**ANÁLISIS**

**Descripción del contexto:**La empresa Epic Games está preocupada por la gran cantidad de recomendaciones que han dejado muchos de sus jugadores en el último par de meses, esta preocupación radica en algunos inconvenientes que los usuarios han sentido con el gameplay de sus partidas.

**Identificación y definición concreta del problema:**

Los usuarios de Fornite requieren mejoras en algunas funcionalidades del juego.

**Recomendaciones de los usuarios:**

* Los usuarios requieren encontrar partidas donde se enfrenten con jugadores que tengan un nivel de juego similar.
* La empresa debe diseñar un ranking en donde se categorice a los jugadores, según su destreza para el juego.
* Los usuarios requieren poder realizar partidas en modo “plataforma”, en la cual se limitan los jugadores que se unirán a la partida, de acuerdo a la plataforma que se esté utilizando.
* Los usuarios piden como una variante de juego, poder celebrar ocasiones especiales.
* La empresa ha aprobado la versión para el día de San Valentín, un modo de juego en donde sólo se puede utilizar la última arma que se levantó.

1. **Recopilación de Información**

**Fornite:**Es un videojuego desarrollado por las empresas Epic Games y People Can Fly que, a pesar de ser anunciado en 2011, no fue lanzado hasta el 25 de julio de 2017. Se trata de un juego de zombis en el que supervivientes controlados por humanos cooperarán online para mantener sus fortalezas a salvo de los ataques en oleadas de los muertos vivientes, que están intentando conquistar la tierra.  (Estructuras de Datos, 2018)

**San Valentín:**Es una festividad de origen cristiana que se celebra anualmente el 14 de febrero como conmemoración a las buenas obras realizada por san Valentín de Roma que están relacionadas con el concepto universal del amor y la afectividad. (Wikipedia, Wikipedia, 2018)

**Ranking:**Es una relación entre un conjunto de elementos tales que, para uno o varios criterios, el primero de ellos presenta un valor superior al segundo, este a su vez mayor que el tercero y así sucesivamente, permitiéndose que dos o más elementos diferentes puedan tener la misma posición. El orden se refleja asignando a cada elemento un ordinal, generalmente números enteros positivos.  (Wikipedia, Wikipedia, 2018)

**Desviación estándar:**Es una medida de dispersión para variables de razón (variables cuantitativas o cantidades racionales) y de intervalo. Se define como la raíz cuadrada de la varianza de la variable.

Su función es:

$ s =\sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i-\bar{x})^2$

N es el tamaño y x-bar la media aritmética de la muestra

Fuente del texto:  (Wikipedia, Wikipedia, 2018)

Fuente de la imagen:  (Arcidiacono, 2009)

**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 por medio del envío de paquetes ICMP de solicitud (ICMP Echo Request) y de respuesta (ICMP Echo Reply). Mediante esta utilidad puede diagnosticarse el estado, velocidad y calidad de una red determinada.

Fuente: (Wikipedia, 2018)

**Plataforma:** Se refiere al sistema operativo o a sistemas complejos que a su vez sirven para crear programas, como las plataformas de desarrollo.

Fuente: (Wikipedia, Wikipedia, 2017)

1. **Especificación de requerimientos funcionales**

|  |  |
| --- | --- |
| **Nombre** | **Req 001 - Rankear jugadores** |
| **Resumen** | Permite a los usuarios encontrar partidas en donde se enfrenten con jugadores que tengan un nivel de juego similar. Los jugadores deben ser categorizados según su destreza en el juego. |
| **Entrada** | Nivel de juego |
| **Salida** | Los usuarios han sido clasificados. |

|  |  |
| --- | --- |
| **Nombre** | **Req 002 - Modo plataforma** |
| **Resumen** | Permite limitar los jugadores que se unirán a la partida de acuerdo a la plataforma que estén utilizando. |
| **Entrada** | Tipo de plataforma de juego. |
| **Salida** | El juego comienza con usuarios de plataformas similares. |

|  |  |
| --- | --- |
| **Nombre** | **Req 003 - Modo San Valentín** |
| **Resumen** | Permite utilizar al usuario únicamente la última arma levantada. No existe un límite de armas para llevar y solamente se puede cambiar de arma encontrando una nueva o agotando las municiones de la actual. Si las municiones del arma actual se agotan, el arma actual desaparece y el usuario se equipará con la arma levantada antes que esa. Si las armas se agotan, el usuario tendrá por defecto un hacha, la cual es el arma que se da inicialmente. |
| **Entrada** | Booleano: tiene municiones o hay una nueva arma. |
| **Salida** | Modo de juego San Valentín está en curso. |

|  |  |
| --- | --- |
| **Nombre** | **Req 004 - Ping** |
| **Resumen** | Permite que la desviación estándar del ping de los jugadores de una partida sea el mínimo posible (latencia cercana). |
| **Entrada** | Ping de los jugadores |
| **Salida** | Los jugadores han sido clasificados según su ping. |

