**Contenido**

[Nota: 1](#_Toc116860341)

[Contexto del problema 3](#_Toc116860342)

[R1: Registro de jugadores 5](#_Toc116860343)

[R2: Registro de nivel 6](#_Toc116860344)

[R3: Identificar resolución 6](#_Toc116860345)

[R4: identificar y establecer nivel de un jugador 7](#_Toc116860346)

[R5: Identificar y establecer la complejidad de un nivel. 8](#_Toc116860347)

[R6: Registro de tesoros 10](#_Toc116860348)

[R7: Asociar tesoro a un nivel 11](#_Toc116860349)

[R8: Registro de enemigos 12](#_Toc116860350)

[R9: Asociar enemigos a un nivel 14](#_Toc116860351)

[R10: Modificar puntaje de un jugador 15](#_Toc116860352)

[R11: informar los tesoros y enemigos de un nivel 16](#_Toc116860353)

[R12: informar la cantidad de un tesoro dentro del juego 17](#_Toc116860354)

[R13: informar el tesoro más repetido en todos los niveles. 18](#_Toc116860355)

[R14: informar la cantidad de consonantes encontradas en los nombres de los enemigos del juego 19](#_Toc116860356)

[R15: informar el top 5 de los jugadores con mayor puntajes 20](#_Toc116860357)

[R16: Informar la cantidad encontrada de un tipo de enemigo en todos los niveles. 21](#_Toc116860358)

[R17: Informar el enemigo que otorga mayor puntaje y el nivel donde se ubica. 22](#_Toc116860359)

[Trazabilidad 23](#_Toc116860360)

