Proyecto - RISK

Juan Diego Echeverry Plazas

Nicolas Rincón Ballesteros

Santiago Yañez Barajas

PRESENTADO A:

Alejandro Castro Martinez

Decanatura Facultad de ingeniería

Carrera de Ingeniería de Sistemas

ESTRUCTURAS DE DATOS

PONTIFICIA UNIVERSIDAD JAVERIANA

BOGOTÁ D.C

2023

Tabla de Contenido

Descripción de entradas, salidas y condiciones

Procedimiento Principal

Antes que nada, es importante entender la forma en la que guardamos los datos en la aplicación, para esto, se decidió crear el siguiente diagrama:

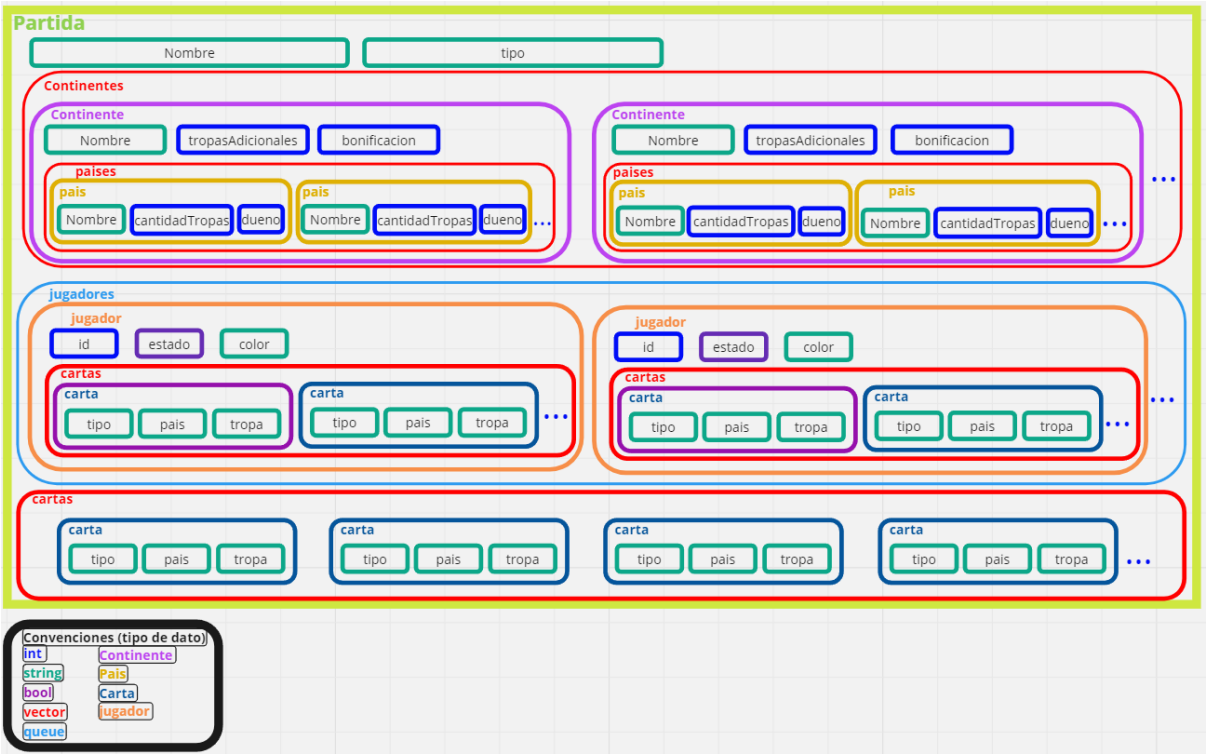

Figura 1. Estructura de datos. Fuente: Elaboración propia.

Como se puede ver, se tiene una estructura llamada Partida, de tipo Partida la cual tiene como atributos, nombre, tipo, setsTradeados, un vector de tipo Continente, una queue de tipo Jugador, un vector de tipo Carta y un vector de tipo Dado. Cabe recalcar que cada partida es un juego de RISK diferente. La siguiente es la explicación de cada uno de los atributos de Partida.

