Laura Rubio – A00346353

Douglas López – A00347533

David Obando – A00348505

Requerimientos:

|  |  |
| --- | --- |
| Nombre | Clasificación de usuarios |
| Resumen | Se encarga de clasificar los jugadores segùn su destreza en el juego |
| Entrada | Usuarios |
| Salida | Clasificados segùn su destreza |

|  |  |
| --- | --- |
| Nombre | Latencia por jugador |
| Resumen | Se encarga de optimizar el ping de los jugadores dentro de las partidas para asegurar una igualdad de condiciones. |
| Entrada |  |
| Salida | Ping optimizado |

|  |  |
| --- | --- |
| Nombre | Modo plataforma |
| Resumen | Proponer una implementación para el modo plataforma |
| Entrada |  |
| Salida | Modo plataforma hecho. |

|  |  |
| --- | --- |
| Nombre | Lanzar nuevo modo de juego de San Valentín |
| Resumen | Se encarga de implementar una nueva modalidad de juego pero que sea en base a la celebración del día de San Valentín. |
| Entrada |  |
| Salida | Modo San Valentín 2018 |

1. Descripción del problema:

El problema radica en algunos inconvenientes que ha presentado el juego durante sus partidas. Entre esos:

* Dificultad de emparejamiento, ya que no incorpora en una partida jugadores de niveles similares.
* La latencia varía mucho en una partida, por lo tanto el ping exhibe problemas durante el juego.
* El constante emparejamiento de plataformas (como PlayStation, Nintendo, Xbox, entre otras) en una sola partida, genera inconvenientes frente a los usuarios.
* Las abundantes recomendaciones demandan creaciones de variantes dentro del sistema de juego, para así celebrar ocasiones especiales, como lo es el día de San Valentín.

**2. Recopilación de información**

**Características de Fornite:**

Emparejamiento:

El Matchmaking o emparejamiento es el sistema presente en la mayoría de juegos multijugador el cual trata de enfrentar jugadores o equipos de un nivel similar. Fornite realiza este tipo de emparejamiento teniendo en cuenta la habilidad de cada jugador, así los jugadores tendrán oponentes similares.

Modalidad:

Para conocer sobre cómo se celebró la última fecha de San Valentín en Fornite, se recurrió a buscar las características más relevantes durante esa temporada, en la cual añadieron nuevos tipos de misión, heroes temáticos y una ballesta especial con munición infinita, así mismo se añadieron mejoras a las armas existentes y adicionalmente se realizaron pequeñas modificaciones en el mapa de juego.

Factores del problema:

Ping: Como programa, ping es una utilidad diagnóstica en redes de computadoras que comprueba el estado de la comunicación del host local con uno o varios equipos remotos de una red IP. Mediante esta utilidad puede diagnosticarse el estado, velocidad y calidad de una red determinada. Muchas veces se utiliza para medir la latencia o tiempo que tardan en comunicarse dos puntos remotos, y por ello, se utiliza el término **PING** para referirse al lag o latencia de la conexión en los juegos en red.

Teniendo en cuenta el problema planteado, el cual exige una alta eficiencia para su resolución se realizó una búsqueda sobre las estructuras de datos que se muestran a continuación:

Tabla Hash:

Las tablas hash son estructuras de datos que se utilizan para almacenar un número elevado de datos sobre los que se necesitan operaciones de búsqueda e inserción muy eficientes. Una tabla hash almacena un conjunto de pares “(clave, valor)”. La clave es única para cada elemento de la tabla y es el dato que se utiliza para buscar un determinado valor.

Árbol:

Un árbol consiste en una colección de nodos, los cuales se encuentran ordenados de forma jerárquica, cada nodo es una estructura de datos con un valor y a su vez con una lista de referencias a los nodos hijos. Este tipo de estructura permite las operaciones básicas: buscar, eliminar y agregar.

(Diapositivas)

Cola:

Es una estructura de datos que se caracteriza por realizar la operación de inserción en un extremo y la de extracción por otro (FIFO). Las colas se utilizan para trabajar con objetos o eventos que son tomados como datos y se almacenan mediante colas para su posterior procesamiento. De las colas se deriva un denominada Cola de prioridades, la cual adicionalmente de las funciones anteriores contiene una prioridad asignada, un elemento con mayor prioridad que será desencolado antes que un elemento de menor prioridad.

Lista doblemente enlazada:

Consiste en un conjunto de nodos enlazados secuencialmente, cada uno de ellos contiene dos enlaces, un siguiente y un anterior, con la intención de facilitar el recorrido de la lista, sin embargo se debe tener en cuenta una desventaja y es que las operaciones de agregar o eliminar requieren la modificación de muchos enlaces para mantener la estructura.

Generics:

Está es una herramienta que nos permite definifr una clase, una interfaz o método, su objetivo fundamental es independizar el tipo de datos a definiciones de modelos más generales, tiene ventajas muy significativas como: corrección de errores en tiempo de compilación, lo que implica que hallarlos y solucionarlos es mucho más rápido y sencillo, otra de sus principales ventajas es la eliminación de casteos, lo cual permite un código más comprensible.