# Contexto del problema

|  |  |
| --- | --- |
| CLIENTE | Empresa de videojuegos |
| USUARIO | Jugadores del videojuego.  Administradores del videojuego. |
| REQUERIMIENTOS FUNCIONALES | R1: Registro de jugadores.  R2: Registro de nivel.  R3: Identificar resolución.  R4: identificar y establecer nivel de un jugador.  R5: Identificar y establecer la complejidad de un nivel.  R6: Registro de tesoros.  R7: Asociar tesoro a un nivel.  R8: Registro de enemigos.  R9: Asociar enemigos a un nivel.  R10: Modificar puntaje de un jugador.  R11: Informar los tesoros y enemigos de un nivel.  R12: Informar la cantidad de un tesoro dentro del juego  R13: Informar el tesoro más repetido en todos los niveles.  R14: Informar la cantidad de consonantes encontradas en los nombres de los enemigos del juego  R15: Informar el top 5 de los jugadores con mayor puntajes.  R16: Informar la cantidad encontrada de un tipo de enemigo en todos los niveles.  R17: Informar el enemigo que otorga mayor puntaje y el nivel donde se ubica. |
| CONTEXTO DEL PROBLEMA | Una empresa de videojuegos necesita que usted colabore con un videojuego, el juego consta de 10 niveles en los que el jugador recoge tesoros y pelea con enemigos.  Han solicitado su ayuda para analizar los requerimientos y realizar el modelo de clases, responsabilidades y colaboraciones, por ahora SOLO para las funcionalidades del sistema relacionadas con el registro de jugadores, niveles, tesoros y enemigos. |
| REQUERIMIENTOS NO FUNCIONALES | * El despliegue de los tesoros y enemigos de un nivel, en la aplicación web no debe tardar más de 2 segundos. * El sistema debe funcionar tanto como aplicación web como en una app para móviles. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R1: Registro de jugadores | | |
| Resumen | El sistema debe permitir la creación de jugadores, los cuales tendrán un nombre, un nivel, un apodo único que servirá como identificador, un puntaje y un número de vidas. Cabe recalcar que el sistema permite como máximo 20 jugadores. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| nickname | String |  |
| name | String |  |
| Actividades generales necesarias para obtener los resultados | 1. Verificar que exista un espacio en el array del controller que almacena los objetivos tipo Player. 2. Si existe un espacio, revisar si el nickname dado es único. 3. En caso tal que se cumplan las anteriores condiciones, se debe crear un objeto Player teniendo como parámetros del constructor el nickname y name brindados, y posteriormente, se declara el puntaje inicial, nivel y número de vidas en 10, 0 y 5 respectivamente. 4. Posteriormente, se debe identificar y establecer el nivel en el que está el jugador, pues, dependiendo de la cantidad de puntos necesarias para pasar de un nivel a otro, puede que este inicie en un nivel distinto de 0. 5. El objeto Player creado debe ser almacenado en el array de la clase controller que administra los objetos tipo Player. | | |
| Resultado o postcondición |  | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Alerta el nickname no es único | String | En caso tal que el nickname coincida que el de otro Player. |
| Alerta no hay espacio para el almacenamiento del objeto | String | En caso tal que no exista un espacio en el array contenedor players. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R2: Registro de nivel | | |
| Resumen | El sistema debe registrar automáticamente los 10 niveles del juego, por lo tanto, sus dos atributos, *identificador* y *puntos requeridos para pasar de nivel*, son asignados por el sistema. Para el caso de los *puntos requeridos para pasar de nivel*, el sistema pide una constante de crecimiento del puntaje entre nivel y nivel, lo cual servirá de guía para la asignación de los puntos requeridos de los siguientes niveles. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| constante de crecimiento | int |  |
| Actividades generales necesarias para obtener los resultados | Establecer un bucle para crear 10 instancias de la clase nivel, donde su *identificador* va a ser igual al número de la iteración, y la cantidad de *puntos requeridos para pasar de nivel,* es equivalente a la constante de crecimiento acumulada en la iteración. | | |
| Resultado o postcondición | Los 10 objetos de tipo Level deben ser en el array de la clase controladora que administra los objetos de tipo Level. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R3: Identificar resolución | | |
| Resumen | El sistema debe consultar al usuario qué tipo de resolución maneja su dispositivo. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del objeto que corresponde a la resolución escogida por el usuario. | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Mostrar al usuario los tipos de resoluciones. 2. Consultar cuál de las resoluciones desplegadas corresponde a la que maneja su dispositivo. 3. A partir de la respuesta dada en el anterior punto, identificar el objeto de la enumeración de tipos de resolución que corresponde a dicha resolución. 4. Asignar el objeto encontrado en el punto 4 a el atributo resolution de la clase controller. | | |