- Nombre: Este es el nombre con el cual se guardará la partida, este es ingresado por el usuario al momento de iniciar la partida.
- Tipo: Este es el tipo de la partida, este es ingresado por el usuario al momento de iniciar la partida. Tiene dos posibles tipos: Normal o MisionSecreta.
- SetsTradeados: Al momento de crear la partida, este valor inicia en 0 por defecto. Este representa el número de sets tradeados en la partida para así saber cuántas cartas serán asignadas cuando un jugador complete ciertas cartas.
- Vector de tipo Continente: Al momento de crear la partida, se cargaran todos los continentes con su respectivo nombre, vector de tipo Pais, con su respectivo nombre y cantidad de tropas que se encuentran en ese pais en especifico y finalmente, por cada Continente, un número entero tropasAdicionales que representa el número de tropas otorgadas si un jugador es dueño de todo el continente.

- Queue de tipo Jugador: Al momento de crear la partida, se le pregunta al usuario la cantidad de jugadores, a cada uno de estos se le asigna un id representando su turno, booleano de estado indicando si sigue en la partida o ya fue eliminado, un color que se le pregunta al usuario, un vector de strings con el nombre de paises que actualmente domina y un vector de cartas de tipo Carta con toda su informacion necesaria.
- Vector de tipo Carta: Al momento de crear la partida, se crean todas las cartas con su respectivo tipo, país y tropa para posteriormente ser asignado a los jugadores.

Tras entender la estructura de cada Partida, se muestra el funcionamiento de los siguientes comandos de manera gráfica:

Comando inicializar:
Para entender mejor cómo es el funcionamiento del comando inicializar, se decidió realizar el siguiente diagrama de flujo:

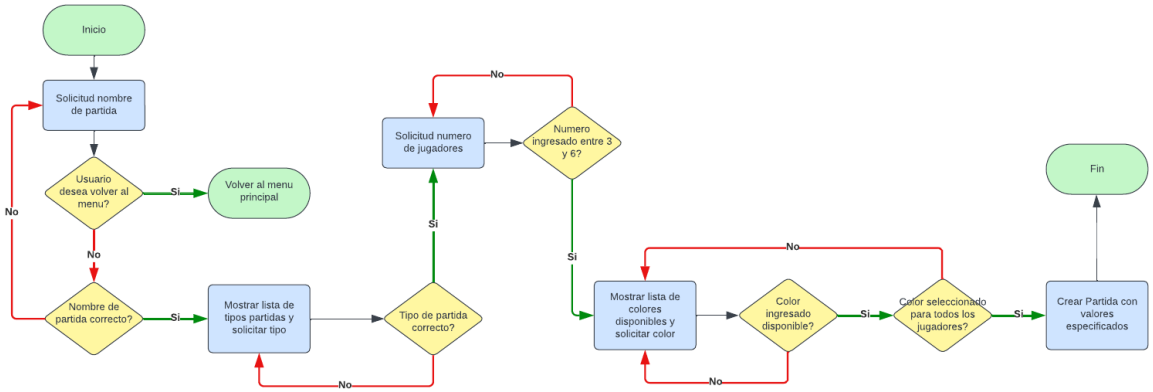

Figura 2. Diagrama de flujo inicializar. Fuente: Elaboración propia.

Cabe recalcar que el usuario puede escribir salir en cualquier momento del flujo de interacción para volver al menú principal si cometió algún error, descartando los cambios realizados.

Tras entrar al comando inicializar, el sistema hace una solicitud para que el usuario ingrese el nombre de la partida a crear, pueden surgir las siguientes cosas:

- Nombre de Partida Vacío: El sistema rechaza el nombre porque envió un espacio y se imprime el siguiente mensaje de error: *"El nombre de la partida no puede contener solo espacios en blanco"*. Se vuelve a pedir el nombre de la partida.
- Nombre de Partida con Espacios: El sistema rechaza el nombre porque envía dos argumentos y se imprime el siguiente mensaje de error: *"El nombre de la partida no puede estar separada por espacios"*. Se vuelve a pedir el nombre de la partida.
- Usuario presiona 'Enter': El sistema vuelve a pedir el nombre de la partida.

Una vez aceptado el nombre para crear la partida, se muestra una lista de los tipos de partida para que el usuario seleccione. El sistema hace la solicitud para que el usuario ingrese el nombre del tipo de la partida, pueden surgir las siguientes cosas:

- Tipo con espacios: El sistema rechaza lo ingresado por el usuario y se imprime el siguiente mensaje de error: *"Ingrese solamente el tipo de la partida"*. El sistema vuelve a mostrar la lista de los tipos de partida y vuelve a pedir el tipo de la partida.
- Tipo inexistente: El sistema rechaza lo ingresado por el usuario y se imprime el siguiente mensaje de error: *"Tipo partida: ' ' no encontrado. Por favor ingrese un tipo de partida valido"*. El sistema vuelve a mostrar la lista de los tipos de partida y vuelve a pedir el tipo de la partida.
- Usuario presiona 'Enter': El sistema vuelve a mostrar la lista de los tipos de partida y vuelve a pedir el tipo de la partida.