**3. Búsqueda de soluciones creativas.**

Se necesita una solución al problema; para esto se basó en la búsqueda de estructuras de datos que me permitieran solucionar las necesidades del juego con mayor eficiencia, debido a que, se van a manejar números de datos bastante grandes . Por ende, optamos por una búsqueda de soluciones exhaustiva a través listas de revisión para examinar los diferentes puntos, áreas y posibilidades de diseño utilizando estructuras de datos.

Para revisar nos apoyamos en unas ciertas preguntas para poder alcanzar la mejor alternativa de solución a nuestro problema:

¿Es realmente eficiente el proceso de ejecución del algoritmo?

¿Es una estructura que almacene datos grandes?

¿Permite una búsqueda, eliminación e inserción eficiente?

Después de una rigurosa recopilación de información, procedemos a tomar alternativas que nos ayudarán a enfocar nuestras necesidades hacía determinadas posibles resoluciones, cabe aclarar que, no son definitivas, pero sí admisibles.

Cuando estamos en este proceso de búsqueda de soluciones creativas, debemos saber cosas importantes acerca de las alternativas que nos vamos encontrando durante todo este proceso del método de diseño de la ingeniería, por tanto, a continuación nos percatamos en informaciones encontradas en fuentes altamente visitadas por los usuarios.

ALTERNATIVAS:

1. Tabla hash:

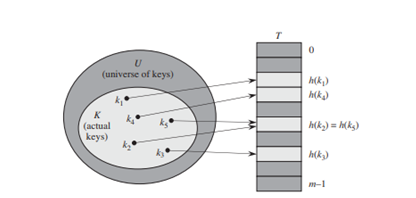
Es una estructura basada en Dictionary Data, y al igual que un Direccionamiento directo, almacenan un conjunto de pares “(key, value)”. Sin embargo, a la hora de eficiencia en el espacio de memoria, usar una tabla hash requiere mucho menos almacenamiento que un direccionamiento directo. Además, mientras que con el direccionamiento directo, un elemento con la clave k se almacena en la ranura k, con hashing, este elemento se almacena en la ranura h (k). Es decir. usamos una función hash h para calcular el espacio donde queda almacenado, a partir de la clave k.

h : U → {0, 1, ..., m − 1}

-Donde m << | U | y cada elemento tiene una clave diferente.

-h (k) es el valor hash de la clave k.

-En lugar de un tamaño de | U |, la matriz puede tener tamaño m



Ahora bien, para asignar un espacio a una clave usamos las funciones hash; método de división (h(k) = k mod m) o el método de multiplicación (h(k) = m × (k A mod 1)). Por otro lado, también se puede presentar que dos claves puedan ir a la misma ranura o espacio, y a esto le llamamos colisión. Sin embargo, existen dos formas de solucionar este tipo de errores, encadenamiento y direccionamiento abierto. Evaluaremos lo que hace cada mecanismo para así definir cual es el más conveniente para responder a nuestro problema con mayor eficiencia.

1.1 Encadenamiento:

Es el mecanismo mediante el cual colocamos todos los elementos que tienen hash en el mismo espacio en la misma lista enlazada, donde el espacio j contiene un puntero a la cabeza de la lista de todos los elementos almacenados que tienen hash a j. Sin embargo,

si no hay tales elementos, el slot j contiene NIL.

1.2 Direccionamiento Abierto

Es un mecanismo en el cual todos los elementos ocupan la tabla hash, es decir que cada espacio de la tabla contiene un elemento o NIL. Además, cuando se busca un elemento, sistemáticamente se examinan los espacios de la tabla hasta que se encuentra el elemento deseado o nos aseguremos de que el elemento no está en la tabla. Por otro lado, no hay listas o elementos guardados fuera de la tabla (a diferencia del encadenamiento).

2. Cola

Es un contenedor de objetos donde se puede insertar y extraer elementos siguiendo el principio FIFO (First In, First Out).

Tiene dos operaciones básicas:

Enqueue (para insertar un elemento)

Dequeue (para extraer un elemento)

Tiene acceso limitado ya que uno solo puede insertar un elemento en la parte posterior de la cola y extraerlo de su frente. Por lo tanto, en su implementación se usan dos punteros, uno que me indique el ultimo y otro que me indique el primero de la cola usando, por lo general, una lista doblemente enlazada. .

3. Pila

Es un contenedor de objetos donde se pueden insertar y extraer elementos de acuerdo con el principio LIFO.

Tiene dos operaciones básicas:

Push(insertar un elemento)

Pop (extraer un elemento)

Se considera una estructura de datos de acceso limitado, ya que solo se puede insertar y extraer desde su parte superior. Por lo tanto, en su implementación se usan un puntero que me indique el elemento que está encima de la pila usando por lo general, una lista enlazada simple.

4. En cuanto a soluciones para San Valentín, la primera alternativa consiste en agregar una skin roja de un corazón que cubra todo el cuerpo, con una aureola sobre la cabeza del jugador que esta sea de color azul claro. Los jugadores deben tener un puntaje mínimo para conseguir el traje.