| Resultado o postcondición | Se identifica y asigna la resolución del dispositivo del usuario a la clase controller. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R4: identificar y establecer nivel de un jugador | | |
| Resumen | El sistema debe identificar el nivel de un jugador basándose en la puntuación acumulada por el mismo y el puntaje requerido para pasar de nivel de cada nivel. Por último, de debe notificar si el jugador aumentó, descendió, o mantuvo el nivel. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del Player en la array players | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Inicialmente, se debe mostrar los jugadores hasta el momento creados con su respectiva posición del arreglo para que el usuario pueda elegir el jugador al que quiere identificar y establecer su nivel. 2. Obtener de la array contenedora de los players el objeto Player que corresponde a la posición dada por el usuario. 3. De dicho Player se debe identificar en qué nivel se encuentra comparando la puntación que él posea y la puntación necesaria para pasar de nivel de cada nivel. | | |
| Resultado o postcondición | Al atributo level del objeto player se le debe asignar el identificador del nivel en el que quedó, ya sea que haya aumentado, descendido o mantenido en el mismo nivel. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Alerta: el objeto player descendió de nivel | String | El nivel en el que quedó el player es menor en comparación al que estaba. |
| Alerta: el objeto player aumentó de nivel | String | El nivel en el que quedó el player es mayor en comparación al que estaba. |
| Alerta: el objeto player conservó el nivel. |  | El player mantuvo el nivel. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R5: Identificar y establecer la complejidad de un nivel. | | |
| Resumen | El sistema debe asignar una complejidad baja, media, o alta a un nivel basándose en: si la suma de los puntos que otorgan los tesoros es mayor que la suma de los puntos que dan los enemigos cuando son vencidos, entonces es de complejidad baja el nivel. Si es igual, es medio, y si es mayor la suma de los puntos de los enemigos que la suma de los puntos de los tesoros, entonces el nivel es alto. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del level a operar | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Desplegar los niveles al usuario con su respectiva posición del array que lo contiene. 2. Obtener de la array contenedora de los levels, el objeto que corresponde a la posición dada por el usuario. 3. Comparar la suma de los puntos que otorgan los tesoros con la suma de los puntos que quitan los enemigos. 4. Según sea el resultado de la comparación (menor que, igual, o mayor que) asignar un objeto del tipo ComplexityType que corresponda con el grado de complejidad obtenido. En caso tal que no existan ni enemigos, ni tesoros asociados al nivel, no se debe asignar una complejidad. | | |
| Resultado o postcondición | Se establece la complejidad de un nivel, indicando si es de tipo bajo, medio, o alto, siempre y cuando este tenga al menos un enemigo o un tesoro asociado. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Notificar la complejidad de l nivel | String | En caso tal que previamente se haya agregado como mínimo un tesoro o un enemigo. |
| Notificar si no se asignó complejidad | String | En caso tal que no se haya agregado como mínimo un tesoro o un enemigo. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R6: Registro de tesoros | | |
| Resumen | El sistema debe permitir la creación de tesoros, los cuales tendrán como atributos un nombre, una URL de la imagen que los representa, y un puntaje que representa el valor que otorga este al ser encontrado, y una posición con coordenadas X y Y que expresan en que parte de la pantalla se encuentra el objeto. Cabe recalcar que los posibles valores de X y Y están dados por la resolución del dispositivo del usuario. Por otro lado, cada nivel puede almacenar hasta 50 copias de los 50 posibles tesoros molde de la clase Controller. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| nombre | String |  |
| url | Strng |  |
| Puntaje que otorga | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Verificar que exista un espacio en el array que almacena los objetivos tipo Treasure. 2. En caso tal que se cumpla la anterior condición, se debe crear un objeto Treasure teniendo como parámetros del constructor el nombre, url y *puntaje que otorga* brindados por el usuario. 3. Verificar que el objeto creado en el punto 2 no sea un duplicado de un treasure existente. 4. Si se cumple el punto 4, entonces el objeto Treasure creado debe ser almacenado en el array de la clase controller que administra los objetos tipo Treasure. | | |
