Ingeniería en Computación
Taller de Programación
Semestre 2, 2022
Profesor: Ing. Cristian Campos Agüero

TEC | Tecnológico de Costa Rica

Proyecto Programado #1
Quién quiere ser Millonario

1. Introducción

¿Quién quiere ser millonario? (en inglés, *¿Who Wants to Be a Millionaire?*) es un concurso de televisión basado en el formato de preguntas y respuestas, que ofrece grandes premios monetarios por responder correctamente a una serie de preguntas de opción múltiple con dificultad creciente. El formato del programa es licenciado por *Sony Pictures Television International*. El premio mayor en la versión original para Reino Unido es un millón de libras. La mayoría de las versiones internacionales ofrecen un premio mayor de un millón de unidades de la moneda local; el valor real del premio varía ampliamente, dependiendo del valor de la moneda.

Fuente: https://es.wikipedia.org/wiki/%C2%BFQui%C3%A9n_quiere_ser_millonario%3F

2. Software por desarrollar

Su trabajo consiste en implementar una aplicación de escritorio para administrar el juego utilizando el lenguaje **Python**.

El sistema inicia con un **Menú Principal** donde se puede ingresar a las:

- (A) Opciones administrativas,
- (J) Opciones de jugador
- (S) Salir.

Si selecciona ingresar a las opciones administrativas o normales se desplegará un menú para las diferentes opciones, y dentro de esos menús de podrá devolver al Menú Principal.

3. Opciones Administrativas

Para acceder a estas funcionalidades el usuario deberá ingresar por medio del **Menú Principal** y proporcionar un **nombre de usuario** y **clave** de acceso (**debe estar generada y guardada en disco, el archivo se llamará Acceso.txt**) y se deben habilitar las siguientes funcionalidades (Menú Administrativo):

- Gestión de Preguntas y Respuestas
- Gestión de Juegos
- Historial juegos
- Estadísticas de juegos
- Retornar

El archivo debo debe contener al menos 3 credenciales de acceso, bajo el siguiente formato; **usuario;clave**, por ejemplo:

pperez;1234
ccastro;admin1234
mporras;mporras1

3.1. Gestión de Preguntas

El sistema debe permitir dar mantenimiento de las preguntas y respuestas, se debe permitir **incluir, eliminar o modificar cada una de ellas, además de mostrarlas**. La información que se debe almacenar será:

- *Identificador (Autogenerado)*
- Pregunta,
- 3 posibles respuestas incorrectas
- La respuesta correcta

No pueden existir preguntas iguales de la misma manera identificadores repetidos y no pueden **eliminarse** preguntas que hayan sido asociados a algún juego.

Debe existir un archivo llamado "**Preguntas.txt**" y el formato interno debe ser de la siguiente forma:

1,PreguntaA,Opcion1,Opcion2,Opcion3,RespuestaCorrecta
2,PreguntaB,Opcion1,Opcion2,Opcion3,RespuestaCorrecta

3.2. Gestión de Juegos

El sistema debe crear un nuevo juego y debe seleccionar un conjunto de preguntas aleatorio del archivo "**Preguntas.txt**", específicamente **15 preguntas**, estas deben guardarse en un archivo por aparte.
El archivo llamado "**ListaPreguntasXX.txt**", donde **XX** será un **número consecutivo**, debe contener:

- Identificador (autogenerado)
- Fecha
- Hora
- La lista de las 15 preguntas

Su formato interno debe ser de la siguiente manera:

Contenido del archivo **ListaPreguntas1.txt**

1,Fecha,Hora
100,PreguntaA,Opcion3,Opcion1,**Respuesta**,Opcion2,**3**
.
.
.
15000,PreguntaN,Opcion2,**Respuesta**,Opcion3,Opcion1,**2**

La escala de los **premios** será de la siguiente manera; 100, 200, 300, 400, **500**, 1000, 2000, 2500, 4000,**5000**, 7000, 8500, 9500, 12000, **15000.** Las cantidades están expresadas en **miles colones**, además los **montos** resaltados en **Negrita** son **zonas seguras** y le da la posibilidad al jugador de poder retirarse con dicho monto ganado. No pueden **eliminarse** juegos que hayan sido asociados a algún **jugador**. Si se desea agregar o modificar el juego actual, este debe **volver a generarse**.

Debe existir un archivo llamado **IndiceJuegos.txt**, en este estará los nombres de los archivos "**ListaPreguntasXX"** creados. En la imagen adjunta se describe la estructura y distribución de los archivos a crear y sus dependencias.