1. **Búsqueda de soluciones creativas**

Alternativa 1: Listas enlazadas.

La lista enlazada es un TDA que nos permite almacenar datos de una forma organizada, al igual que los vectores pero, a diferencia de estos, esta estructura es dinámica, por lo que no tenemos que saber "a priori" los elementos que puede contener.

En una lista enlazada, cada elemento apunta al siguiente excepto el último que no tiene sucesor y el valor del enlace es null. Por ello los elementos son registros que contienen el dato a almacenar y un enlace al siguiente elemento. Los elementos de una lista, suelen recibir también el nombre de nodos de la lista.

Alternativa 2: ArrayList.

La clase ArrayList en Java, es una clase que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el número de elementos que almacena, lo hace de forma dinámica, es decir, que no es necesario declarar su tamaño. Además permite modificar, insertar y eliminar elementos en cualquier lugar del arreglo.

Alternativa 3: Pila.

Una pila es una lista ordenada o [estructura de datos](https://es.wikipedia.org/wiki/Estructura_de_datos) que permite almacenar y recuperar datos, el modo de acceso a sus elementos es de modo que el primer elemento en entrar es el primer elemento en salir ([LIFO](https://es.wikipedia.org/wiki/LIFO)). Esta estructura se aplica en multitud de supuestos en el área de [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica) debido a su simplicidad y capacidad de dar respuesta a numerosos procesos. En ésta estructura de datos se puede apilar o des apilar elementos, es decir modificar únicamente el extremo superior de la estructura.

Alternativa 4: Cola.

Una cola (también llamada fila) es una [estructura de datos](https://es.wikipedia.org/wiki/Estructura_de_datos), caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pop por el otro. También se le llama estructura [FIFO](https://es.wikipedia.org/wiki/FIFO) (del inglés First In First Out), debido a que el primer elemento en entrar será también el primero en salir.

Las colas se utilizan en sistemas [informáticos](https://es.wikipedia.org/wiki/Inform%C3%A1tica), [transportes](https://es.wikipedia.org/wiki/Transporte) y operaciones de [investigación](https://es.wikipedia.org/wiki/Investigaci%C3%B3n) (entre otros), donde los objetos, personas o eventos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento. Este tipo de estructura de datos abstracta se implementa en [lenguajes orientados a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos) mediante clases, en forma de listas enlazadas.

Alternativa 5: Hash table.

es una [estructura de datos](https://es.wikipedia.org/wiki/Estructura_de_datos) que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos almacenados a partir de una clave generada. Funciona transformando la clave con una [función hash](https://es.wikipedia.org/wiki/Funci%C3%B3n_hash) en un [hash](https://es.wikipedia.org/wiki/Hash), un número que identifica la posición donde la tabla hash localiza el valor deseado.

Las tablas hash se suelen implementar sobre [vectores](https://es.wikipedia.org/wiki/Vector_(programaci%C3%B3n)) de una dimensión, aunque se pueden hacer implementaciones multi-dimensionales basadas en varias claves. Como en el caso de los arrays, las tablas hash proveen tiempo constante de búsqueda promedio [O(1)](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica),[1](https://es.wikipedia.org/wiki/Tabla_hash#cite_note-1)​ sin importar el número de elementos en la tabla. Sin embargo, en casos particularmente malos el tiempo de búsqueda puede llegar a O(n), es decir, en función del número de elementos.

Las anteriores estructuras de datos serán utilizadas en este caso para:

* Almacenar a los jugadores con sus respectivos puntajes para encontrar a los contrincantes con niveles de juego parecidos.
* Mostrar el ranking de los jugadores y así, poder acceder a ellos de forma rápida y más efectiva.
* Manejar los jugadores con el tipo de plataforma de juego para que los usuarios conozcan a sus contrincantes o compañeros de juego.