| Resultado o postcondición | Se obtiene un objeto tipo Treasure único, el cual queda almacenado en el array de la clase controller que administra los objetos tipo Treasure. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R7: Asociar tesoro a un nivel | | |
| Resumen | El sistema debe permitir que se asocie una cantidad X de tesoros a un nivel. Cabe recalcar que, un nivel puede almacenar como máximo 50 tesoros. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| treasuresQuantity | int |  |
| Posición del nivel a operar | int |  |
| Posición del tesoro a usar | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Se debe verificar si hay tesoros creados. 2. Se debe verificar si existe espacios suficientes en la array de tesoros del nivel para almacenar la cantidad de tesoros que desea asociar el usuario. 3. En caso tal que se cumpla con las condiciones anteriores, desplegar los tesoros al usuario con su respectiva posición del array que lo contiene. 4. Posteriormente, se obtiene del array contenedora de los treasures, el objeto que corresponde a la posición dada por el usuario. 5. Del anterior objeto encontrado se debe crear tantas copias como la entrada *treasuresQuantity* lo indique. Además, a cada copia se le debe asignar al atributo position una coordenada generada aleatoriamente dentro del tamaño de la resolución del dispositivo del usuario. 6. Se debe obtener de la array contenedora de los *levels*, el objeto que corresponde a la posición dada por el usuario. 7. Asociar las copias generadas al array contenedor de los *treasures* del nivel obtenido en el punto 5. 8. Identificar y establecer complejidad al nivel al cual fue asociado los tesoros. | | |
| Resultado o postcondición | Finalmente se tiene que, las cantidad de copias pedidas de un tesoro especifico de la clase controller quedan asociados a un nivel elegido por el usuario. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Alerta no hay tesoros creados | String | En caso tal que no haya tesoros creados. |
| Alerta no hay espacio | String | En caso tal que no haya espacio en el array contenedora de *treasures* del nivel. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R8: Registro de enemigos | | |
| Resumen | El sistema debe permitir la creación de enemigos, los cuales tendrán como atributos: un nombre único que usaremos como identificador, un tipo, un puntaje que brinda al ser derrotado, un puntaje que resta al jugador en caso de que este sea vencido por el enemigo, y una posición con coordenadas X y Y que expresan en que parte de la pantalla se encuentra el objeto. Cabe recalcar que, los posibles valores de X y Y están dados por la resolución del dispositivo del usuario. Por último, el sistema puede almacenar hasta 25 enemigos únicos en la clase Controller. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| name | String |  |
| positionType | int |  |
| scoreToGive | int |  |
| scorteToSubtract | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Verificar si existe un espacio para agregar un enemigo. 2. Verificar si el atributo *name* dado por el usuario es único dentro de sus similares. 3. En caso tal que se cumplan amabas condiciones, obtener del array de enemyTypes el objeto que corresponde con la posición dada por el usuario. 4. Crear una instancia de la clase Enemy, donde el constructor tome como parámetros el *name*, el objeto *enemyType* encontrado en el punto 4 y las entradas *scoreToGive* y *scoreToSubtract.* 5. Guardar el objeto creado en el punto 4 en el array del controller que administra los objetos del tipo Enemy. | | |
| Resultado o postcondición | Se obtiene un objeto único del tipo Enemy, el cual es almacenado en el array del controller que gestiona los objetos del tipo Enemy. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Alerta no hay espacios para almacenar una instancia de Enemy |  | En caso tal que no haya espacios vacíos en el array del controller que gestiona los objetos tipo Enemy. |
| Alerta el nickname no es único | String | En caso tal que el nickname coincida que el de otro Player. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R9: Asociar enemigos a un nivel | | |
| Resumen | El sistema debe permitir que se asocie un enemigo en un nivel especifico en caso tal de que este anteriormente no se haya asociado. Cabe recalcar que cada nivel puede almacenar a lo sumo 25 enemigos. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del enemigo a asociar | int |  |
| Posición del nivel a usar | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Desplegar los niveles al usuario con su respectiva posición del array que lo contiene. 2. Verificar si existe enemigos creados. 3. Si se cumple la condición anterior, obtener del array de enemigos del controller el objeto de tipo Enemy que corresponde con la posición dada por el usuario, y obtener el nivel elegido por el usuario de acuerdo con la posición seleccionada. 4. Verificar si en el nivel elegido no se ha agregado el objeto tipo Enemy seleccionado. 5. En caso tal que no se haya agregado, crear una copia del objeto tipo Enemy y establecer aleatoriamente el atributo posición de este, teniendo en cuenta la resolución elegida por el usuario. 6. Asociar la copia creada en el punto 4 al array de enemigos del nivel elegido. 7. Por último, Identificar y establecer complejidad al nivel al cual fue asociado los tesoros. | | |