Una vez aceptada el tipo de partida, el sistema hace la solicitud para que el usuario ingrese el número de jugadores para la partida, pueden surgir las siguientes cosas:

- Número con espacios: El sistema rechaza lo ingresado por el usuario porque ingresó dos argumentos e imprime el siguiente mensaje de error: *"Ingrese solamente la cantidad de jugadores"*. El sistema vuelve a pedir el número de jugadores.
- Número incorrecto: El sistema rechaza lo ingresado por el usuario porque el número de jugadores debe estar entre 3 y 6. Se imprime el siguiente mensaje de error: *"Error. Debe ingresar entre 3 a 6 jugadores"*. El sistema vuelve a pedir el número de jugadores.
- No ingresa número: El sistema rechaza lo ingresado por el usuario porque no ingresa un dígito y se imprime el siguiente mensaje de error: *"Error. Debe ingresar el número de jugadores para la partida"*. El sistema vuelve a pedir el número de jugadores.
- Usuario presiona 'Enter': El sistema vuelve a pedir el número de jugadores.

Una vez aceptados el número de jugadores, el sistema muestra una lista de los colores disponibles hasta que se seleccionen para todos los jugadores creados previamente, pueden surgir las siguientes cosas:

- Ingresa color inexistente: El sistema rechaza lo ingresado por el usuario y se imprime el siguiente mensaje *"Color ' ' no encontrado o ya fue seleccionado. Por favor ingrese un color válido"*. El sistema vuelve a mostrar la lista de colores posibles para que el usuario seleccione.
- Ingresa color previamente seleccionado: El sistema rechaza lo ingresado por el usuario y se imprime el siguiente mensaje *"Color ' ' no encontrado o ya fue seleccionado. Por favor ingrese un color válido"*. El sistema vuelve a mostrar la lista de colores posibles para que el usuario seleccione.
- Usuario presiona 'Enter': El sistema vuelve a mostrar la lista de colores posibles para que el usuario seleccione.

Una vez que se ha solicitado la información necesaria, que comprende el nombre de la partida, el tipo de partida, la cantidad de jugadores y el color asignado a cada uno, el sistema procederá a asignar los países a cada jugador. Este proceso se llevará a cabo mediante un funcionamiento automatizado que implica la distribución aleatoria de los países y tropas de manera equitativa por parte del sistema.

Para lograr la distribución aleatoria de los países el sistema primero genera un número aleatorio entre la cantidad de jugadores que hay en la partida, por ejemplo, si hay 4 jugadores, es un número entre el 1 y el 4. Posteriormente, la lógica que se utiliza es tener un vector auxiliar de enteros llamado “vectorJugadoresIguales” el cual tendrá el número de posiciones como tantos jugadores haya en la partida. En este, se va a ir guardando el número aleatorio generado, sin embargo, este solo puede aparecer una vez porque se va a ir asignando de a 4 países a la vez, de forma que el primer número (representado el id de cada jugador) que se genere es quien va a recibir el primer país, el segundo número que salga será quien recibirá el segundo país y de esta forma hasta completar el número de jugadores en la partida.

Para representar esto mejor, se crea el siguiente diagrama el cual es el vector “vectorJugadoresIguales”:

Figura 3. Vector vectorJugadoresIguales. Fuente: Elaboración propia.

En este caso, ya salieron los siguientes números aleatorios: 3,1 y 4. Cabe mencionar que cada que se encuentre un número y se agregue a este vector, se creará el respectivo país con nombre y continentes ya guardados en otro vector.

Si vuelve a salir el 3, este se rechaza porque este vector se utiliza para asegurar que todos los jugadores tengan igual cantidad de países (en los casos que sea posible). Una vez se genere el valor aleatorio 2, este se agrega al vector.

Figura 4. Vector tras encontrar el último valor restante. Fuente: Elaboración propia.

Una vez se encuentre que el tamaño del vector sea igual al tamaño de jugadores, a este vector se le hará .clear() para volver a iniciar el proceso de selección.

