## **Primera Entrega Triviathon**

**Programación Orientada a Objetos**

## **Autores:**

## **Samuel Gallego**

## **Valentina Ramos**

## **Sara Moncada**

**Profesor:**

**Jesús Andrés Hincapie Londoño**

## **Medellín, Colombia**

**Universidad de Medellín**

## **24 de mayo de 2024**

## **Triviathon: Un juego de preguntas desafiante y atractivo.**

En el juego "Triviathon", los jugadores deberán elegir entre 1 y 3 categorías de las 8 disponibles: Historia, Arte, Cultura Pop, Ciencia y Tecnología, Geografía, Deportes, Entretenimiento y Ciencia Ficción. Cada categoría cuenta con 5 preguntas, y por cada respuesta correcta, el jugador suma un punto. Sin embargo, si se equivoca en alguna pregunta, la ronda llega a su fin.

Para comenzar, los usuarios deben registrarse con el nombre de su elección y luego seleccionar las categorías en las que desean poner a prueba su conocimiento.

Durante la ronda, los jugadores disponen de 3 opciones de ayuda: 1. "50/50", que reduce las opciones de respuesta de 4 a 2; 2. "Cambio de pregunta", que reemplaza la pregunta actual por una nueva; y 3. "Texto de IA", donde se muestra un texto relacionado con la pregunta que contiene la respuesta implícita y el jugador debe interpretarla.

Si el jugador elige una sola categoría, solo podrá utilizar una ayuda en toda la ronda. Si opta por 2 categorías, tendrá acceso a 2 ayudas, y si selecciona 3 categorías, podrá utilizar las 3 ayudas disponibles. Las ayudas se utilizan en el orden elegido por el jugador y una vez agotadas, quedan bloqueadas.

En caso de escoger 2 o 3 categorías, el jugador tendrá la opción de continuar jugando hasta responder todas las preguntas o retirarse cada vez que haya respondido 6 preguntas. Si decide retirarse, la ronda finaliza; de lo contrario, la ronda continúa.

El jugador tiene la opción de jugar hasta 3 rondas si así lo desea. Una vez que una ronda finaliza, se le da al jugador la opción de continuar a la siguiente ronda o finalizar el juego. Si decide continuar a la siguiente ronda, deberá escoger nuevamente entre 1 y 3 categorías, las cuales no pueden repetirse con las categorías de la ronda anterior. Al finalizar el juego, se muestra al jugador las rondas que jugó, los puntos obtenidos y las categorías elegidas en cada ronda.

**Requisitos funcionales**

1. Registrar al usuario
2. Seleccionar Categoría
3. Iniciar el juego
4. Generar preguntas aleatorias
5. Generar ayudas
6. Evaluar respuesta del jugador
7. Registrar puntos
8. Finalizar la ronda
9. Visualizar la puntuación

