Método de la ingeniería - Fortnite Lab

1. **Identificación del problema:**

    a. Los usuarios presentan dificultades para encontrar partidas con otros usuarios que tengan un nivel de juego parecido.
    b. Posible aumento de tiempo promedio de espera después de implementar el sistema de ranking.
    c. Posible variación en la latencia de los jugadores, ya que al emplear búsqueda de partidas por ping y geolocalización, si la latencia varía mucho aumenta el lag en las partidas.
    d. Creación de un modo “plataforma” donde los usuarios puedan jugar con otros usuarios que jueguen desde la misma plataforma
    e. Creación de un modo de San Valentín que logre llamar la atención de las personas encargadas.

Definición del problema:
Epic Games requiere que se implemente un sistema de ranking donde se pueda categorizar los jugadores según su destreza de juego. También se requiere que la latencia entre jugadores sea cercana para evitar el lag y además, se requiere implementar dos modos de juego: un modo de juego de “San Valentín” y otro modo de juego “Plataforma” donde cada plataforma tiene sus partidas exclusivas (Nintendo, PC, PlayStation).

Requerimientos funcionales

Nombre	RF.#1 Clasificar jugador.
Resumen	El software clasificar a los jugador con base en su nivel de habilidad, ping y consola.
Entradas	
Resultados	Jugadores clasificados en una estructura de datos.

Nombre	RF.#2 Buscar partida.
Resumen	Al buscar partida el software debe emparejar a jugadores que tengan una latencia cercana

Entradas	Nombre de jugador, ping.
Resultados	Partida de jugadores con ping cercanos.

Nombre	RF.#3 Buscar partida en plataforma.
Resumen	El software debe permitir la opción de buscar partidas para jugadores de una misma plataforma, emparejando con jugadores de nivel de habilidad y ping cercanos.
Entradas	Nombre jugador, plataforma.
Resultados	Partida de jugadores con la misma plataforma y ping cercanos.

Nombre	RF.#4 Proponer implementación del nuevo modo de juego.
Resumen	Generar una implementación para el modo de juego de San Valentín teniendo en cuenta la última arma que el jugador levantó
Entradas	
Resultados	Modo de juego

2. Recopilación de la información

¿Qué es un ranking?
Es un sistema de relación entre los elementos de un conjunto donde el primero presenta un valor superior al segundo en algún criterio. El ranking es un sistema de clasificación ordinal.

¿Qué es latencia?
La latencia es la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

¿Qué es 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.

¿Qué es geolocalización?
La geolocalización es la capacidad para obtener la ubicación geográfica real de un objeto, como un radar, un teléfono móvil o un ordenador conectado a Internet. La geolocalización puede referirse a la consulta de la ubicación, o bien para la consulta real de la ubicación.

¿Qué es lag?
Es un retardo excesivo producido por una telecomunicación en tiempo real, puede darse por una latencia alta en la red o por el ordenador.

3. Búsqueda de soluciones creativas

Alternativa 1:
Para crear una partida mediante emparejamiento por ranking y latencia, se emparejan los jugadores que dan como resultado la menor desviación estándar tanto de ranking como de latencia. En este caso, el ranking se basaría en una comparación por Kill/Death Ratio (K/D Ratio) y la latencia se basaría en una comparación de ping.

Alternativa 2:
Otra alternativa para crear una partida mediante emparejamiento por ranking y latencia, es que se emparejen los jugadores que den como resultado la menor desviación estándar tanto de ranking como de latencia. En este caso, el ranking se basaría en una comparación por Won Matches/Played Matches Ratio (Wins/Plays Ratio) y la latencia se basaría en una comparación de ping.

Alternativa 3:
Una nueva alternativa para crear una partida mediante emparejamiento por ranking y latencia, es que se emparejen los jugadores que den como resultado la menor desviación estándar tanto de ranking como de latencia. En este caso, el ranking se basaría en una comparación por Won Matches/Played Matches Ratio (Wins/Plays Ratio) y un Kill/Possible Kills Ratio (K/PK Ratio) a la vez. Para lo anterior se utilizaría una especie de puntaje, que se obtiene al sumar ambos Ratios, y se emparejan los jugadores cuyo puntaje esté entre un determinado intervalo. La latencia se basaría en una comparación de ping.