1. **Transición de las ideas a los diseños preliminares**

Alternativa 1:

* Tamaño dinámico que implica una mejor optimización de la memoria.
* Los elementos se pueden insertar en una lista indefinidamente.
* las listas enlazadas sólo permiten acceso secuencial a los elementos.
* Las listas enlazadas simples, solo pueden ser recorridas en una dirección.
* Las listas simples requieren la dirección del nodo anterior para insertar o suprimir correctamente.

Alternativa 2:

* Un ArrayList tiene un tamaño dinámico.
* el ArrayList es unidimensional.
* un ArrayList puede cambiar de tamaño según se necesite en tiempo de ejecución.
* admite elementos duplicados y nulos, así como acceso por índice y otros métodos para trabajar con sus elementos.

Alternativa 3:

* En cada momento sólo se tiene acceso a la parte superior de la pila.
* Introduce un elemento en el extremo de la pila.
* Elimina un elemento de la pila (superior extremo).
* Indica si la pila está o no vacía.
* Permite realizar apilamiento, desapilamiento y verificación de vacío.

Alternativa 4:

* Los elementos salen en el mismo orden en que llegan.
* Almacena al final de la cola un elemento.
* Saca de la cola al elemento que se encuentra al frente.
* Se crea la cola vacía.

Alternativa 5:

* El acceso a los datos suele ser muy rápido.
* Debe tener una razón de ocupación no muy elevada.
* Función bien diseñada para evitar que hayan muchas colisiones.
* Necesidad de ampliar el espacio cuando los datos crecen mucho.
* Se suelen emplear listas para recorrer la totalidad de los elementos.

1. **Evaluación y selección de la mejor solución**

Criterios para la evaluación:

Criterio A: Complejidad (peor caso).

* [8] complejidad constante.
* [7] complejidad logarítmica.
* [6] complejidad radical.
* [5] complejidad lineal.
* [4] complejidad n(log n).
* [3] complejidad polinómica.
* [2] complejidad exponencial.
* [1] complejidad factorial.

Criterio B: Eficiencia en manejo de datos

* [2] Eficiente para muchos datos.
* [1] Eficiente para pocos datos.

Criterio C: Dificultad de implementación.

* [3]Facilidad de implementación.
* [2]Normalidad en la implementación.
* [1]Dificultad para la implementación.

Evaluando los algoritmos bajo los anteriores criterios obtenemos la siguiente tabla de calificaciones:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Criterio A | Criterio B | Criterio C | Total |
| Listas enlazadas | Inserción O(n^2) 2 | Se adquieren posiciones de [memoria](https://www.monografias.com/trabajos13/memor/memor.shtml) a medida que se necesitan y se liberan cuando ya no se requieren 2. | Se requiere recorrerlas. 2 | 6 |
| ArrayList | En promedio O(k) 8 | Se debe recorrerlos. 1 | En ocasiones se necesitan más de un solo recorrido. 1 | 10 |
| Pila | Todas las operaciones de la pila se pueden manejar en tiempo constante.  8. | No se deben recorrer todos los datos.  2 | Posee tres métodos de modificación; insertar, modificar y eliminar. 2 | 12 |
| Cola | Las operaciones pueden ser ejecutadas en O(1).  8 | Se pude acceder al elemento más antiguamente insertado.  1 | Posee por defecto tres operaciones naturales.  3 | 12 |
| Hash table | En el peor de los casos y utilizando una buena función: O(1). 8 | Se puede establecer y modificar el tamaño de la tabla. 1 | Facilidad de acceder a los datos.  2 | 11 |

Analizando cada uno de los casos, llegamos a la conclusión de que las alternativas 3,4 y 5 son las mejores a implementar, debido a su eficiencia algorítmica.