Comando turno:

Este comando recibe un id de turno para el cual se va a efectuar el turno. Para entender el funcionamiento de cómo se tratan los turnos en la aplicación, se muestra el siguiente gráfico que representa un ejemplo de una cola con 5 jugadores:

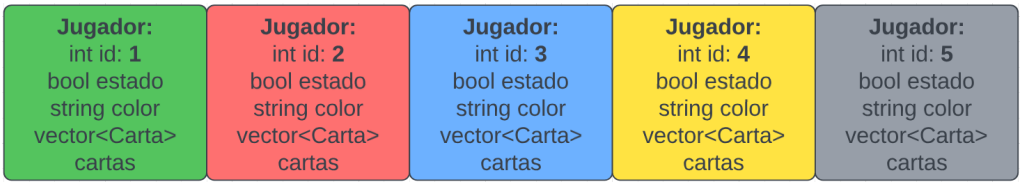

Figura 5. Cola de jugadores

Primero, se crea una variable tipo Jugador auxiliar para guardar el frente de la cola y utilizar el método ObtenerId() para compararlo con el que llegó como parámetro, si este no coincide, quiere decir que no es turno de ese jugador y se rechaza la petición.

Si el turno de jugador es correcto, se ejecutan las tres fases correspondientes del turno, se hace .pop() de la cola de Jugadores y se hace .push() para agregarlo al final de la cola. La siguiente es la representación visual de esto:

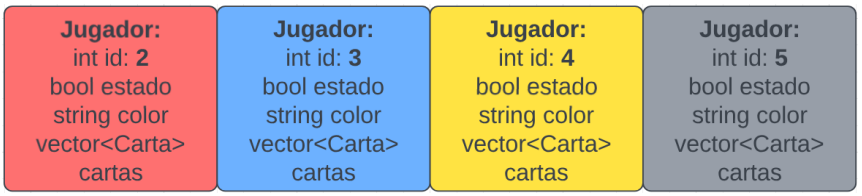

Figura 6. pop() de la cola de Jugadores. Fuente: Elaboración propia.

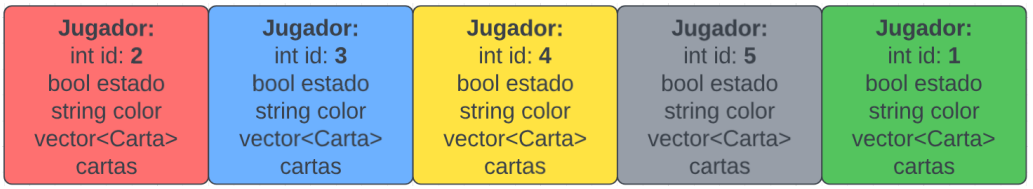

Figura 7. push() de la cola de Jugadores. Fuente: Elaboración propia.

Cada que un jugador sea eliminado de la partida, este se elimina de la cola de jugadores y el único jugador que quede en la cola será el ganador de la partida.

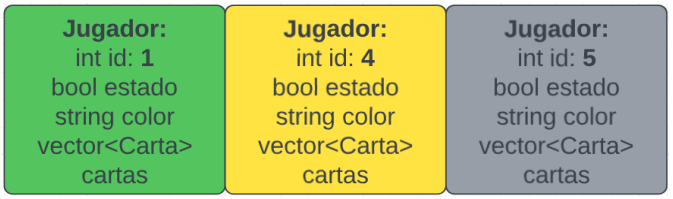

Figura 8. Jugadores 2 y 3 eliminados. Fuente: Elaboración propia.

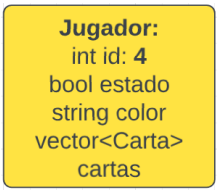

Figura 9. Jugador 4 ganador. Fuente: Elaboración propia.

Tras entender cómo funciona el sistema de turnos mediante colas en la partida, para entender mejor el funcionamiento de cada unas de las fases de turno, se decidió realizar el siguiente diagrama de flujo:

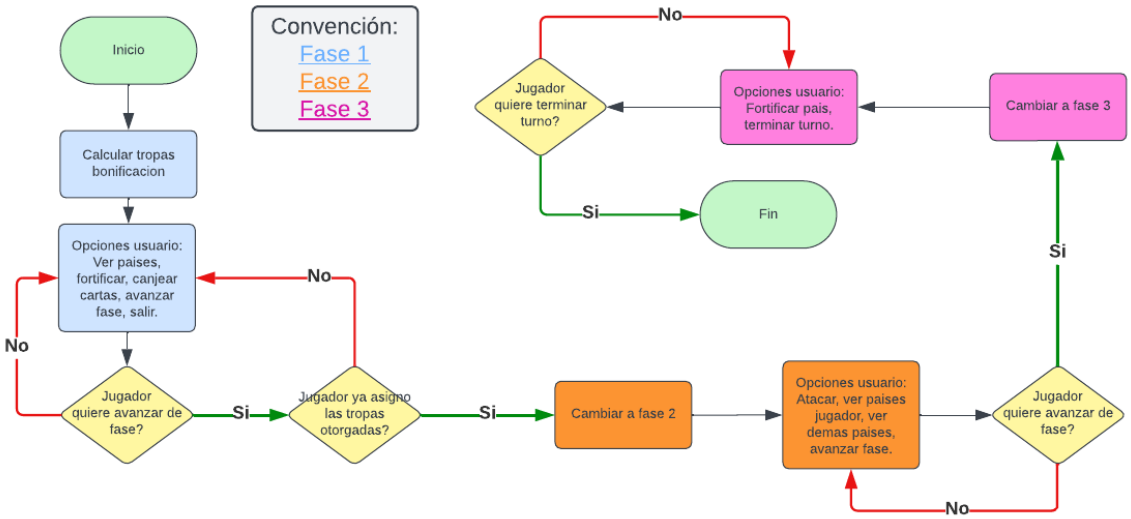

Figura 10. Flujo sistema de turnos. Fuente: Elaboración propia.

- **Fase 1 (Obtener nueva unidades):** Durante esta fase se le informará al usuario cuantas tropas adicionales puede reclamar, para luego preguntar en cuál(es) de los territorios de su propiedad las quiere asignar y en qué cantidad. Ahora, el sistema tendrá la tarea de calcular la bonificación de las tropas, que este variará dependiendo de la cantidad de territorios que tenga el usuario. Posteriormente se le mostrará un menú al usuario donde podrá ver todos los países, ver sus países, fortificar el país con las tropas de bonificación, canjear cartas, dirigirse a la siguiente fase o salir, es importante resaltar que el usuario no podrá avanzar a la siguiente fase si no ha distribuido las tropas de bonificación anteriormente.

- **Fase 2 (Atacar):** Durante el transcurso de esta etapa, el usuario dispondrá de la oportunidad de llevar a cabo la configuración de su estrategia ofensiva. En este proceso, podrá seleccionar tanto el punto de origen del ataque como el destino al cual desea dirigirse, teniendo en cuenta que el usuario solo podrá seleccionar este territorio si tiene más de una tropa en este, ya que no está permitido atacar desde un territorio en el que solo exista una tropa. Además, el sistema proporciona un menú de opciones para cada jugador en esta fase, en el cual podrá ejecutar opciones como llevar a cabo

ataques, ver sus propios territorios, mostrar los territorios de los otros jugadores y progresar a la siguiente fase del juego. Cabe recalcar que es posible no atacar y simplemente ir a la siguiente fase.

- **Fase 3 (Fortificar):** En la fase final del turno de comando, el usuario tendrá la oportunidad de fortalecer uno de sus territorios, mediante la selección de la cantidad de tropas que desee movilizar desde un territorio propio hacia el territorio elegido previamente, sin embargo, sólo podrá movilizar tropas si el territorio en cuestión tiene más de una tropa, ya que, no es posible dejar un territorio sin tropas.

Operaciones Auxiliares (Comandos)

- inicializar:
  - Comando: inicializar <nombre_archivo>
  - Descripción: Se inicializa una nueva partida si no se especifica el nombre del archivo de la partida guardada. De especificarse uno se reanuda la partida.
  - Condiciones: El usuario no ingrese un espacio o un vacío.
- turno:
  - Comando: turno <id_jugador>
  - Descripción: Informa al jugador la cantidad de unidades que puede reclamar, para luego asignarlas y en qué cantidad. Después, se encarga de la configuración del ataque desde que territorio y hacia cual, luego informa los valores obtenidos con los dados y la cantidad de unidades que se ganan o se pierden, este proceso se repite hasta que alguno de los territorios se quede sin unidades o hasta que el atacante decida detenerse. Por último, pregunta al jugador los territorios vecinos que desea seleccionar para la fortificación y también la cantidad de unidades que se trasladan de uno al otro.
  - Condiciones: El usuario debe indicar el id del jugador del cual quiere usar el turno.
- guardar:
  - Comando: guardar nombre_archivo
  - Descripción: El estado actual del juego es guardado en un archivo de texto plano, se guarda la cantidad de jugadores, nombre de cada jugador, color de cada jugador, países que ocupa, etc.
  - Condiciones: Debe existir una partida previamente creada.
- guardar_comprimido:
  - Comando: guardar nombre_archivo
  - Descripción: el estado del juego actual es guardado en un archivo binario. Se guarda la cantidad de jugadores, nombre de cada jugador, color de cada jugador, países que ocupa, etc.
  - Condiciones: Debe existir una partida previamente creada.
- costo_conquista:
  - Comando: costo_conquista <territorio>