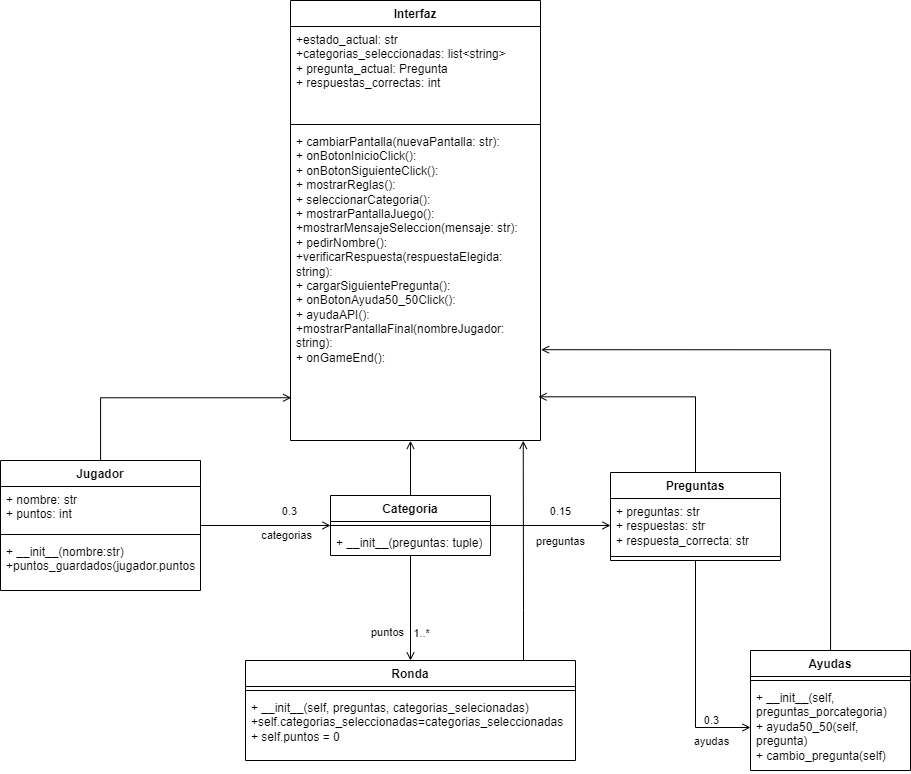
**Requisitos funcionales de consola**

1. Generar preguntas aleatorias
2. Generar ayudas
3. Evaluar respuesta del jugador
4. Registrar puntos

**Requisitos funcionales de Interfaz**

1. Registrar al usuario
2. Seleccionar categoría
3. Iniciar el juego
4. Finalizar la ronda
5. Visualizar la puntuación

**Modelo del mundo consola e Interfaz**

****

| **Nombre** | R1 - Registrar jugador |
| --- | --- |
| **Resumen** | El sistema debe permitir registrar un jugador para poder iniciar un juego |
| **Entradas** | -Nombre del jugador |
| **Resultado** | 1. El sistema muestra un mensaje dando la bienvenida al jugador  2. El sistema carga las categorías al jugador  3. El sistema le muestra el menú con la opción para iniciar un nuevo juego |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Registrar jugador | registrar\_jugador(nombre) | Jugador |
| Crear jugador | \_\_init\_\_(nombre,categoría) | Jugador |

| **Nombre** | R2 - Seleccionar Categoría |
| --- | --- |
| **Resumen** | El sistema permite al jugador seleccionar qué tipo de categorías desea elegir. |
| **Entradas** | - Categorías Seleccionadas |
| **Resultado** | 1. El sistema imprime las categorías. 2. El usuario debe elegir entre dos o tres categorías,con las que el jugador puede iniciar el juego   2.1.Si el usuario selecciona la categoria mixta, ya no podrá elegir otra categoría.   1. El sistema recibe la selección de categorías del jugador 2. El sistema elimina las preguntas de las categorías no seleccionadas. 3. El sistema selecciona aleatoriamente preguntas de la categoría elegida. |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Mostrar categorías | solicitar\_categorias() | categorías |
| Escoger categorías | categorias\_seleccionadas = [] | categorías |
| Generar preguntas categoría | generar preguntas() | preguntas |

| **Nombre** | R3 - Iniciar Juego |
| --- | --- |
| **Resumen** | El sistema inicia el juego |
| **Entradas** | -respuestas |
| **Jugador.** | 1.se muestra la pregunta  2.se muestra las opciones de respuesta  3.1Se ejecuta el requisito R4  3.1.1 se elimina la opción de ayuda seleccionada del requisito R4  3.2se responde la pregunta  4.el sistema evalúa la respuesta del jugador  4.1 si la respuesta es correcta se ejecuta R5  4.2si es incorrecta se ejecuta otra vez R5 |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| opciones de respuesta | \_\_init\_\_(self, pregunta, respuestas, respuesta\_correcta, categoria) | preguntas |
| Proceso de respuesta | jugar\_pregunta(self, pregunta, ayuda) | Juego |
| ayudas | ayuda\_50\_50(self, pregunta)  def cambio\_pregunta(self) | ayudas |