5. La segunda alternativa con respecto al evento de San valentín es un arma de color rosada, que lance bombas de corazones y permita a los jugadores disparar balas de forma infinita.

Evaluación y selección de la mejor solución:

|  |  |
| --- | --- |
| Estructura |  |
| Árbol | Esta primera estructura de datos, consideramos descartarla de las posibles soluciones dado que, para el planteamiento del problema no necesitamos aplicar una jerarquía entre los jugadores que buscan partidas, no necesitamos encontrar a los jugadores con las mejores habilidades, sino encontrar habilidades similares para su correcto emparejamiento. |
| Pila | Como se mencionó anteriormente una pila cuenta con una característica denominada LIFO, esta consiste en último en entrar primero en salir, para las necesidades del problema no podemos asegurarnos de que esta propiedad se pueda cumplir al momento de evaluar cada jugador. |
| Lista doblemente enlazada | Esta estructura de datos se aproxima un poco más a la solución que buscamos, cuenta con varias ventajas, sin embargo su tiempo de búsqueda incluso si la lista se encuentra ordenada es en promedio O(n) realiza una búsqueda lineal, para acceder a los jugadores se debe recorrer muchos enlaces en el caso de que se esté trabajando con un número muy grande de datos, por lo anterior, la lista doblemente enlazada no se considera apta para el problema planteado. |
|  |  |
| Cola | La cola es una estructura que nos permite utilizar el criterio de FIFO, por lo tanto es una posible alternativa para la solución. Esta nos permite que el primer jugador que ingrese en busca de una partida sea el primero en encontrar una partida con jugadores que tengan una destreza similar. |
| Tabla Hash | Tabla Hash cuenta con una gran variedad de ventajas, es por esto que la incluimos como posible alternativa para el problema, gracias a la función hash podemos encontrar jugadores con habilidades similares de manera muy eficiente, incluso si es un número muy grande de jugadores. |

**4. Transición de las ideas a los Diseños Preliminares**

. Nuestro propósito en esta fase es determinar unas características que permitan diseñar una información que satisfaga los requerimientos, funciones y restricciones establecidas con las fases anteriores. Tenemos una variedad de alternativas para cada requerimiento, pero no todas cumplen con sus principios básicos para solución a las necesidades del problema.

A continuación discutiremos un poco, acerca de las alternativas que se descartaron, ya que no se tuvieron en cuenta para la solución de nuestro problema, y esto se debe a unas ciertas particularidades presentes o características que tiene cada una de estas, en el cual no abarcaban todas las exigencias del problema.

Hasta ahora, conectando con las fases anteriores, nos encontramos de que las estructuras e ideas descartadas son las siguientes:

* Árbol binario
* Pila
* Lista doblemente enlazada
* Skin para San Valentín

Seguidamente de lo anterior, se procederá a explicar y a justificar la causa de nuestro descarte, pero, cabe aclarar que, esto va muy relacionado a los diferentes propósitos que se plantearon en las fases anteriores, esto quiere decir que, se tomarán las percepciones usadas anteriormente para analizarlas, evaluarlas y refinarlas, para así comprenderlas y pasar a un diseño preliminar de nuestras alternativas seleccionadas como parte de nuestra resolución al problema.

**Árbol** **binario.**

Un árbol binario es una estructura de datos, que nos permite almacenar datos de forma jerarquizada; ésto inicia prácticamente de una raíz, generando unos nodos derechos e izquierdos (también llamados hijos) desde la raíz, implicando así que, en la parte izquierda de la raíz estarán los datos con menor valor y en la derecha los datos con mayor valor.

Teniendo en cuenta lo anterior y relacionándolo con nuestros requerimientos, vemos que no necesitaremos una jerarquización en cuanto a un valor o dato, ya que simplemente se requiere almacenar jugadores sin importar un orden, esto quiere decir que, no nos importará saber quién es el más habilidoso o el menos habilidoso, solamente queremos buscar jugadores con destrezas similares. Por lo anterior, se descartó esta estructura de dato, ya que si hacemos una pre-evaluación a nuestro caso, nos tomaría cierta dificultad llegar a buscar dentro del árbol niveles o destrezas similares dado un jugador.

**Pila**

Una pila es otra estructura de dato o lista ordenada, que nos permite almacenar y recuperar datos, dándonos un modo de acceso a sus elementos de tipo LIFO (Last in, First out = Último en entrar, primero en salir).

Para nuestro problema, pensamos que podría servir esta estructura para almacenar a nuestros jugadores, pero para poder empezar una partida con jugadores de destreza semejante no tendría sentido que los últimos que entran sean los primeros en salir al campo de batalla, por lo que conlleva a un desorden en el emparejamiento; además de que no conocemos en realidad el último jugador que entra, por esto, se descartó la idea del uso de una pila para el correcto funcionamiento de nuestro aplicativo.

**Lista doblemente enlazada**

Básicamente, es una estructura de datos que consiste en una serie de nodos enlazados que nos permite almacenar una variedad de datos.

