones del juego a tomar en cuenta antes de jugar:

1. Selecciona tu apuesta: Antes de comenzar a jugar, debes elegir cuánto dinero estás dispuesto a apostar en cada tirada.

2. Introduce el dinero: Debes digitar la cantidad de dinero que quieras apostar, el dinero que apuestes es el que tengas depositado en tu cuenta del casino con anterioridad.

3. Tira de la palanca: Una vez que hayas establecido tu apuesta, debes oprimir la tecla "Enter" para empezar

4. Espera a que los carretes se muestran: Los carretes van a ir apareciendo y luego se detendrá de manera aleatoria. Cada carrete contiene símbolos diferentes.

5. Verifica las combinaciones: Una vez que los carretes se detengan, se verificarán las combinaciones de símbolos. Si los símbolos forman una combinación ganadora según las reglas del juego, ganarás un premio.

6. Cobra tus ganancias: Si obtuviste una combinación ganadora, se te otorgará un premio en función de la tabla de pagos del juego y tu apuesta. Tus ganancias se depositarán en tu salario del casino.

7. Sigue jugando: Puedes continuar jugando, ajustando tus apuestas y repitiendo los pasos anteriores. Recuerda que el juego es completamente aleatorio, y las probabilidades de ganar pueden depender de la cantidad de veces que juegues.

Se te desplegará el saldo actual de tu cuenta del casino y además la cantidad de la apuesta mínima a realizar.

Se le pedirá que digite la cantidad de dinero que desea apostar, la cual debe cumplir con el mínimo.

Luego debe presionar la tecla enter en su teclado para jalar la palanca del tragamonedas y empezar a jugar. A continuación se le irá desplegando el resultado caracter por caracter.

Al final de cada tirada, se le preguntará si desea jugar nuevamente. Deberá ingresar “S” para continuar con una nueva tirada o “N” si se desea retirar del juego y volver al menú principal.

Eliminar usuario:

Se le solicitará su PIN de la cuenta para confirmar que en verdad desea eliminar la cuenta y no ocurra ningún problema.

A continuación se borrarán todos los datos de su cuenta. **Es importante mencionar, que si su cuenta tiene dinero disponible, no se realizará la eliminación del usuario, debe retirar o jugar su saldo antes.**

Salir:

Se procederá a volver al menú principal del casino.

Configuración Avanzada.

Se le pedirá el PIN especial para entrar a la configuración avanzada del sistema.

Este PIN especial solo lo tienen el equipo de desarrollo

A continuación se le desplegará el menú de configuración avanzada, donde debe ingresar el número de la opción que desee realizar:

1. **Eliminar Usuario**
2. **Modificar Valores del Sistema**
3. **Salir**

Eliminar usuario:

Se le solicitará el ID de la cuenta que desea eliminar del registro. A continuación se borrarán todos los datos de la cuenta ingresada.

Modificar Valores del Sistema:

Se le mostrará el menú de opciones que tienes para modificar en el archivo de configuración avanzada

1. **Valor mínimo depósito inicial**
2. **Tipo de cambio: valor del colon**
3. **Tipo de cambio: valor del bitcoin**
4. **Acumulado del tragamonedas**
5. **Apuesta mínima tragamonedas**
6. **Apuesta mínima blackjack**

Para cada uno de las anteriores opciones, se le pedirá que digite el nuevo valor y se le mostrará un mensaje de confirmación

Salir:

Se procederá a volver al menú principal del casino.

Salir del programa:

Se procederá a terminar la ejecución del programa.

Conclusiones

- El resultado del proyecto ha demostrado una comprensión de los módulos aprendidos, necesarios para el desarrollo del software.
- A través de los requisitos del proyecto, se ha evidenciado la aplicación y práctica de los conocimientos adquiridos durante el curso de manera efectiva.
- La secuencia lógica del programa desarrollado refleja un enfoque ordenado en el diseño del proyecto. El software sigue un flujo de ejecución claro, importante para entregar un producto de calidad.
- Por medio de la defensa, se proporciona una justificación apropiada para sustentar los diferentes módulos y requerimientos del software
- Se proporciona un resultado de casino virtual a la compañía Global Casinos Inc., el cual cumple con los requerimientos de la empresa que solicitaba implementar una idea a nivel mundial para su casino, libre para sus jugadores a nivel mundial

Bibliografía

Chaudhary, S., (30 de enero de 2023). *Variables globales y cómo cambiar desde una función en Python.* Obtenido de https://www.delftstack.com/es/howto/python/global-variables-in-python-and-how-to-change-them-from-a-function/

Schmidt, E., (2019). *getpass - Solicitud segura de contraseña.* Obtenido de https://rico-schmidt.name/pymotw-3/getpass/index.html

Lozano, J., (2023). *Generar números aleatorios en Python.* Obtenido de https://j2logo.com/python/generar-numeros-aleatorios-en-python/