4. Transición de las ideas a los diseños preliminares

Alternativa 1:
- Se agrupan los jugadores cuyos valores den como resultado la menor desviación estándar de ranking y de ping.
- El jugador será evaluado por un K/D Ratio
- Evaluar los jugadores por K/D Ratio trae como desventaja que usualmente los jugadores con un alto valor en este criterio resultan siendo los mejores jugadores, lo que incrementa la posibilidad de que un jugador con un K/D normal o promedio sea eliminado rápidamente. Esto a su vez, disminuye el tiempo de duración de una partida, lo que no sería agradable para los jugadores.

- Si un jugador es eliminado rápidamente puede decir que no está jugando con jugadores de su mismo “nivel”, esto debido a lo expuesto en el punto anterior.
- Se agregan los jugadores a un arraylist.
- Hay una lista enlazada de partidas.

Alternativa 2:

- Se agrupan los jugadores cuyo resultado den como resultado la menor desviación estándar de ranking y de ping.
- El jugador será evaluado por un Wins/Plays Ratio.
- Evaluar a los jugadores por un Wins/Plays Ratio es más justo que evaluar por un K/D Ratio, ya que aquí todos tendrían posibilidades más altas para ganar debido a que sus valores del Ratio no van a variar mucho, es decir, son jugadores de similares niveles.
- La desventaja de evaluar a los jugadores por un Wins/Plays Ratio es que existe la posibilidad de que los jugadores con más alto valor en este criterio sean “campers” o “hackers/modders”. Es decir, se pueden filtrar jugadores que no hacen parte del enfrentamiento intenso en la partida sino que se esconden hasta que queden pocos jugadores y jugadores que utilizan hacks o mods para ganar partidas de manera sucia, pues esto los hace superiores a los demás jugadores.
- Si un jugador es eliminado por un hacker o camper, puede decir que no está jugando con jugadores que sigan el objetivo inicial de Fortnite (enfrentamiento campal intenso constante y sin trampas)
- Cada partida es un arreglo de tamaño fijo 100.
- Hay un arraylist de partidas.

Alternativa 3:

- Se agrupan los jugadores cuyo resultado den como resultado la menor desviación estándar de ranking y de ping.
- Se introduce el criterio de evaluación Skill, como resultado de la suma de los Ratios del jugador (W/P y K/PK).
- El jugador será evaluado por su Skill.
- Evaluar a los jugadores por el criterio anteriormente nombrado reduce de gran manera las partidas con hackers,modders y campers. Esto a su vez, aumenta considerablemente la probabilidad de que los jugadores que se encuentran en la partida tengan similar o igual nivel de juego o habilidad.
- Las partidas tienen más enfrentamientos constantemente, se juega a un nivel intenso y los jugadores tienen estadísticas similares, por lo que es un enfrentamiento parejo.
- Los jugadores que están buscando una partida son almacenados en una cola.
- Se distribuyen los jugadores en una tabla hash para facilitar su posterior comparación.
- Después los jugadores son comparados por su skill y ping, y se agregan a una lista enlazada que representa la partida, el tamaño máximo de la lista es 100, que es el número de jugadores por partida.
- La lista va a tener un rango de ping y skill que sería el rango en el que debería estar todos los jugadores que ingresan en ella. Es decir, si un jugador no está en los rangos de la partida, no se agrega a ella y se busca otra partida.

5. Evaluación y selección de la mejor solución

Criterios de evaluación: precisión de la solución, nivel de dificultad de programación de la solución, diferencia de nivel y/o habilidad entre los jugadores, experiencia de juego, cumplimiento de los requerimientos solicitados.

Precisión de la solución (Criterio A): este criterio se refiere a qué tan acertada es la solución para el problema inicial, es decir, si se soluciona el problema correctamente sin generar más problemas.

Nivel de dificultad de programación (Criterio B): este criterio se refiere a qué tan difícil es pasar de las ideas propuestas como alternativas en la búsqueda de soluciones creativas a la implementación de las mismas, es decir, al código.

Diferencia de nivel y/o habilidad entre los jugadores (Criterio C): este criterio se refiere a qué tanta diferencia hay entre los jugadores respecto a sus habilidades para jugar, por ejemplo, si no se emparejan jugadores demasiado experimentados con jugadores novatos.

Experiencia de juego (Criterio D): este criterio evalúa qué tan agradable o similar a la idea original del juego es la experiencia de juego. Es decir, si la partida es dinámica, intensa y con enfrentamientos continuos y habituales.