De aquí partimos que, la lista nos permitirá almacenar los objetos o jugadores de nuestro juego, sin embargo, esto no es suficiente, ya que no nos ofrece una gran eficiencia para un gran almacenamiento de jugadores; recordemos que Fortnite es un juego altamente preferido por los usuarios Gamers, por esto la cantidad de jugadores son extensos. Por lo tanto, esta estructura fue descartada para el uso en nuestro aplicativo por su poca eficiencia.

**Skin para San Valentín**.

Nuestra última alternativa descartada fue la Skin para la celebración del día de San Valentin para todas las plataformas en que está disponible Fortnite, porque un traje generalmente está de paga en la tienda, y esto genera una incertidumbre por parte de los usuarios que no tienen los recursos suficientes para obtenerlos. Esta fecha especial será más inclusiva, esto quiere decir que, no habrá necesidad de invertir dinero por parte de los usuarios para poder disfrutar de las diferentes actividades de este grandioso día dentro del juego.

**Diseño Preliminar**

Anteriormente hablamos acerca de las alternativas que consideramos descartar, pero aún nos queda lo más relevante, y es llevar a cabo las razones de nuestras alternativas que servirán para la resolución de nuestro problema (ideas no descartadas), sin embargo, estas razones se plantearon a través de ciertas ideas concretas, por esto, se formuló un diseño preliminar básico y sencillo para comprender la relación que existe entre las alternativas y el problema que se nos plantea.

Antes de analizar el diseño preliminar, cabe recalcar que se inducirá una estructura de diseño, que a continuación se plantea:

* Ideas
* FODA
* Análisis
* Conclusión

En primer lugar, las ideas se trataron un poco dispersas, por eso es un proceso que no es organizado, pero es importante para una buena disidencia de alternativas, aunque esto no quiere decir que es un mal comienzo, ya que es una base para el seguimiento de nuestras escogencias. Algunas de nuestras ideas fueron:

- En el emparejamiento de Fornite, cuando se entra a una partida a combatir, necesariamente va entrando el que va llegando, para así evitar problemas en el momento de espera de los jugadores. En realidad, la partida no comenzará hasta que se cumpla con el límite de jugadores por partida.

- Otra de las ideas fue la busca de una herramienta que nos ayude almacenar la cantidad de jugadores que están afiliados a nuestro juego; es incuestionable que la cantidad de jugadores que pertenecen a Fortnite es bastante extensa, por lo que nos llevó a pensar en una estructura eficiente para poder suplir este requerimiento.

- La última de las ideas más relevantes fue la novedosa celebración de día de San Valentín ya que, consta de algo basado en el interés hacia nuestros usuarios; además de ser algo inclusivo que no tenga dependencia de dinero para adquirirlo. En virtud de lo anterior, se propuso una nueva arma que dispara bombas en forma de corazón.

En segundo lugar, para hacer un énfasis en cada idea, se clasificó las fortalezas, oportunidades, debilidades y amenazas de nuestras alternativas para así verificar la potencialidad de nuestra resolución.

Para nuestra primera idea se tomó como base la estructura de datos **Queue** (Cola).

|  |  |  |
| --- | --- | --- |
|  | Factores internos | Factores externos |
| Puntos positivos | Fortalezas   * Eficiencia | Oportunidades   * FIFO * Se puede insertar en cualquier momento. |
| Puntos negativos | Debilidades   * Sólo se puede acceder al primero de la cola. | Amenazas   * Insertar otro tipo de dato diferente al que se declaró la cola. |

*tabla 1*

Para la **tabla 1** tenemos que los puntos positivos se asemeja más a lo planteado en el problema, es eficiente en cuanto a tiempo de ejecución y nos ofrece un sistema de obtención que se puede usar claramente en la parte del listado de jugadores para una partida, sin embargo, también tenemos puntos negativos, pero no nos afecta ampliamente para la resolución en base a nuestros puntos positivos.

Para nuestra segunda idea se tomó como base la estructura de datos **Tabla Hash**

|  |  |  |
| --- | --- | --- |
|  | Factores internos | Factores externos |
| Puntos positivos | Fortalezas   * Eficiencia * Función Hash | Oportunidades   * Diferentes alternativas para su inserción. |
| Puntos negativos | Debilidades   * No es dinámica. | Amenazas   * Pasarse del tamaño de la tabla. * Colisiones |

*tabla 1.1*

Para la **tabla 1.1** nos encontramos con nuestra segunda estructura de datos, donde se nos presenta varias fortalezas y oportunidades, pero antes de tocar este punto, pasemos mejor a nuestros puntos negativos; pensemos por un momento una tabla que contenga una serie de objetos con un ID único, el cual nos ayudará a determinar su posición de la tabla a través de una función **H(x)**, se puede dar que un objeto ocupe una posición donde ya exista otro objeto, a esto le llamaremos **Colisiones**, por lo que esto sería una amenaza para nuestro problema, ya que sería muy complejo y nos tomaría más tiempo encontrar o buscar jugadores dependiendo de sus destrezas (nivel de juego).

Por otro lado, pensemos que tenemos una cierta cantidad n-jugadores, y una tabla de m-posiciones, donde **n > m**. Esto nos presentaría una amenaza y debilidad porque nos quedarían jugadores por fuera de nuestra tabla hash.