**ETAPA 3: DISEÑO**

1. **Definición de tipos abstractos de datos (TAD)**

|  |  |
| --- | --- |
| **TAD** | **Hashtable** |
| Imagen relacionada    (wiki, 2018) | |
| ***Invariantes***  El tamaño de la tabla hash está determinado por un valor ingresado por parámetro. | |
| ***Operaciones:***  Hashtable() 🡪 **Hashtable 🡪** **Constructor**  removeKey() 🡪 **Elemento 🡪 Hashtable**  put() 🡪 **Elemento, Elemento 🡪 Hashtable**  searchKey() 🡪 **Elemento🡪 boolean**  searchValue() 🡪 **Elemento 🡪 boolean**  isEmpty() 🡪 **Hashtable 🡪 boolean**  getKey() **🡪 Hashtable 🡪 Elemento**  setKey() **🡪 Elemento 🡪 Hashtable**  getValue() 🡪 **Hashtable 🡪 Elemento**  setValue() **🡪 Elemento 🡪 Hashtable**  getTamanio() **🡪 Hashtable 🡪 integer**  setTamanio() **🡪 Integer 🡪 Hashtable**  getArray() **🡪 Hashtable 🡪 Arreglo**  setArray() 🡪 **Arreglo –> Hashtable** | |
| * ***Pre:*** *Las variables están declaradas* * ***Post:*** *Las variables han sido instanciadas* * ***Def:*** *Constructor de la clase.*   ***public Hashtable()***   * ***Pre:*** *La llave está declarada e instanciada.* * ***Post:*** *la llave encontrada ha sido eliminada.* * ***Def:*** *El método se encarga de eliminar una llave.*   ***public void removeKey(Elemento k)***   * ***Pre:*** *La llave y el valor, han sido declarados e instanciados.* * ***Post:*** *El objeto ha sido agregado a la hashtable.* * ***Def:*** *El método se encarga de agregar un objeto a la tabla hash.*   ***public void put(Elemento k, Elemento v)***   * ***Pre:*** *la tabla hash tiene objetos.* * ***Post:*** *Se ha verificado si la llave está en la tabla hash.* * ***Def:*** *Se encarga de verificar si una llave se encuentra dentro de la tabla hash.*   ***public boolean searchKey(Elemento k)***   * ***Pre:*** *la tabla hash tiene objetos.* * ***Post:*** *Se ha verificado si el valor está en la tabla hash.* * ***Def:*** *Se encarga de verificar si una valor se encuentra dentro de la tabla hash.*   ***public boolean searchValue(Elemento v)***   * ***Pre:*** *La tabla hash ha sido creada, vacía.* * ***Post:*** *Se ha verificado si la tabla hash no tienen ningún componente.* * ***Def:*** *Verifica si la tabla hash no tiene elementos.*   ***public boolean isEmpty()*** | |

|  |  |
| --- | --- |
| **TAD** | **Pila** |
| Imagen relacionada  (MSRUAZ Coding & Programming Blog, 2014) | |
| ***Invariantes***  *El último objeto en ingresar es el primero en salir. Estructura LIFO.(Last In First Out).* | |
| ***Operaciones:***  *insertWeaponStack()* ***🡪 Weapon 🡪 Player***  *getTop()* ***🡪 Player 🡪 Weapon***  *longitude()* ***🡪 Player 🡪 Integer***  *isEmpty()* ***🡪 Player 🡪 boolean***  *search()* ***🡪 Weapon 🡪 boolean***  *remove()* ***🡪 Weapon 🡪 Player*** | |
| * ***Pre:*** *La pila ha sido inicializada.* * ***Post:*** *Se ha agregado un elemento a la pila* * ***Def:*** *El método se encarga de insertar un elemento en la pila, el cuál estará en el tope (cima).*   ***public void insertWeaponStack(Weapon w)***   * ***Pre:*** *la pila ha sido inicializada, el primer elemento siempre es un arma de tipo hacha.* * ***Post:*** *Se ha retornado el elemento top.* * ***Def:*** *El método retorna el elemento que se encuentra en la cima de la pila. El elemento top.*   ***public Weapon getTop()***   * ***Pre:*** *La pila tiene un hacha como primer elemento.* * ***Post:*** *Se ha retornado el número de elementos de la pila.* * ***Def:*** *El método retorna el tamaño de la pila, el cual corresponde al número de elementos contenidos.*   ***public int longitude()***   * ***Pre:*** *La pila tiene un hacha como primer elemento.* * ***Post:*** *Se ha verificado que la pila tiene por lo menos un elemento.* * ***Def:*** *El método retorna un booleano que determina si la pila contiene elementos.*   ***public boolean isEmpty()***   * ***Pre:*** *La pila tiene un hacha como primer elemento.* * ***Post:*** *Se ha verificado que el elemento buscado está en la pila.* * ***Def:*** *El método se encarga de buscar si existe un elemento igual al ingresado por parámetro en la pila.*   ***public boolean Search(Weapon w)*** | |