| Resultado o postcondición | Como resultado se asocia un objeto tipo Enemy que previamente no se había registrado en el array de enemigos del nivel seleccionado por el usuario. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Alerta: no hay espacios para agregar el objeto. | String | En caso tal que el array de enemigos del nivel no tengo espacios vacíos. |
| Alerta: Enemigo previamente registrado. | String | En caso tal que en el array de enemigos del nivel se encuentre el enemigo que se iba a asociar. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R10: Modificar puntaje de un jugador | | |
| Resumen | El sistema debe permitir que se modifique la cantidad de puntaje de un jugador en específico. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del jugador a usar | int |  |
| Puntaje por establecer | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Se debe verificar si existen jugadores creados. 2. Si se cumple la condición anterior, entonces mostrar los jugadores hasta el momento creados con su respectiva posición del arreglo para que el usuario pueda elegir el jugador al que quiere modificar su puntaje. 3. Obtener el jugador que corresponde con la posición dada por el usuario. 4. Asignar al objeto tipo Player del punto anterior el puntaje que quiere establecer el usuario. 5. Identificar y establecer el nivel en el quedó el jugador después de la modificación del puntaje. | | |
| Resultado o postcondición | Se modifica el atributo *score* de un objeto tipo Player elegido por el usuario. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  | Notificar el nivel en el que quedó el jugador. | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R11: informar los tesoros y enemigos de un nivel | | |
| Resumen | El sistema debe generar un informe donde se exprese los tesoros y enemigos que le corresponde a un nivel elegido por el usuario. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del nivel | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Desplegar los niveles al usuario con su respectiva posición del array que lo contiene. 2. Obtener el nivel que corresponde con la posición dada por el usuario. 3. Identificar y contabilizar los tesoros del nivel. 4. Identificar los enemigos de un nivel. 5. Concatenar la información encontrada en loa puntos 2 y 3. | | |
| Resultado o postcondición | Se genera una cadena de texto que expresa los tesoros y su cantidad dependiendo del tipo, y los enemigos del nivel | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  | Informe final de tesoros y enemigos | String |  |
|  | Alerta: No hay tesoros en el nivel seleccionado | String | En caso tal que la lista de tesoros del nivel seleccionado no tenga tesoros asociados. |
|  | Alerta: No hay enemigos en el nivel seleccionado | String | En caso tal que la lista de enemigos del nivel seleccionado no tenga enemigos asociados. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R12: informar la cantidad de un tesoro dentro del juego | | |
| Resumen | El sistema debe generar un informe donde se exprese la cantidad de tesoros que hay en todo el programa de un tesoro en especifico | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del tesoro | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Verificar si existen tesoros creados. 2. En caso tal que se cumpla con la condición anterior, desplegar los tesoros al usuario con su respectiva posición del array que lo contiene. 3. Posteriormente, se obtiene del array contenedora de los treasures, el objeto que corresponde a la posición dada por el usuario. 4. Contar por cada nivel la cantidad de tesoros que hay del tipo encontrado en el punto 2. 5. Guardar y acumular la cantidad encontrada por cada nivel. | | |
| Resultado o postcondición | Se genera una cadena de texto que expresa la cantidad de tesoros que hay en un tesoro en específico dentro del juego. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  | Informe: cantidad de tesoros de un tesoro en especifico | String |  |
| Alerta: No hay tesoros creados | String | En caso tal que no haya al menos un tesoro creado. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R13: informar el tesoro más repetido en todos los niveles. | | |
| Resumen | El sistema debe generar un informe donde se indique cuál es el tesoro más repetido dentro del juego. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  |  |  |
| Actividades generales necesarias para obtener los resultados | 1. Verificar si existen tesoros creados. 2. Contar y almacenar por cada nivel la cantidad de tesoros que hay por cada tipo de tesoro. 3. Comparar la cantidad obtenida por cada tesoro y mostrar el tesoro que tenga la mayor cantidad. En caso tal que sean varios, mostrar todos los que alcanzaron la cantidad máxima. | | |