Ahora bien, los inconvenientes que se pueden presentar (mencionados anteriormente) pueden ser solucionados por nuestros puntos positivos. Para la función Hash, se presentan varias oportunidades como lo son funciones cuadráticas o lineales, esto quiere decir que podremos evitar las colisiones en cuanto a la inserción en la tabla Hash. Por otro lado, el tamaño de la tabla se puede solucionar conociendo nuestro universo, para aún así, hacer más específico y eficiente nuestra solución.

Para nuestra última idea se tomó como base la nueva arma.

|  |  |  |
| --- | --- | --- |
|  | Factores internos | Factores externos |
| Puntos positivos | Fortalezas   * Novedosa * Gratuita | Oportunidades   * Ingreso de más usuarios * Críticas constructivas. |
| Puntos negativos | Debilidades   * Difícil de encontrar. | Amenazas   * Inconformidad de los usuarios |

*tabla 1.2*

La **tabla 1.2** nos genera la evaluación de la novedosa arma para de la celebración del día de amor y amistad, esto es algo importante porque es una funcionalidad de juego, lo más notorio para los usuarios. Una fortaleza de tener una nueva arma, es la cualidad de llamar la atención de nuestros usuarios y el índice de jugabilidad e ingreso de otras personas aumenta, esto quiere decir que nos va generar la oportunidad de poder leer o escuchar lo que piensan los usuarios , para así poder acomodar o satisfacer lo que se demanda. Por lo dicho anteriormente, esto sería una buena alternativa para solucionar los puntos negativos que se nos puedan presentar según nuestro FODA.

Para resumir, todas las alternativas seleccionadas para la resolución de nuestro problema tienen puntos negativos, sin embargo, estos puntos pueden repararse gracias a las fuerzas

y oportunidades de cada alternativa.

**Fase 5. Evaluación y Selección de la mejor opción**

En la evaluación vamos a tener en cuenta varios criterios:

**Criterio A:** Es dinámico

[2] Dinámico

[1]Fijo

Este criterio se evalúa si la alternativa presenta un comportamiento activo o cambiante para el aplicativo.

Se califica 1 si no varía dentro del programa.

Se califica 2 si varía dentro del programa.

**Criterio B:** Suple con la solución planteada en el problema.

[3] Sí

[2] Suple pero no del todo.

[1] No

Este segundo criterio, evalúa si la alternativa satisface totalmente el requerimiento que se nos presentó en el problema.

Se califica 3, si lo satisface totalmente.

Se califica 2, si lo satisface de vez en cuando.

Se califica 1, si no lo satisface.

**Criterio C**: Mejora el rendimiento para el aplicativo

[2] Sí

[1] No

Para este tercer criterio, se evalúa si la alternativa aumenta el rendimiento del juego.

Se califica 2, si en realidad sirve para mejorar la jugabilidad y la ejecución del juego.

Se califica 1. si no mejora en lo absoluto.

**Criterio D:**  Los usuarios notarán los cambios dentro del juego

[3] Bastante

[2] Poco

[1] Nada

Para este último criterio, se evalúa si las alternativa presentan cambios notables a la vista de los usuarios. En pocas palabras, si la actualización los usuarios la van a notar gráficamente.

Se califica 3, si se va a notar bastante.

Se califica 2, si se notará muy poco.

Se califica 1, si no se notará.

***Evaluación:***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Métodos / Criterios | **Criterio A** | **Criterio B** | **Criterio C** | **Criterio D** | **Total** |
| Tabla Hash | 1 | 3 | 2 | 2 | 8 |
| Cola | 2 | 3 | 2 | 2 | 9 |
| Nueva Arma | 2 | 2 | 1 | 3 | 8 |

*tabla 2*

En la **tabla 2** se puede ver claramente una calificación objetiva para cada una de las alternativas seleccionadas, por lo que empezaremos analizar cada una de estas dependiendo de su puntaje.

En primer lugar, la tabla Hash procede a tener un puntaje total de 8, para el criterio A, se puede deducir que por ser una tabla que se le asigna un valor fijo en cuanto al tamaño, no varía dentro del aplicativo. Esto quiere decir que no va variar con respecto a variaciones de corto plazo del juego. Para el criterio B y C, se tomó la opción sí porque cumple con sus objetivos para la solución del problema; además de que lo optimizará porque tiene una ventaja de eficiencia temporal de O(1). Para el último criterio, en cuanto gráficamente no lo notarán mucho los usuarios, pero sí en cuanto la fluidez del juego, ya que es un proceso que se lleva interno en el código del juego.

En segundo lugar, la cola obtuvo un puntaje total de 9 puntos, en consideración con la tabla Hash, en la cola sucede algo parecido, solamente varía en el criterio A, por el simple hecho de que la Cola contiene un sistema de tamaño dinámico, esto quiere decir que no se le da un tamaño fijo desde un comienzo, por esto puede variar en un corto plazo dentro del juego.