    - Descripción: Se calcula el costo y secuencia de territorios a ser conquistados para lograr controlarlo. El territorio donde debe atacar debe ser aquel que el jugador tenga controlado más cerca al dado por el usuario.
    - Condiciones: Debe existir una partida previamente creada y se debe ingresar un territorio válido.
- conquista_mas_barata:
    - Comando: conquista_mas_barata
    - Descripción: Calcula la conquista más barata para el jugador actual de todos los territorios posibles. Es decir, aquel territorio que pueda implicar un menor número de unidades perdidas.
    - Condiciones: Debe existir una partida previamente creada.
- clear:
    - Comando: clear
    - Descripción: Limpia la consola.
    - Condiciones: Ninguna.
- salir:
    - Comando: salir
    - Descripción: Termina la ejecución del programa. Toda partida que no se haya guardado se perderá.
    - Condiciones: Ninguna.

Diseño de TAD'S

TAD JUGADOR

Datos mínimos:

- id, entero, representa el identificador único de cada jugador (puede ser o un número o un nombre.
- estado, booleano, representa si el jugador ha sido eliminado o no.
- color, cadena de caracteres, representa el color del jugador.
- cartas, lista de Cartas del jugador, representa las cartas que posee el jugador.

Operaciones:

- ObtenerId( ): Retorna el id del jugador
- ObtenerEstado( ): Retorn a el estado del jugador
- ObtenerColor( ): Retorna el color elegido por el jugador
- ObtenerCartas( ): Retorna la baraja de cartas del jugador
- FijarId(nid): Fija el id del jugador a nid.
- FijarEstado(nbool): Fija el estado de juego del jugador a nbool (eliminado, activo).
- FijarColor(ncolor): Fija el color del jugador a ncolor.
- FijarCartas(ncartas): Fija la baraja de cartas que posee el jugador a ncartas.

TAD CARTA

Datos mínimos:

- tipo, cadena de caracteres, representa el tipo de carta (normal, comodín o misión secreta).
- pais, cadena de caracteres, representa el país de la tarjeta.
- tropa, cadena de caracteres, representa la tropa de la tarjeta.

Operaciones:

- ObtenerTipo(): Retorna el tipo de carta
- ObtenerPais(): Retorna el nombre del país de la tarjeta
- ObtenerTropa(): Representa la tropa de la tarjeta
- FijarTipo(ncarta): Cambia el tipo de carta a ncarta
- FijarPais(ntarjeta): Cambia el país de la tarjeta a ntarjeta
- FijarTropa(ntropa): Cambia la tropa a ntropa

TAD PAÍS

Datos mínimos:

- nombre, cadena de caracteres, representa el nombre del país.
- cantidadTropas, entero, representa la cantidad de tropas ubicadas en ese país.
- dueno, entero, representa el dueño del país en cuestión.

Operaciones:

- ObtenerNombre( ): Obtiene el nombre del país
- ObtenerCantidadTropas( ): Obtiene la cantidad de tropas que defienden el país.
- ObtenerDueno(): Obtiene el id del dueño del país.
- FijarNombre(nnombre): Fija el nombre del país a nnombre.
- FijarCantidadTropas(ncantidadtropas): Fija la cantidad de tropas que defienden el país a ncantidadtropas
- FijarDueno(ndueno): Fija el id del dueño de cada país.