| Resultado o postcondición | Se genera una cadena de texto que expresa el tesoro más repetido en todo el juego. En caso tal que haya empate con otro tesoro, mostrar todos los tesoros que alcanzan la cantidad máxima. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  | Tesoro más repetido | String |  |
| Alerta: No hay tesoros creados | String | En caso tal que no haya al menos un tesoro creado. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R14: informar la cantidad de consonantes encontradas en los nombres de los enemigos del juego | | |
| Resumen | El sistema debe generar un informe donde se indique la cantidad de consonantes encontradas en los nombres de los enemigos del juego | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Actividades generales necesarias para obtener los resultados | 1. Verificar si existe al menos un enemigo. 2. Si se cumple la anterior condición, contar y almacenar por cada enemigo la cantidad de consonantes que posee su nombres; Tener en cuenta tanto las mayúsculas como las minúsculas. 3. Sumar la cantidad de consonantes encontradas por cada enemigo. 4. Mostrar la suma final dada por el anterior punto. | | |
| Resultado o postcondición | Se genera una cadena de texto que indica la cantidad de consonantes encontradas en los nombres de los enemigos de todo el juego. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  | Cantidad de consonantes | String |  |
|  | No existen enemigos creados | String | En caso tal que no haya enemigos registrados en el juego. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R15: informar el top 5 de los jugadores con mayor puntajes | | |
| Resumen | El sistema debe generar un informe donde se indique los 5 jugadores que tienen mayor puntaje de todo el juego. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Actividades generales necesarias para obtener los resultados | 1. Verificar si existe al menos un jugador. 2. Si se cumple la anterior condición, 3. Comparar y organizar de mayor a menor puntaje los jugadores que están en el juego en un array temporal. 4. Concatenar la información de los jugadores del top 5 en una cadena de texto. 5. Imprimir la cadena de texto generada en el punto 4. | | |
| Resultado o postcondición | Se genera un reporte de los jugadores con mayor ´puntaje dentro del juego. | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Cantidad de consonantes | String |  |
| No existen enemigos creados | String | En caso tal que no haya enemigos registrados en el juego. |
| Reporte | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R16: Informar la cantidad encontrada de un tipo de enemigo en todos los niveles. | | |
| Resumen | El sistema debe generar un informe donde se exprese la cantidad de un tipo de enemigo elegido por el usuario. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Posición del tipo de enemigo seleccionado | int |  |
| Actividades generales necesarias para obtener los resultados | 1. Verificar si existe al menos un enemigo. 2. Si se cumple la anterior condición, contar por cada nivel la cantidad de enemigos que coinciden con el tipo de enemigo seleccionado. 3. Imprimir en pantalla la cantidad encontrada del enemigo seleccionado. | | |
| Resultado o postcondición | Se genera una cadena de texto que indica la cantidad de enemigos encontrados en todo el juego de un tipo de enemigo en especifico | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  | Reporte de tipo de enemigos encontrados | String |  |
|  | No existen enemigos creados | String | En caso tal que no haya enemigos registrados en el juego. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | R17: Informar el enemigo que otorga mayor puntaje y el nivel donde se ubica. | | |
| Resumen | El sistema debe generar un informe donde se exprese el enemigo que otorga mayor puntaje dentro del juego, e indicar en cuál nivel se encuentra. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  |  |  |
| Actividades generales necesarias para obtener los resultados | 1. Verificar si existe al menos un enemigo. 2. Si se cumple la anterior condición, identificar y guardar el puntaje más alto que puede otorgar un enemigo que esté asociado a algún nivel. 3. Iterar los enemigos de todos los niveles para identificar los enemigos que poseen ese puntaje máximo encontrado en el punto 2. 4. Imprimir en pantalla los enemigos con su respectivo nivel que poseen el puntaje más alto encontrado | | |
| Resultado o postcondición | Se genera una cadena de texto que contiene la información de los enemigos que brindan mayor puntaje al ser derrotados | | |
| Salidas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
|  | Reporte de tipo de enemigos encontrados | String |  |
|  | No existen enemigos creados | String | En caso tal que no haya enemigos registrados en el juego. |