Por último, la nueva arma obtuvo un puntaje total de 8, en cuanto al primer criterio es una alternativa dinámica, ya que agiliza la calidad de las partidas, además de que puede cambiar en cada partida (frente a posición, jugador,etc). Para el criterio B, esto si es un caso común, por ser un día especial un arma no trata de cumplir todas las expectativas de los usuarios, pero eso depende ya que no sabemos cómo lo tomaron, será después de recibir las críticas. Para el criterio C, por ser un añadimiento de arma, generalmente no optimiza el juego, solamente es una nueva “función”, y para el último criterio D, aquí sí totalmente los usuarios podrán visualizar la arma y usarla dentro de una partida.

**6. Preparación de Informes y Especificaciones**

En esta fase se llevará a cabo la parte del diseño de nuestro método de ingeniería para la solución del problema.

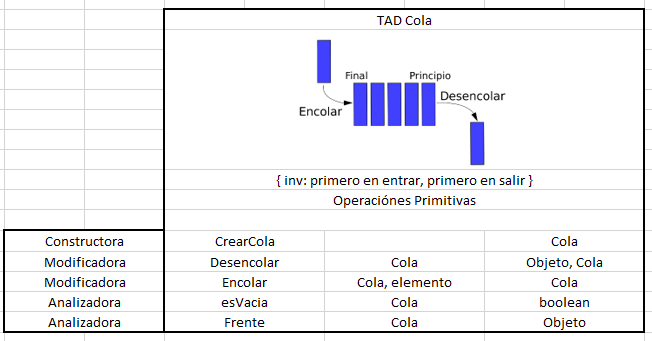
En primer lugar, se presentará los Tipos de Datos Abstractos de nuestras estructuras de datos que se implementarán en la codificación, para así llevar una comprensión detallada de las operaciones que nos ofrece cada una de estas.

En segundo lugar, también se incluirá el diagrama de clases respectivo para la descripción de nuestra codificación de manera que cualquier programador o persona relacionada con el tema pueda entender las clases, los atributos, las operaciones(métodos) y las relaciones entre objetos.

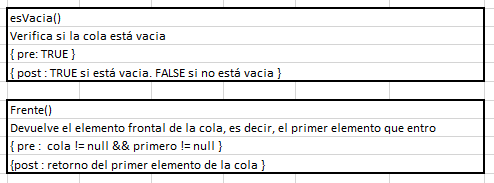
Por último, para comprobar y verificar el comportamiento de los métodos o las salidas que esperamos en cada operación, se usará el diseño de casos de pruebas unitarias, para detallar las entradas, y las salidas que se deben dar para un buen funcionamiento del aplicativo.