Cumplimiento de los requerimientos solicitados (Criterio E): este criterio evalúa si se cumplen o no los requerimientos inicialmente solicitados.

Escalas de los criterios de evaluación
Las posibles respuestas que pueden tomar los criterios son las siguientes:

Criterio A: Precisión.
- [2] Exacta
- [1] Aproximada
- [0] Deficiente

Criterio B: Dificultad de programación.
- [5] Muy difícil
- [4] Difícil
- [3] Regular
- [2] Fácil
- [1] Muy Fácil

Criterio C: Diferencia de nivel y/o habilidad entre jugadores.
- [3] Alta
- [2] Media
- [1] Baja

Criterio D: Experiencia de juego.
- [5] Muy agradable y competitiva
- [4] Agradable y competitiva

- [3] Normal
- [2] Mala
- [1] Muy Mala

Criterio E: Cumplimiento de requerimientos.
- [2] Cumple con todos.
- [1] Cumple con algunos.
- [0] No cumple los requerimientos.

Evaluación:

Alternativa	Criterio A	Criterio B	Criterio C	Criterio D	Criterio E
1	(Aproximada)1	(Regular) 3	(Alta) 3	(Mala) 2	(Cumple algunos) 1
2	(Aproximada)1	(Regular) 3	(Media) 2	(Mala) 2	(Cumple algunos) 1
3	(Exacta) 2	(Difícil) 4	(Baja) 1	(Muy agradable y competitiva) 5	(Cumple todos) 2

Después de analizar esta tabla de evaluación llegamos a la conclusión de que la mejor solución pertenece a la alternativa 3 y por consiguiente será la alternativa elegida para la implementación.

6. Preparación de informes y especificaciones.

Especificaciones de los problemas con base a la entrada y la salida.

Problema: calcular el “Skill” de los jugadores.

Entrada: número de bajas del jugador, cantidad de partidas ganadas y cantidad de partidas jugadas.

Posibles bajas = Número de partidas jugadas * Número de jugadores por partida (99).

K/PK = Número de bajas del jugador/ Número de posibles bajas.

Ws/Ps = Número de partidas ganadas/ Número de partidas jugadas.

Skill = K/PK + Ws/Ps.

Salida: se obtiene el skill, que se encuentra entre 0 y 2.

Diagrama de clases:
Para facilitar la comprensión de los diagramas de clases debido a su gran tamaño, se decidió ubicarlos en la carpeta del proyecto y no en el presente documento.

Diseño de casos de pruebas unitarias:

Prueba No.1	**Objetivo de la prueba:** Verifica que encuentre una partida con el ping requerido			
Clase para probar	**Método**	**Escenario**	**Valores de entrada**	**Resultado**
Fortnite	`searchMatch(int ping min)`	escenario1()	una partida m agregada en la lista de partidas	Que la partida buscada sea igual a la de la entrada

Prueba No.2	**Objetivo de la prueba:** Verificar que los jugadores sean iguales			
Clase para probar	**Método**	**Escenario**	**Valores de entrada**	**Resultado**
Fortnite	`CreatePlayer(String name, String pla)`	escenario1()	Un jugador	Que el jugador creado sea igual al de la entrada

Prueba No.3	**Objetivo de la prueba:** Verificar que los jugadores sean iguales			
Clase para probar	**Método**	**Escenario**	**Valores de entrada**	**Resultado**
Match	`addPlayerMatch(Player player)`	escenario1()	Un jugador	Que el jugador agregado a la partida sea igual al jugador de la

				entrada

Prueba No.3	**Objetivo de la prueba:** Verificar que compare y muestre quien tiene skills mayor			
Clase para probar	**Método**	**Escenario**	**Valores de entrada**	**Resultado**
Player	Compareto(Player p)	Escenario1()	Un jugador	Que el método retorne 1 si es mayor o -1 si el menor

7. Implementación del diseño

Enlace al repositorio de git: https://github.com/jfcastillo/FortniteLab.git

Referencias:

Ranking: https://es.wikipedia.org/wiki/Ranking

Latencia: https://es.wikipedia.org/wiki/Latencia

Ping: https://es.wikipedia.org/wiki/Ping

Lag: https://computerhoy.com/noticias/software/que-es-lag-60090

Geolocalización: https://es.wikipedia.org/wiki/Geolocalizaci%C3%B3n