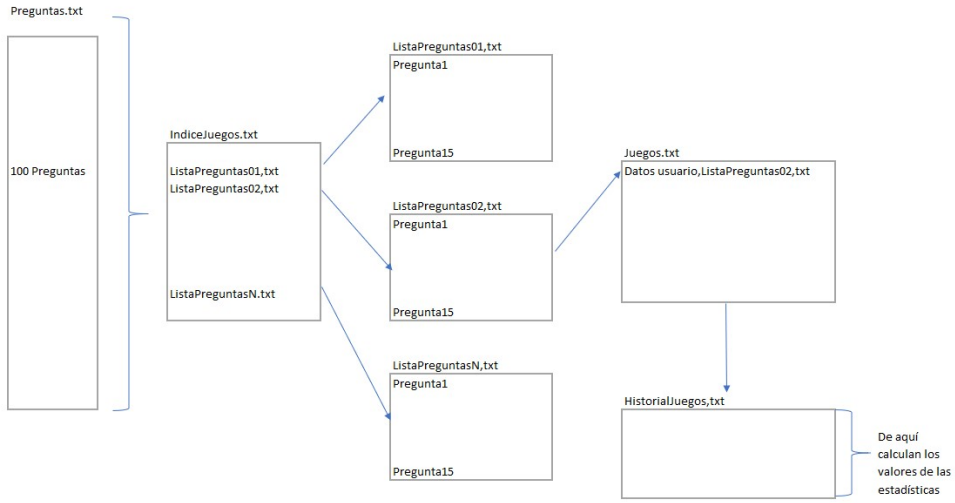

Ilustración 1 Archivos a crear par el juego

3.3. Historial de Juegos

Mostrará la lista de los Juegos realizados en la aplicación, por cada juego se mostrará lo siguiente:

- Cédula
- Nombre de la persona quién jugó
- Sexo de la persona

- Fecha y hora de inicio del juego
- Fecha y hora de finalización del juego
- Archivo de juego (ej: “ListaPreguntas12.txt”)
- Premio obtenido
- Total de preguntas correctas

Los datos anteriores deben guardarse **por línea** dentro del archivo. Además, se debe permitir filtrar la información por:

- Rango de fecha,
- Por de Nombre del jugador o parte de este,
- Por premio

Para esto anterior deben implementar un menú para seleccionar qué tipo de filtro. Todos estos datos anteriores deben de estar en un archivo llamado **“HistorialJuegos.txt”,** su contenido será después que finalice cada juego.

3.4. Estadísticas de Juegos

Se debe mostrar el siguiente detalle:

- Total de juegos
- Total de juegos ganados
- Total de juegos perdidos
- Suma de premios entregados
- Monto máximo obtenido como premio
- Monto mínimo obtenido como premio diferente a CERO

4. Opciones de Jugador

Para acceder a estas funcionalidades el usuario deberá ingresar por medio del **Menú Principal** y se deben habilitar las siguientes funcionalidades (Menú General):

- Jugar
- Retornar

4.1. Jugar

El sistema solicita los siguientes datos al jugador:

- Cédula, validar su formato de 9 dígitos y su formato propio
- Nombre completo
- Sexo: **H**ombre o **M**ujer

Una vez obtenido sus datos personales, el programa mostrará un mensaje al jugador si ya desea iniciar con el juego, al **aceptar**, **se guarda la fecha y hora de inicio.**

El juego debe seleccionar en forma aleatoria el archivo de juego a utilizar por ejemplo "**ListaPreguntas12.txt**"**,** y este debe guardarse en este archivo.

Debe presentar pregunta por preguntar al usuario, con sus cuatro opciones, si el usuario selecciona la opción correcta, va ganado el premio monetario asociado a esa pregunta.

El usuario en todo momento tiene la posibilidad de salir del juego

4.1.1. Opcional

Al igual que el juego real existirá el uso de los **comodines**, tales como:

- Cambio de pregunta, para esto solo debe seleccionar una nueva pregunta del archivo **Preguntas.txt** que no se esté utilizando en el juego actual
- 50% (Para esto usted debe mostrar solo 2 opciones, donde una de ellas será la correcta)

Deberá llevar un control de los comodines restantes

4.2. Salir

Finaliza el programa. Al seleccionar esta opción se debe generar un mecanismo (manejo de archivos) para que los datos sean persistentes. La información se debe mantener para los siguientes usos del programa.

5. Opcionales

Se darán **10 puntos adicionales** si la **interfaz** del juego en su forma textual se asemeja a la versión que se visualiza en la siguiente imagen.