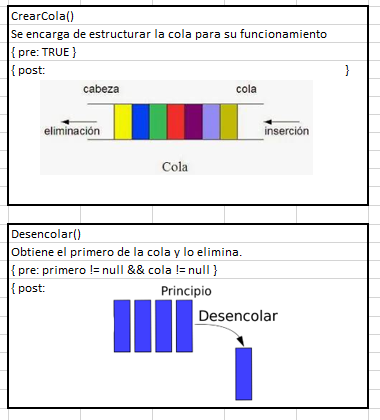
**TADs**

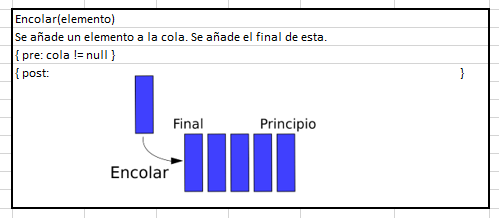
**COLA**

****

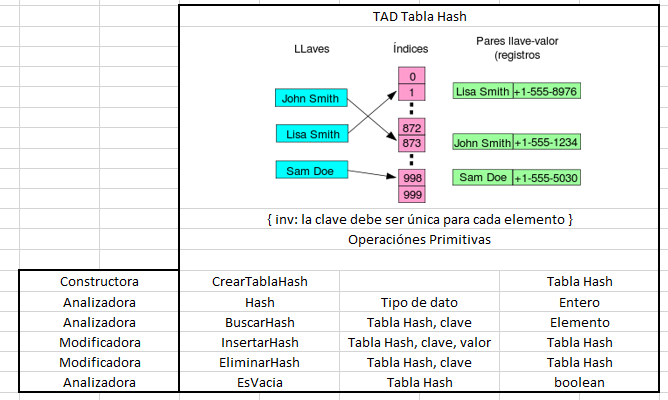
**OPERACIONES:**

****

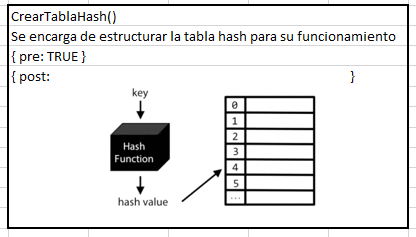
****

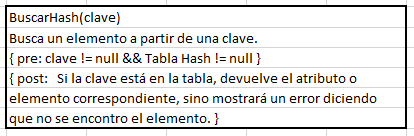


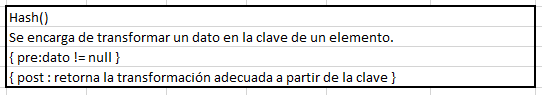
**TABLA HASH**

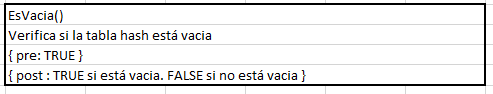
****

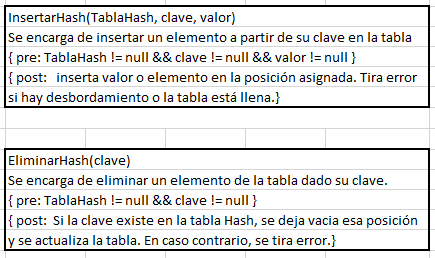
**OPERACIONES:**



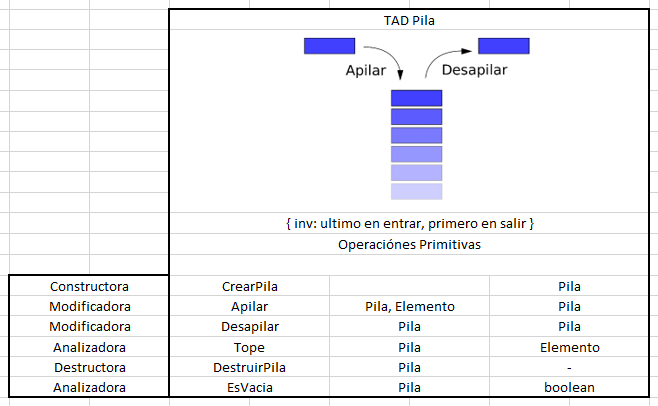




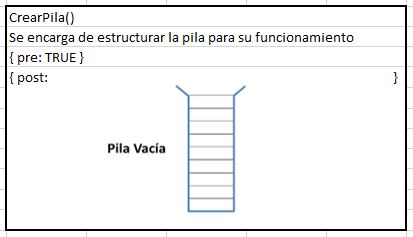
****

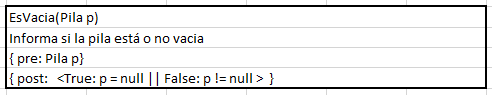
****

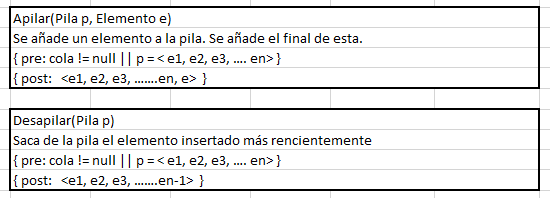
**PILA**

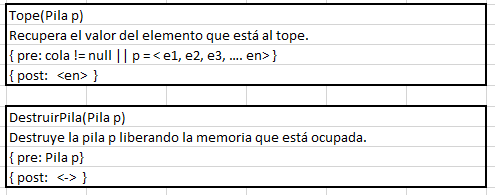
****

**OPERACIONES:**

****

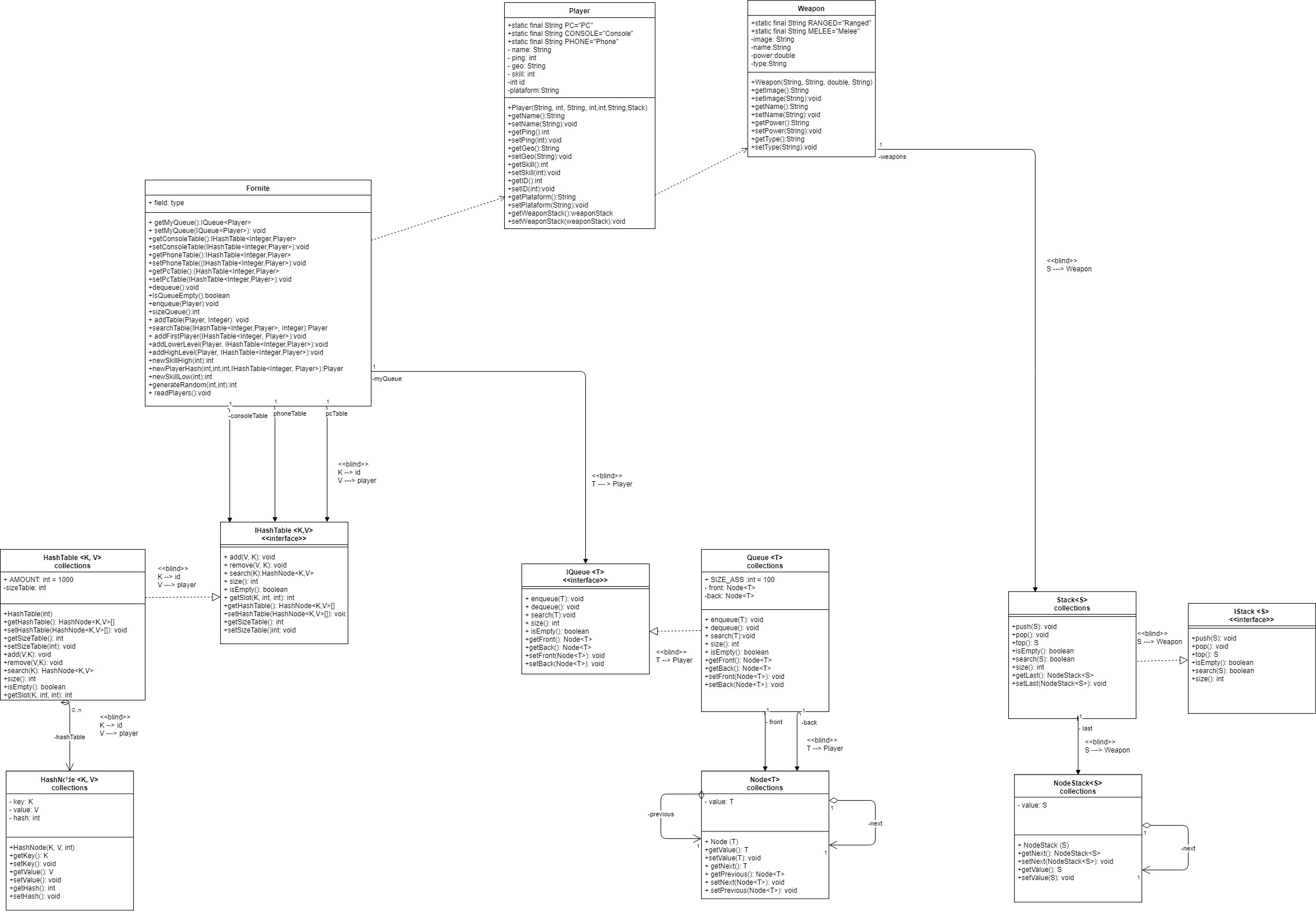
****

****

****

**Diseño de pruebas unitarias.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores** | **Resultado** |
| **FortniteTest** | enqueueTest() | Se instancian dos objetos de Player y un objeto de Fornite | Se agregan los dos player a la cola. | Cola != null  Se agregaron dos players a la cola. |
| **ForniteTest** | dequeueTest() | Se instancian dos objetos de Player y un objeto de Fornite | Se agregan los dos player a la cola. | Se ha eliminado el primer player en entrar. |
| **ForniteTest** | sizeQueueTest() | Se instancian dos objetos de Player y un objeto de Fornite | Se agregaron dos player a la cola | El tamaño de la cola es de 2. |
| **ForniteTest** | addTableTest() | Se instancian dos objetos de Player y un objeto de Fornite | Se agregan dos players a la tabla hash con su debida clave(ID). | Se han agregado los players a la tabla satisfactoriamente. |
| **ForniteTest** | searchTableTest() | Se instancian dos objetos de Player y un objeto de Fornite | Se agregan los players a la tabla hash. Después se pasa el nombre de uno de los players dentro de la tabla. | Se ha encontrado player satisfactoriamente. |
