# Identificación del problema y análisis de requerimientos

## Caso de Estudio: Calculo del daño Martin Parodi

|  |  |
| --- | --- |
| Cliente | Desarrolladora Hoyo Verse |
| Usuario | Jugador |
| Requerimientos funcionales | 1. El programa registrara las estadísticas de un personaje 2. El programa debe calcular el daño Row(Nota en jerga de videojuegos el daño Row se refiere al daño normal) 3. El programa debe almacenar un registro de los daños realizados 4. El programa debe ser capas de mostrar el daño más alto realizado |
| Contexto del problema | Dentro del juego de rol Genshin impact los jugadores son capases de aplicar diversos tipos de daño a los enemigos con una potencia X dependiendo de la formula como se calcula el daño entre los tipos (daño transformativo, daño amplificador y daño aditivo) y la resistencia del objetivo, el programa debe res capas de determinar tras aplicar los respectivos cálculos que tanto daño le hace el jugador al objetivo |
| Requerimientos no funcionales | 1. Identifica si el objetivo puede o no recibir daño 2. El programa debe ser capas de calcular el daño causado por un equipo de 2 a 4 personajes 3. Realizar el proceso en un tiempo lo mas corto posible |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *[RF1-Registro de Stads]* | | | |
| Resumen | *El programa registra las estadísticas de un personaje y las almacena para usarlas posteriormente en el cálculo de daño* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| ATK | double | | Formato de estadísticas estándar del juego |
| maestría elemental (ME) | double | | Formato de estadísticas estándar del juego |
| la probabilidad de daño crítico (PDC) | double | | Formato de estadísticas estándar del juego |
| **cantidad porcentual de daño crítico (DC)** | double | | Formato de estadísticas estándar del juego |
| Resultado o Postcondición | Se registrarán el ataque físico y ataque especial del personaje para realizar los cálculos | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| Registro de Stdas | | Void |  |

*RF1-*EJ: registro a Paimon

Sus estadísticas son

ATK: 150

(ME): 60

(PDC): 30%

**(DC):60%**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *[RF2-Daño Base Row]* | | | |
| Resumen | *El programa calculara el daño base o Raw dependiendo de si es un ataque físico o especial sin contar el bono con la formula*  Daño base = AtaquePersonaje \* (1 + PDC + DC) | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
|
| ATK | Double | | Ataque base del personaje sin modificar |
| Resultado o Postcondición | Se calculará el daño base de acuerdo a la formula anteriormente mencionada y se guardara para usarse mas tarde | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| Daño Base | | String | Mensaje: Daño base = (resultado de la operación) |

*RF2-*EJ: Daño base = Ataque Personaje \* (1 + PDC + DC)

285 = 150 \* (1 + 0.30+ 0.60)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *[RF3-cálculo de daño]* | | | |
| Resumen | *El programa calculara cuanto daño ara el personaje por cada tipo de daño especial y tipo de ataque(esto para los* ***Multiplicadores Transformativos****)* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| Daño clasificado | String | | *Que el daño se haya aplicado* |
| Formulas | String | | Formulas apropiadas para el cálculo (nota esto lo coloco aquí porque no savia en que otro lugar ponerlo ya que se necesitaba ingresarlo si o si) |
| Resultado o Postcondición | El programa aplicara la formula y calculara el daño para luego almacenarlo en el registro | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
|  | Daño calculado | | Doublé | Mensaje el daño realizado es (Daño calculado) |

RF3-EJ:

Los***Multiplicadores Transformativos*** son los siguientes: 0.25 para quemadura, 0.50 para superconductor, 0.6 para torbellino, 1.2 para electro cargado, 1.5 para cristalización, 2.0 para sobrecargado, 2.0 para florecimiento, 3.0 para sobre florecimiento y super quemadura.

Nivel actual de Paimon: 30

Multiplicador de resistencia generado al asar entre 0.5 y 2.0

Daño Transformativo = Multiplicador Transformativo\*Nivel personaje

  \*(1 + %BonoME) \*Multiplicador resistencia

%BonoME = 16 \* Maestría Elemental/Maestría Elemental + 2000

ATK: 150

(ME): 60

(PDC): 30%

**(DC):60%**

Ataque quemadura

**8,24=0.25\*30\*(1+(16\*60/60+2000))\*0.75**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *[RF4-registro de daños]* | | | |
| Resumen | El programa almacenara un registro de los últimos daños realizados y ser capaz de mostrarlos para que el jugador los conozca | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| Daño calculado | Double | | *Daño tras ser procesado y calculado* |
|  |  | |  |
| Resultado o Postcondición | El programa ira organizando los datos a medida que se ballan realizando yendo del mas antiguo al mas reciente y mostrando en ese orden los últimos 10 daños hechos | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| Lista de daños | | Arrai | Datos ordenados del más al menos reciente |

*RF4-*EJ:lista de daños

1. 8.00
2. 8,24
3. 7.77
4. 9.01
5. 10.00
6. 7.50
7. 8.30
8. 9.22
9. 7.24
10. 9.05

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *[RF5-daño Máximo]* | | | |
| Resumen | *El programa determinara cual es el mayor daño hecho de entre la lista de 10 daños mostrados* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| Lista de daños | Arrai | | Datos ordenados del más al menos reciente |
| Resultado o Postcondición | El programa determinara cual de esos daños es el mayor y lo resaltara | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| Daño mayor | | Double | El Daño mayor será señalado |

RF5-EJ:

1. 8.00
2. 8,24
3. 7.77
4. 9.01
5. **10.00**
6. 7.50
7. 8.30
8. 9.22
9. 7.24
10. 9.05