Se darán **5 puntos adicionales** si se incluye el **manejo de los comodines** tal como se menciona en la sección 4.1.1**.**

6. Aspectos técnicos

El proyecto deberá estar escrito en el lenguaje de programación **Python**. En caso de requerir librerías adicionales para compilar y ejecutar el programa, deberán **especificarlo** en la documentación (***ejemplo, manejo de fechas, random, para borrar archivos***), ya que de lo contrario se descontarán

puntos en la evaluación. Al iniciar el programa se carga la información de disco y al salir se guarda (o una la manera permita que todos los datos registrados sean persistentes).
Y considerar lo siguiente:

- Se debe utilizar **iteración** en el desarrollo del proyecto.
- Se deben manejar **mensajes claros** al usuario.
- Realizar **validaciones de captura de campos**. Deben hacer uso de mecanismos de validación eficientes
- Número de **identificador** es único y autogenerado
- Toda **función bult-in** de Python no están permitidas, a menos que sea necesario según la forma de cómo esté programando el proyecto, pero esto debe ser consultado al profesor.
- El proyecto debe ser guardado en el repositorio del **GitHub** (https://classroom.github.com/a/_TysqVWK).

7. Documentación

La documentación es un aspecto de gran importancia en el desarrollo de programas, especialmente en tareas relacionadas con el mantenimiento de estos.

Para la **documentación interna**, deberán incluir comentarios descriptivos para cada función, con sus **entradas, salidas, restricciones**.

La **documentación externa** deber ser en **PDF** (si lo suben en otro formato, no será leído) y deberá incluir:

1. Portada.
2. Manual de usuario: **instrucciones de compilación, ejecución y uso**.
3. Un enlace a un video en **youtube** donde muestre la funcionalidad de su aplicación. Su duración no de ser mayor a los 15 minutos.
4. Descripción del problema.
5. Diseño del programa: decisiones de desarrollo, algoritmos usados.
6. Librerías usadas: creación de archivos, etc.
7. Análisis de resultados: lista detallada de objetivos alcanzados, objetivos no alcanzados, y razones por las cuales no se alcanzaron los objetivos (en caso de haberlos).
8. **Bitácora** en GitHub con los **commit** por usuario incluyendo su descripción al momento de hacerlo.
9. Conclusión (es), sería su punto de vista sobre el programa desarrollado

8. Evaluación

La evaluación se va a centrar en dos elementos: programación y documentación. El proyecto programado tiene un valor de **15%** de la nota final, en el rubro de Proyectos.

Desglose de la evaluación del proyecto programado:

1. Documentación interna 5 ptos.
2. Documentación externa 5 ptos.
3. Funcionalidad 75 ptos (ver detalle en Software a Desarrollar)
4. Revisión del proyecto 10 ptos.
5. Hora de Entrega 5 ptos.

9. Forma de trabajo

El trabajo se debe realizar de forma **individual**.

10. Aspectos administrativos

Crear 2 carpetas llamadas **documentación** y **programa**, en la primera deberá incluir el documento **PDF** solicitado y en la segunda los archivos y/o carpetas necesarias para la implementación de este proyecto programado.
Deben crear un archivo llamado **README.md**, este archivo debe contener la siguiente información:

a. Nombre del curso
b. Número de semestre y año lectivo
c. Nombre del Estudiante
d. Número de carné del estudiante
e. Estatus de la entrega (debe ser **CONGRUENTE** con la solución entregada): [Deplorable|Regular|Buena|MuyBuena|Excelente|Superior]

Con este enlace https://docs.github.com/es/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax le ayudará a dar formato al archivo README.md

11. Rúbrica de evaluación

FUNCIONALIDAD	**75%**	**0**	
Rubro	**Valor**	**Obtenido**	**Observaciones**
Menú	**5**		
Opciones Administrativas			
Control de acceso	**5**		
Gestión de preguntas	**15**		
Gestión de Juegos	**20**		
Histórico de juegos	**10**		
Estadísticas del juego	**15**		
Retornar	**2,5**		
Opciones de Jugador			

FUNCIONALIDAD	**75%**	**0**	
Rubro	**Valor**	**Obtenido**	**Observaciones**
Jugar	**25**		
Retornar	**2,5**		
Extras			
Interfaz gráfica	**10**		
Manejo de los comodines	**5**		
TOTAL	**115**	**0**	
DOCUMENTACIÓN INTERNA	**5%**		
DOCUMENTACIÓN EXTERNA	**5%**		
REVISIÓN DEL PROYECTO	**10%**		
HORA DE ENTREGA	**5%**		
NOTA FINAL	**100%**	**0%**	

12. Entrega

En el rubro de "Hora de Entrega" valdrá 5 puntos de la nota total del proyecto, según la siguiente escala:

a. Si se entrega antes de las 11:55:55 PM del domingo **25 de setiembre de 2022**, 5 puntos.
b. Si se entrega antes de las 11:55:55 AM del lunes **26 de setiembre de 2022**, 2.5 puntos.
c. Si se entrega antes de las 11:00:00 PM del lunes **26 de setiembre de 2022**, 0 puntos.
   Después de este punto, **NO SE ACEPTARÁN** más trabajos.

Todo el contenido de cada proyecto debe ser 100% original y en caso de plagio se asignará nota cero.