TAD CONTINENTE

Datos mínimos:

- nombre, cadena de caracteres, representa el nombre del continente.
- tropasAdicionales, entero, representa la bonificación de tropas adicionales del continente cuando un jugador domina todos los países de este.
- paises, lista de Pais, representa los países que se encuentran en ese continente.w
- bonificación, entero, representa cuando el jugador completa alguno de los continentes recibiría tropas extra

Operaciones:

- ObtenerNombre( ): Obtiene el nombre del continente.
- ObtenerTropasAdicionales( ): Obtiene el número de tropas adicionales.
- ObtenerPaises( ): Obtiene la lista de países que componen al continente.
- ObtenerBonificacion(): Obtiene la bonificación por completar continente
- FijarNombre(nnombre): Fija el nombre del continente a nnombre.
- FijarTropasadicionales(nTropasAdicionales): Fija el número de tropas adicionales.
- FijarPaises(npaises): Fija la lista de países que componen al continente a npaises
- FijarBonificacion(nbonificacion): Fija la bonificación por completar continente a nbonificacon.

<u>TAD PARTIDA</u>

Datos mínimos:

- nombre, cadena de caracteres, representa el nombre del juego actual.
- tipo, cadena de caracteres, representa la variante del juego actual.
- setsTradeados, entero, representa el número de sets tradeados en la partida.
- continentes, lista de Continente, representa los continentes que se encuentran en la partida.
- jugadores, cola de Jugador, representa los jugadores que se encuentran en la partida en su respectivo orden.
- cartas, lista de Carta, representa las cartas que se encuentran en la partida.

Operaciones:

- ObtenerNombre( ): Obtiene el nombre de la partida
- ObtenerTipoPartida( ): Obtiene el modo de juego de la partida
- ObtenerSetsTradeados( ): Obtiene la cantidad de tradeos de la partida
- ObtenerContinentes( ): Obtiene los continentes de la partida
- ObtenerJugadores( ): Obtiene los jugadores de la partida
- ObtenerCartas( ): Obtiene las cartas de la partida
- FijarNombre(nnombre): Fija el nombre de la partida a nnombre
- FijarTipoPartida(ntipo): Fija el modo de juego de la partida a ntipo
- FijarSetsTradeados(ntradeos): Fija la cantidad de tradeos de la partida a ntradeos
- FijarContinentes(ncontinentes): Fija los continentes de la partida a ncontinentes
- FijarJugadores(njugadores): Fija los jugadores de la partida njugadores
- FijarCartas(ncartas): Fija las cartas de la partida a ncartas

