# Introducción

Este documento presenta y define la métrica escogida para medir el desarrollo del proyecto.

Se analizan las posibles métricas explicadas en clase explicando por qué se ha rechazado o se ha aceptado cada una.

# Índice

[1. Decisión de la métrica utilizada](#h.x2quj05lnjbz)

[1.1. Descarte de otras métricas](#h.svatozjh3t1y)

[1.2. Decisión de los puntos de función como métrica utilizada](#h.yo19gyy8v96b)

[2. Definición de los puntos de función utilizados](#h.w1cnjc7l2ey5)

[2.1.Módulos contemplados y división de los puntos de función](#h.q7lwpcen12vj)

[2.2 Definición de la progresión en las tareas](#h.7f39wkrjxxi0)

[3. Anexos](#h.i8hetwudl2nv)

# 

# 1. Decisión de la métrica utilizada

Se ha definido un sistema de medición basado en puntos de función como métrica al desarrollo del proyecto. El baremo utilizado; así como las razones por las que se han rechazado otras métricas se explican a lo largo del documento.

## 1.1. Descarte de otras métricas

Se ha rechazado utilizar el **número de líneas de código** por los siguientes motivos:

1. Considerar que no representan una medida fiable sobre el estado el proyecto: posibilidad de avanzar en el desarrollo del mismo reduciendo el número de líneas; por lo tanto no sirve como métrica del desarrollo
2. Métrica fuertemente dependiente de factores ajenos como pueden ser el lenguaje utilizado (incluso de la versión del lenguaje) o el estilo de programación de los desarrolladores.
3. Posibilidad de crear una tendencia de escribir líneas “inútiles” o “superfluas” con el fin de engordar los resultados de cara a una revisión del proyecto.

Sin embargo, para poder comparar el proyecto con el resto de grupos de la asignatura, también se han definido las líneas de código previstas. Para cada cifra se ha recurrido a la comparación con proyectos similares para intentar acertar lo más posible en nuestra previsión ([anexo1](#id.l3sidv1r4f6q)).

Igualmente, se ha rechazado la **actividad en servidor remoto** como métrica para el proyecto (frecuencia y número de *commits*) por lo siguientes motivos, similares a los anteriores:

1. No representa una medida fiable sobre el estado del proyecto: igual que para el número de líneas de código, un mayor número de commits no implica un avance en el desarrollo del proyecto al existir la posibilidad de tratarse de modificaciones a código ya existente.
2. Métrica dependiente del estilo de programación de los desarrolladores. No existe una definición para “cuándo deben hacerse los *commits*” y cada desarrollador es libre de marcarlos en el momento que se crea oportuno
3. Posibilidad de crear una tendencia a hacer *commits* “vanos” para intentar crear la ilusión de un mayor avance en el proyecto.

## 1.2. Decisión de los puntos de función como métrica utilizada

La métrica escogida son los **puntos de función** por los siguiente motivos:

1. Mide directamente la funcionalidad entregada al usuario; por lo tanto, un mayor número de puntos de función completados sí implicará un avance en las funcionalidades disponibles.
2. Son independientes de la tecnología utilizada para la construcción.
3. Capacidad de medir un avance en el desarrollo en cualquiera de las fases de vida del software (desde inicio hasta transición) de forma que podemos hablar de “puntos de función completados” antes de la implementación del sistema.
4. Métrica adaptable al proyecto; es decir, los puntos de función se pueden definir para las funcionalidades propias del proyecto.

# 2. Definición de los puntos de función utilizados

El proyecto consta de **350 puntos de función** que representan funcionalidad del sistema agrupada en varios módulos (aproximadamente 7 puntos de función / persona en un mes).  
Cada módulo lleva asociado un valor que representa el **peso del módulo** sobre el proyecto completo. A mayor peso, mayor funcionalidad asociada a ese módulo.   
Sobre este valor base, se aplica un modificador que representa la **dificultad prevista** para realizar el módulo. Este modificador varía entre uno y cinco; siendo uno “sin dificultad extra” y cinco “módulo con una gran dificultad prevista”.   
Los puntos de función totales para un módulo se calculan multiplicando el peso base por el modificador de dificultad. Siendo M los módulos totales:

## 2.1.Módulos contemplados y división de los puntos de función

Se ha dividido el proyecto en 20 módulos que agrupan funcionalidad del sistema. Qué agrupa cada módulo, así como cuántos puntos de función totales (incluyendo peso y dificultad) ocupa cada módulo se detalla a continuación.