| **StackTest** | testPush() | Se instancio un objeto genérico de String de la clase Stack. Se instancio 3 Strings. | Se agregaron los 3 string a la pila. | Se han agregado los 3 Strings satisfactoriamente. |
| **StackTest** | testPop() | Se instancio un objeto genérico de String de la clase Stack. Se instancio 3 Strings. | Se agregaron los 3 Strings a la pila. | Se ha eliminado el objeto más recientemente ingresado. |
| **StackTest** | testTop() | Se instancio un objeto genérico de String de la clase Stack. Se instancio 3 Strings. | Se agregaron los 3 Strings a la pila. | Devuelve el último objeto añadido a la Pila. |
| **StackTest** | tesIsEmpty() | Se instancio un objeto genérico de String de la clase Stack. Se instancio 3 Strings. | Se agrego los 3 Strings a la pila. | false |
| **StackTest** | testSize() | Se instancio un objeto genérico de String de la clase Stack. Se instancio 3 Strings. | Se agrego los 3 Strings a la pila | 3 |
| **StackTest** | testSearch() | Se instancio un objeto genérico de String de la clase Stack. Se instancio 3 Strings. | Se agrego los 3 String a la pila.  Se busca la palabra “Hola”. | true |

****

Para ver mejor el diagrama, está subido en GitHub.

**Bibliografía**

<https://es.slideshare.net/YorkaMarisolPerezFeliz/listaspilas-y-colas-estructura-de-datos>

<http://artemisa.unicauca.edu.co/~nediaz/EDDI/cap07.pdf>

<https://www.infor.uva.es/~mserrano/EDI/cap6.pdf>

<https://esports.xataka.com/fortnite/fortnite-celebra-san-valentin-con-una-ballesta-con-municion-infinita>

<https://es.wikipedia.org/wiki/Lista_doblemente_enlazada#lista_doblemente_enlazada>

<https://es.wikipedia.org/wiki/Cola_de_prioridades#Caracter%C3%ADsticas_generales>

<http://www.ciberaula.com/articulo/colas_en_java>

<https://gamerant.com/fortnite-skill-matchmaking-rank/>