Diagrama TAD'S

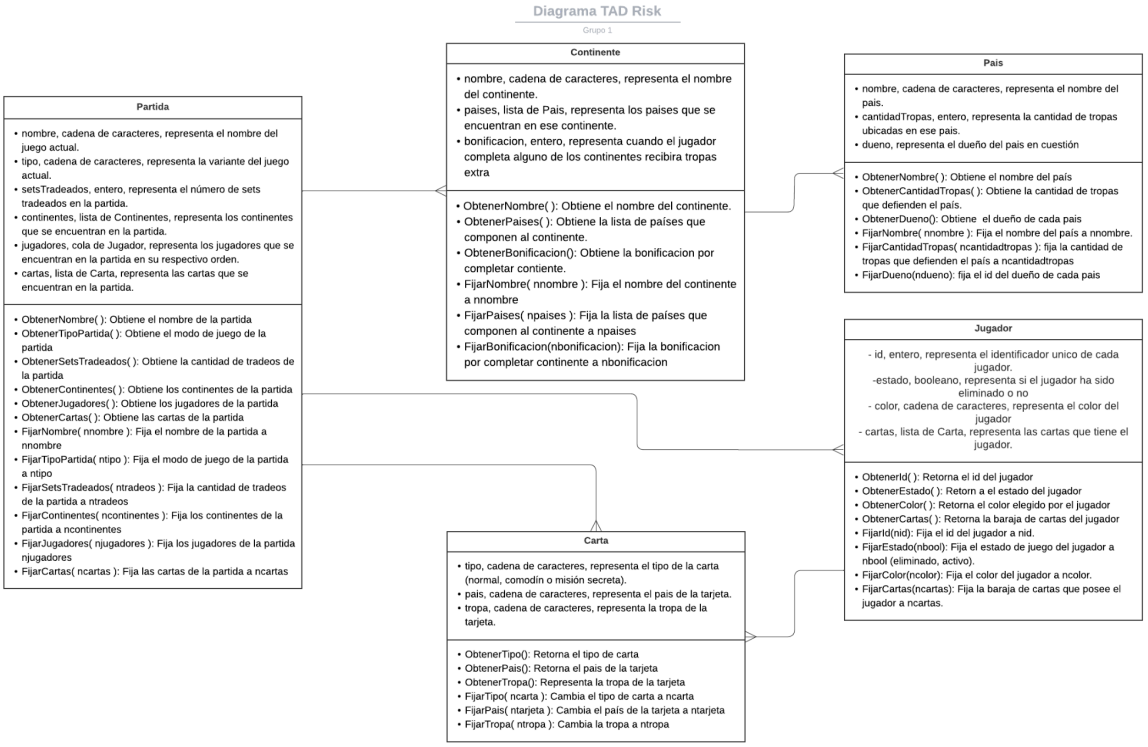

Figura 11. Diagrama de TADS. Fuente: Elaboración propia.

Planes de Prueba

Inicializar

Plan de pruebas: Nombre de partida			
Descripción del caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
Se ingresa solo letras	Cualquier letra del abecedario	Se crea la partida correctamente	Continua con el flujo normal
Se ingresa solo números	Cualquier numeros	Se crea la partida correctamente	Continua con el flujo normal
Se ingresa solo caracteres especiales	Cualquier carácter especial	Se crea la partida correctamente	Continua con el flujo normal
Se ingresa solo un espacio	Solo un espacio	No se crea la partida	"El nombre de la partida no puede contener solo espacios en blanco."
Se ingresan dos	Mi primera partida	No se crea la partida	"El nombre de la

cosas separadas por espacio(s)	Hola mundo 2 2 ? ?		partida no puede estar separada por espacios.”

Plan de pruebas: Tipo de partida			
Descripción del caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
Se selecciona el tipo de partida “Normal”	Normal	Se selecciona a partida para inicializar como partida “Normal”	Continua con el flujo normal para inicializar como partida “Normal”
Se selecciona el tipo de partida “Misión Secreta”	Misión Secreta	Se selecciona a partida para inicializar como partida “Mision Secreta”	Continua con el flujo normal para inicializar como partida “Mision Secreta”
El usuario ingresa un una opción inexistente o incorrecto	Cualquier entrada diferente a “Normal”, “Mision secreta”, “Salir”	Opción inválida	“Opción inválida, ingrese un modo de juego”
El usuario ingresa una cadena vacía		Espera por input del usuario	“Eliga el modo de juego”
El usuario selecciona la opción “Salir”	“Salir”	Sale al menú principal	Va al flujo normal del menpu principal

Plan de pruebas: Cantidad de jugadores			
Descripción del caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
Inicializa con cantidad de jugadores negativos	Números negativos	Cantidad de usuarios no válida	“Error. Debe ingresar entre 3 y 6 jugadores”
Inicializa con menos de dos jugadores	1	Cantidad de usuarios no válida	“Error. Debe ingresar entre 3 y 6 jugadores”

Inicializa entre 3 y 6 jugadores	Números entre 3 y 6	Cantidad de usuarios válida	Continua con el flujo normal
Inicializa con más de 6 jugadores	Números mayores a 6	Cantidad de usuarios no válida	“Error. Debe ingresar entre 3 y 6 jugadores”
Inicializa con números decimales	Ingresa un número no entero	Cantidad de usuarios no válida	“Error. Debe ingresar entre 3 y 6 jugadores”
Inicializa con un carácter no numérico	Ingresa algo diferente a un número	Cantidad de usuarios no válida	“Error. Debe ingresar entre 3 y 6 jugadores”
Inicializa con una entrada vacía	No ingresa nada	Vuelve a preguntar	“Cuantos jugadores tendrá la partida ‘nombre_partida’ (min 2, max 6):”

Plan de pruebas: Elección de color del jugador			
Descripción del caso	**Valores de entrada**	**Resultado esperado**	**Resultado obtenido**
El jugador selecciona un color de la lista	Cualquier color de la lista de colores disponibles	Conecta al usuario con ese color	Continúa con el flujo normal
El jugador selecciona un color que ya ha sido escogido	Color elegido por un jugador anteriormente	Vuelve a pedir que ingrese el color	“Color: 'ingreso' no encontrado o ya fue seleccionado. Por favor ingrese un color válido
El jugador ingresa una opción inválida o inexistente	Color inexistente	Vuelve a pedir que se ingrese color	“Color: 'ingreso' no encontrado o ya fue seleccionado. Por favor ingrese un color válido
El jugador ingresa una entrada vacía		Vuelve a preguntar	Vuelve a pedir que se ingrese el color