# Trazabilidad

|  |  |  |
| --- | --- | --- |
| **Requerimiento Funcional** | **Nombre de la Clase** | **Nombre del método** |
| R1: Registrar jugadores | Clase Controller | IsThereSpaceToRegisterAPlayer(): boolean  isNicknameUnique(String) : boolean  registerPlayer(String, String): void  getPlayers (): Player [] |
| Clase Manager | registerPlayer() : void |
| Clase Player | Player(String, String) : void  getNickname (): String |
| R2: Registrar niveles | Clase Level | Level (int, int) |
| Clase Controller | getLevels() : Level []  initLevels (int) : void  Controller (int) |
|  | Clase Manager | initValues() : void  askForIncrementValueToInitLevels() : void |
| R3: Identificar resolución. | Clase Controller | getTypeOfResolution() : String  establishResolution(int) : void  setResolution(TypeOfResolution) : void |
| Clase TypeOfResolution | values() : TypeOfResolution  toString() : String |
| Clase Manager | askForResolution() |
| R4: Identificar y establecer nivel de un jugador. | Clase Controller | identifyAndEstablishLevel (int) : String  getLevels() : Level []  getPlayers () : Player []  isTherePlayers() : boolean  concatenatePlayers () : String |
| Clase Manager | identifyAndEstablishLevel () : void |
| Clase Player | getScore () : int  setLevel (int) : void  getLevel () : int |
| Clase Level | getScoreRequiredToPass() : int |
| R5: Establecer complejidad de un nivel. | Clase Controller | toString() : String  getLeveles() : Level[] |
| Clase Manager | establishLevelComplexity () : void |
| Clase Level | toString () : String  getLevelTreasures () : Treasue []  getLevelsEnemies () : Enemy []  setComplexity (ComplexityType) : void  getComplexity() : ComplexityType |
| Clase Controller | establishAndIdentifyComplexity (int ) : String  getLevels () : Level[] |
| Clase Treasure | getScoreToGive() : int  getLevelEnemies () : int |
| R6: Registrar tesoro | Clase Controller | isThereSpaceToRegisterATreasure() : boolean  getTreasure (): Treasure []  registerTreasure(String, int, String) : boolean |
| Clase Manager | registerTreasure() : void |
| Clase Treasure | Treasure(String, int, String)  isEqual(Treasure) : boolean |
| R7: Asociar tesoro a un nivel | Clase Controller | isThereTreasures() : boolean  concatenateTreasures() : String  concatenateLevels () : String  getLevels () : Level []  getTreasures () : Treasure []  associateTreasureToLevel(int, int, int) : boolean  establishAndIdentifyComplexity (int) : String  getResolution () :TypesOfResolution  associateTreasureToLevel(int, int, int) : boolean |
| Clase Treasure | Treasure(Treasure, int, int) |
| Enum TypesOfResolution | getHeight() : int  getWidth() : int |
| Clase Level | getLevelTreasures |
| Clase Manager | associateTreasureToLevel() : void |
| R8: Registro de enemigos | Clase Controller | getEnemies() : Enemy []  registerEnemy(String, int, int, int) : boolean  isThereSpaceToRegisterEnemy() : boolean  concatenateEnemyTypes() : String |
| Clase Manager | registerEnemy() : void |
| Clase Enemy | getName () : String  Enemy (String, EnemyType,int,int) |
|  |  |
| R9: Asociar enemigo a un nivel| | Clase Controller | getEnemies() : Enemy []  getLevels () : Level []  isThereEnemies (): boolean  associateEnemyToLevel (int,int) : boolean  establishAndIdentifyComplexity (int) : String |
| Clase Manager | associateEnemyLevel () : void |
| Clase Level | getLevelEnemies () : Enemy [] |
| Clase Enemy | getName () : String |
| R10: Modificar puntaje | Clase Player | toString() : String  setScore (int) : void  getLevel () : int |
| Clase Controller | getPlayers() : Player []  concatenatePlayers () : String  modifyPlayerScore(int, int) : void  isTherePlayers () : boolean  identifyAndEstablishLevel (int) : String  concatenateLevels() : String |
| Clase Manager | modifyPlayerScore () : void |
| R11: Informar los tesoros y enemigos de un nivel | Clase controller | treasuresAndEnemiesOfaLevel (int) : String  getLevel() : Level[]  getTreasures () : Treasure[ ]  getLevelTreasure () : Treasure[] |
| Clase Treasure | getName () : String |
| Clase Enemy | getName () : String |
| Clase Manager | treasuresAndEnemiesOfaLevel () : void |
| R12: Informar la cantidad encontrada de un tesoro en todos los niveles | Clase Controller | amountOfASpecificTreasure (int) : String  getLevels () : Level []  getTreasure () : Treasure []  isThereTreasures () : boolean |
| Clase Level | getLevelTreasures () : Treasure [] |
| Clase Treasure | getName () : String |
| Clase Manager | amountOfASpecificTreasure () : void |
| R13: informar el tesoro más repetido en todos los niveles. | Clase Level | getLevelTreasures() : Treasure [] |
|  | Clase Controller | getTreasures() : Treasure []  getLevels() : Level [] |
| Clase Manager |  |
| Clase Treasure | isEqual(Treasure): boolean  getName() : String |
| R14: informar la cantidad de consonantes encontradas en los nombres de los enemigos del juego | Clase Controller | amountOFConsonants () : void  getEnemies () : Enemy [] |
| Clase Enemy | getName () : String |
| R15: informar el top 5 de los jugadores con mayor puntajes | Clase Controller | isTherePlayers () : boolean  getPlayers () : Level [] |
| Clase Player | getScore () : String  getNickname () : String |
| R16: Informar la cantidad encontrada de un tipo de enemigo en todos los niveles. | Clase Controller | amountOfSpecificEnemyType(int) : String  getLevels() : Level [] |
| Clase EnemyType | values() : EnemyType []  toString() : String |
| Clase Enemy | getType() : EnemyType |
| Clase Manager | amountOfSpecificEnemyType () : void |
|  | Clase Level | getLevelEnemies() : Enemy [] |
|  |  |  |
| R 17: Informar el enemigo que otorga mayor puntaje y el nivel donde se ubica. | Clase Controller | getLevels () : Level [] |
|  | Clase Level | getLevelEnemies() :Enemy [] |
|  | Clase Enemy | getScoresToGive () : int  getName () : String |