| **Nombre** | R4 - Generar Preguntas |
| --- | --- |
| **Resumen** | El sistema debe generar las preguntas |
| **Entradas** | -respuestas  -ayudas |
| **Jugador** | 1.Se hace un banco de preguntas y respuestas con base a las categorías seleccionadas  2.se revuelven y se mezclan  3. se inicia el R.3 |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Crear la lista de preguntas | \_\_init\_\_(self, jugadores):self.todas\_las\_preguntas = self.crear\_lista\_preguntas() | Juego |
| Generar preguntas | crear\_lista\_preguntas(self): | Juego |

| **Nombre** | R5 - Generar Ayudas |
| --- | --- |
| **Resumen** | El sistema genera ayudas cuando el jugador lo solicite |
| **Entradas** | -respuestas  -ayudas |
| **Jugador** | 1.El jugador selecciona “ayuda”  1.1 Si selecciona la ayuda de 50/50  1.1.1 El sistema elimina dos respuestas incorrectas  1.2 Si opta por cambiar la pregunta  1.2.1. Se ejecutan los requisitos R3,R4  1.3 Si solicita ayuda con I.A  1.3.1 Se llama a una I.A la cual asiste al usuario en la selección de la respuesta más adecuada  2. El jugador responde la pregunta |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Ayuda 50/50 | ayuda\_50\_50(self, pregunta): | Ayudas |
| Cambio Pregunta | cambio\_pregunta(self) | Ayudas |

| **Nombre** | R6 -Evaluar respuesta del jugador |
| --- | --- |
| **Resumen** | El sistema evalúa si la respuesta del jugador es correcta o incorrecta. |
| **Entradas** | Ninguna |
| **Jugador** | 1.Si responde correcto  1.1. Se ejecuta R.7  1.2. El ejecuta R.8  1.3. Se ejecuta R.3,R.4  1.4. El Jugador continúa jugando  2. Si responde incorrecto  2.1. Se ejecuta R.7  2.2. El ejecuta R.8  2.3. Se ejecuta R.9 |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Responder Preguntas | jugar\_pregunta(self, pregunta, ayuda) | Juego |

| **Nombre** | R7 -Registrar puntos |
| --- | --- |
| **Resumen** | Se usa un contador para sumar el puntaje del jugador |
| **Entradas** | Ninguna |
| **Jugador** | 1.si el jugador responde correctamente se sumará su punto  1.2. Si el jugador responde incorrectamente antes de llegar al seguro R.8  1.2.1. Se guardará el resultado del último punto de seguro  1.2.2. Se mostrará el puntaje en R.9  1.3. Si responde correctamente y alcanza el punto de seguro  1.3.2. Se guardará la respuesta del último seguro para el requisito R.9 |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Suma de puntos | incrementar\_puntos(self, puntos) | Rondas |
| contador de rondas | jugar(self) | Jugar |

| **Nombre** | R8 -Finalizar la ronda |
| --- | --- |
| **Resumen** | El sistema debe permitir que el jugador finalice la ronda |
| **Entradas** | Ninguna |
| **Jugador** | 1.Si el usuario responde mal o selecciona la opción “retirar”, entonces:  1.1. La ronda se finaliza y se muestran los puntos obtenidos  1.2. Aparecen las opciones de “Continuar a la siguiente ronda” o “Finalizar el juego”  1.2.1 Si selecciona “Continuar a la siguiente ronda”  1.2.1.1 se guardan los puntos de la ronda anterior y se inicializa una ronda nueva (no se acumulan los puntos)  1.2.2. Si selecciona “Finalizar el juego”  1.2.2.1. Se ejecuta R9 |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| finalizó ronda | jugar(self): | Juego |
| casilla pausa | jugar\_ronda(self, ronda) | Juego |
| término juego | jugar(self): | Juego |

| **Nombre** | R9 - Visualizar la puntuación |
| --- | --- |
| **Resumen** | El sistema debe permitir visualizar la puntuación mayor del jugador |
| **Entradas** | Evaluar respuesta  Nombre usuario |
| **Jugador** | 1. El sistema imprime el nombre del jugador con:  1.1. El mayor puntaje que obtuvo de todas las rondas que realizó  1.2. Mensaje de finalización |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Muestra el puntaje total | mostrar\_resultados\_finales  (resultados\_rondas) | Rondas |

**Link del Repositorio de Github**

https://github.com/luna1860/Proyecto\_POO.git