|  |  |
| --- | --- |
| **TAD** | **Cola Jugador** |
| Resultado de imagen para jugador fortnite  (Steemit beta, 2018) | |
| ***Invariantes***  *Estructura FIFO(Firts In First Out), los jugadores se organizan según el orden en que han sido agregados a la cola.* | |
| ***Operaciones:***  *Ranking\_Players() 🡪* ***Constructor***  *LevelUp()*  *Longitude()*  *isEmpty()*  *insertStack(Player)* | |
| * ***Pre:*** *Las variables han sido inicializadas.* * ***Post:*** *Las variables han sido instanciadas.* * ***Def:*** *Constructor de la clase*   ***public Ranking ()***   * ***Pre:*** * ***Post:*** * ***Def:***   ***Public int longitude()***   * ***Pre:*** * ***Post:*** * ***Def:***   ***Public boolean isEmpty()***   * ***Pre:*** * ***Post:*** * ***Def:***   ***Public Player search(String s)***   * ***Pre:*** * ***Post:*** * ***Def:***   ***Public boolean remove(String)***   * ***Pre:*** * ***Post:*** * ***Def:***   ***Public void levelUp()*** | |

1. **Diseño de casos de pruebas unitarias**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DISEÑO DE CASOS PARA PRUEBAS UNITARIAS | | | | | |
| Entrada | **Salida** | **Descripción** | **Nombre del método** | **Clase** | **Escenario** |
| “Lanzallamas” | True | El método retorna true, si la última arma ha sido retornada, de lo contrario, retornará false. | insertWeaponStack(Weapon) | Player | SceneOne() |
| “Ak-47” | False |
| 7 | True | El método retorna true, si la cantidad de jugadores ingresados en la cola coinciden, retorna false en caso contrario | tamanio() | Ranking\_Players | SceneTwo() |
| 6 | False |
| 2 | True | El método retorna true si los jugadores han sido rankeados, retorna false si no lo son | QueueUp(player) | Ranking\_Players | SceneThree() |
| 0 | False |
| 13,1 | True | El método retorna true si la desviación estándar del ping de los jugadores de una partida es el mínimo posible, de lo contrario retorna false. | Ping() | Player | SceneFour() |
| 0 | False |

# **Bibliografía**

Arcidiacono, G. (2009). Obtenido de http://www.alcula.com/es/calculadoras/estadistica/desviacion-estandar/

Estructuras de Datos. (Agosto de 2018). *Universidad Icesi*. Obtenido de https://docs.google.com/document/d/1bqwF-1Vk-vd4-JlrMB7eY\_AThAMdl0taRiEHogiHaAA/edit

MSRUAZ Coding & Programming Blog. (2014). *MSRUAZ Coding & Programming Blog*. Obtenido de MSRUAZ Coding & Programming Blog: http://static.wixstatic.com/media/766985\_b664b6f86763445d875a7c768dae2949.jpg

POETS&QUANTS, INC. ALL RIGHTS RESERVED. (2018). *POETS&QUANTS*. Obtenido de POETS&QUANTS: https://poetsandquants.com/wp-content/uploads/2017/11/Rankingillo.jpeg

Steemit beta. (2018). *Steemit beta*. Obtenido de Steemit beta: https://www.google.com.co/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjP29m6mazdAhWjzVkKHeV3B6UQjRx6BAgBEAU&url=https%3A%2F%2Fsteemit.com%2Fspanish%2F%40gerardoalfred%2Ffortnite-el-juego-del-momento&psig=AOvVaw1Bw49lUnhUwHpvyLYRTi

Vandal. (2018). *Vandal*. Obtenido de Vandal: https://media.vandal.net/i/600x300/2-2018/201822211756\_1.jpg

Wikipedia. (17 de diciembre de 2017). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Plataforma

Wikipedia. (19 de julio de 2018). Obtenido de https://es.wikipedia.org/wiki/Ping

Wikipedia. (29 de julio de 2018). *Wikipedia*. Obtenido de Wikipedia.

Wikipedia. (27 de abril de 2018). *Wikipedia*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Ranking

Wikipedia. (27 de agosto de 2018). *Wikipedia*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Desviaci%C3%B3n\_t%C3%ADpica