El desglose completo de los puntos de función y gráficos de la división del proyecto se encuentra en el [anexo2](#id.skd8rtr5uec9)

1. **Definición del juego**: engloba la parte de creación y definición de las reglas del juego que se va a implementar.  
   Puntos de función totales: **42**
2. **Diagramas y casos de uso**: mide el proceso de definición, discusión y pasos necesarios en el modelo de desarrollo escogido (guiado por casos de uso).  
   Puntos de función totales: **16**.
3. **Login y registro de usuarios**: engloba la implementación, validación y pruebas de seguridad para la parte de registro y manejo de usuarios en el sistema.  
   Puntos de función totales: **6**.
4. **Selección de personaje y afición**: implementa la “creación” de un nuevo personaje   
   Puntos de función totales: **1**.
5. **Desbloqueo de habilidades:** engloba la implementación y la interacción del usuario con el árbol de habilidades. Puntos de función totales: **2**.
6. **Gestión de recursos:** implementación de la gestión completa de recursos de un jugador (transacciones y ganancia automática de recursos).Puntos de función totales: **18**.
7. **Uso de habilidades de jugador:** implementación de todo lo relacionado con las habilidades de jugador (habilidades individuales).Puntos de función totales: **14**.
8. **Participar en acciones grupales:** interacción de los jugadores con acciones grupales ya abiertas (donación de recursos en las acciones)Puntos de función totales: **3**.
9. **Registro de acciones completadas:** Representa la implementación del efecto al completar las acciones grupales. Por ejemplo, aplicar el beneficio correspondiente a todos los participantes de una acción grupal.Puntos de función totales: **36**.
10. **Funcionamiento de la liga (encuentros):** Engloba la gestión de los equipos y el calendario; es decir la creación de partidos y el registro de partidos jugados.Puntos de función totales: **27**.
11. **Asistir a los partidos (jugados o no):** Implementación de la funcionalidad de “asistir a un partido”Puntos de función totales: **3**.
12. **Partido: jugarse:** Implementación del algoritmo de un partido. Puntos de función totales: **40**.
13. **Partido: participar: Implementación de**Puntos de función totales: **60**.
14. **Ganancia de experiencia:** implementación de la evolución del personaje.Puntos de función totales: **3**.
15. **Mensajería: Entendido como “añadido” al proyecto al estar relacionado**Puntos de función totales: **1**.
16. **Chat**Puntos de función totales: **15**.
17. **Gestión de objetos:** definición e implementación de objetos que puedan comprar los jugadores para mejorar el personaje (añadido a la arquitectura)  
    Puntos de función totales: **2**.
18. **Arte:** engloba todo lo relacionado con la parte artística propia de a) un juego; y b) diseño UX (*User eXperience*): Diseño de la página web con vistas a una posible comercialización, incluye la elección de los colores, disposición de elementos y menús para el usuario.   
    Puntos de función totales: **27**.
19. **Maquetación de la página web:** “Implementación” del diseño de las vistas de la página webPuntos de función totales: **14**.
20. **Tareas estructurales de apoyo al proyecto:** Módulo que representa la organización del grupo de programación y otras tareas estructurales.Puntos de función totales: **20**.

## 2.2 Definición de la progresión en las tareas

Además de los puntos de función, se ha definido una escala para medir la progresión en una determinada tarea.

Se tendrán en cuenta cuatro factores:

1. **Implementación** de la funcionalidad. Hasta un 40% del total del trabajo.
2. **Seguridad**: la implementación es segura; esto es, si hay acceso a una base de datos asegura la atomicidad por medio de transactions, se controla la la inyección sql por parte del usuario. Hasta un 20%
3. **Testing**: la implementación supera una serie de test automatizados. Hasta un 20%
4. **Integración**: la funcionalidad se integra con el resto de funcionalidades y se pasan nuevos test de funcionalidad conjunta. Hasta un 20%

Por ejemplo; en un punto del desarrollo la implementación del partido está prácticamente completa (75%) y asegurada la seguridad; pero no se han diseñado aún tests ni se ha pasado a la parte de la integración.   
En esas condiciones la funcionalidad del partido se ha completado en un 50%. Si los puntos de función totales previstos son 40, se habrían completado 20 puntos de función.

# 3. Anexos

1. Manuel Artero, [Documento con la previsión de las líneas de código del proyecto](https://docs.google.com/document/d/1zIIt-PmU0VfOqBMmNYBOliZVvQXJMVVifPBM4NFi5XU/edit#heading=h.a8e2sbpde1ha) : https://docs.google.com/document/d/1zIIt-PmU0VfOqBMmNYBOliZVvQXJMVVifPBM4NFi5XU/edit#heading=h.a8e2sbpde1ha
2. Manuel Artero, [Documento con el desglose de los puntos de función y la división del proyecto](https://docs.google.com/spreadsheet/ccc?key=0Ao09kH8yz2wzdEtUZWhzWi02bWtjLUt5bjJXVXVzOEE#gid=0). : https://docs.google.com/spreadsheet/ccc?key=0Ao09kH8yz2wzdEtUZWhzWi02bWtjLUt5bjJXVXVzOEE#gid=0