UNIDAD TEMÁTICA 3 – Diseño y UML – Trabajo final de unidad

Equipo 5

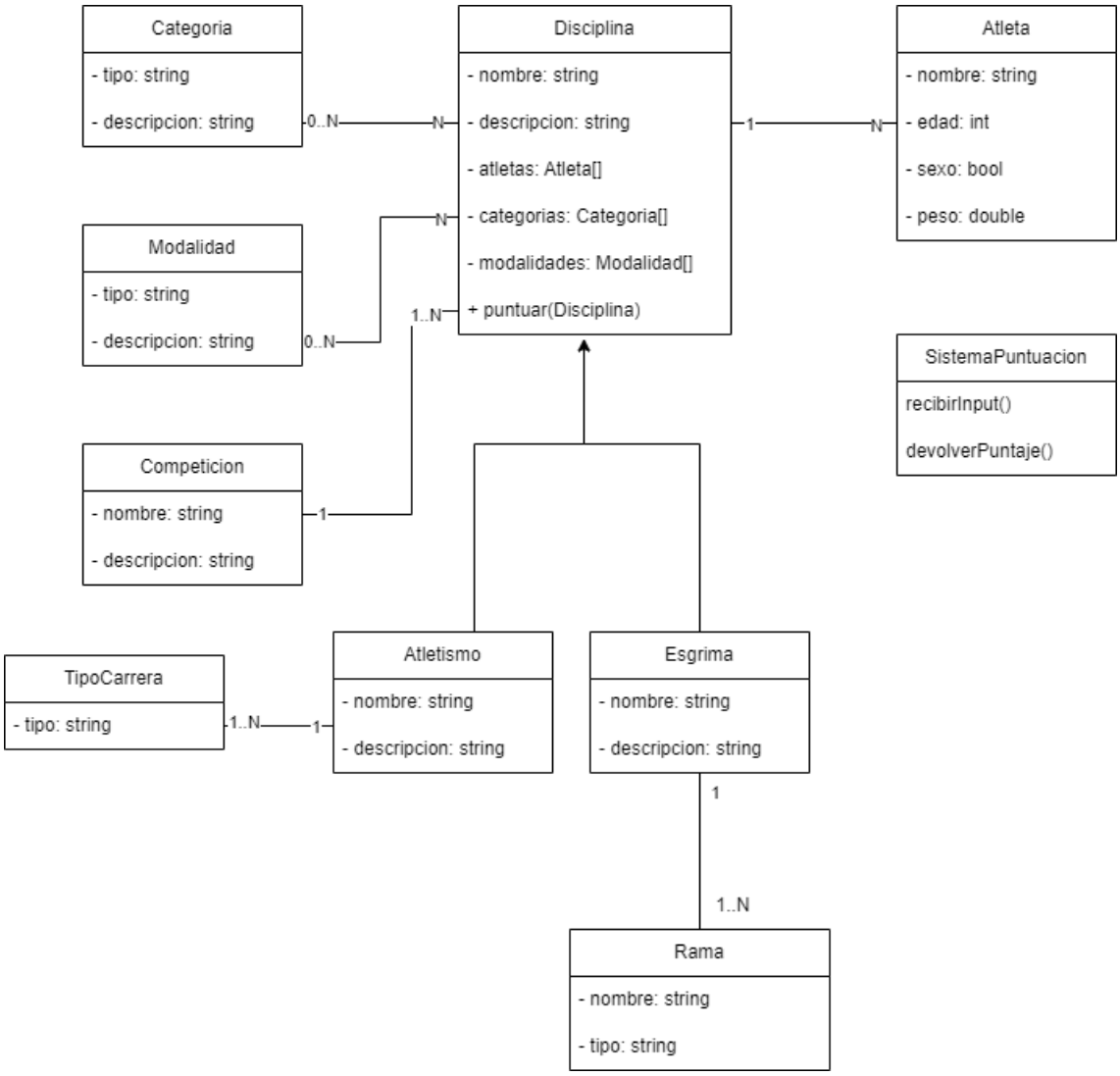
Parte 1: Requisitos

Requerimientos funcionales:

1. El sistema debe gestionar las puntuaciones de la disciplina atletismo en función del tiempo o la distancia recorrida.
2. El sistema debe poder calificar la disciplina atletismo según los tres diferentes tipos de carrera: pista, campo, combinadas.
3. El sistema debe gestionar las puntuaciones de la disciplina natación en función del tiempo.
4. El sistema debe diferenciar la disciplina de natación en sus distintas modalidades: natación libre, espalda, braza, mariposa, relevos, natación en aguas abiertas.
5. El sistema debe especificar la categoría de la competencia: juvenil, abierta, paralímpica y de maestros.
6. El sistema debe gestionar las puntuaciones de la disciplina gimnasia en función de la dificultad y ejecución de la rutina.
7. El sistema debe gestionar las puntuaciones de la disciplina clavados en función de la dificultad y ejecución de los saltos.
8. El sistema debe gestionar las puntuaciones de la disciplina halterofilia en función del total del peso levantado.
9. El sistema debe diferenciar las categorías en halterofilia: femenino y masculino.
10. El sistema debe puntuar la disciplina halterofilia según sus dos levantamientos: arrancada y dos tiempos.
11. El sistema debe gestionar las puntuaciones de la disciplina esgrima según el que obtenga mayores puntos en los asaltos.
12. El sistema debe puntuar según la rama de esgrima: individual y por equipos
13. El sistema debe gestionar las puntuaciones de la disciplina surf según los distintos criterios: selección de la ola, maniobras, variedad, potencia y radicalidad y progresión.
14. El sistema debe gestionar las puntuaciones de la disciplina kitesurf en función de la técnica, dificultad, altura y amplitud, variedad e impresión general.
15. El sistema debe diferenciar las distintas categorías de kitesurf: velocidad, freestyle y de olas.
16. El sistema debe determinar en qué puesto quedó cada competidor luego de terminada la competición.

Parte 2: Diagramas

- Diagrama de clases:

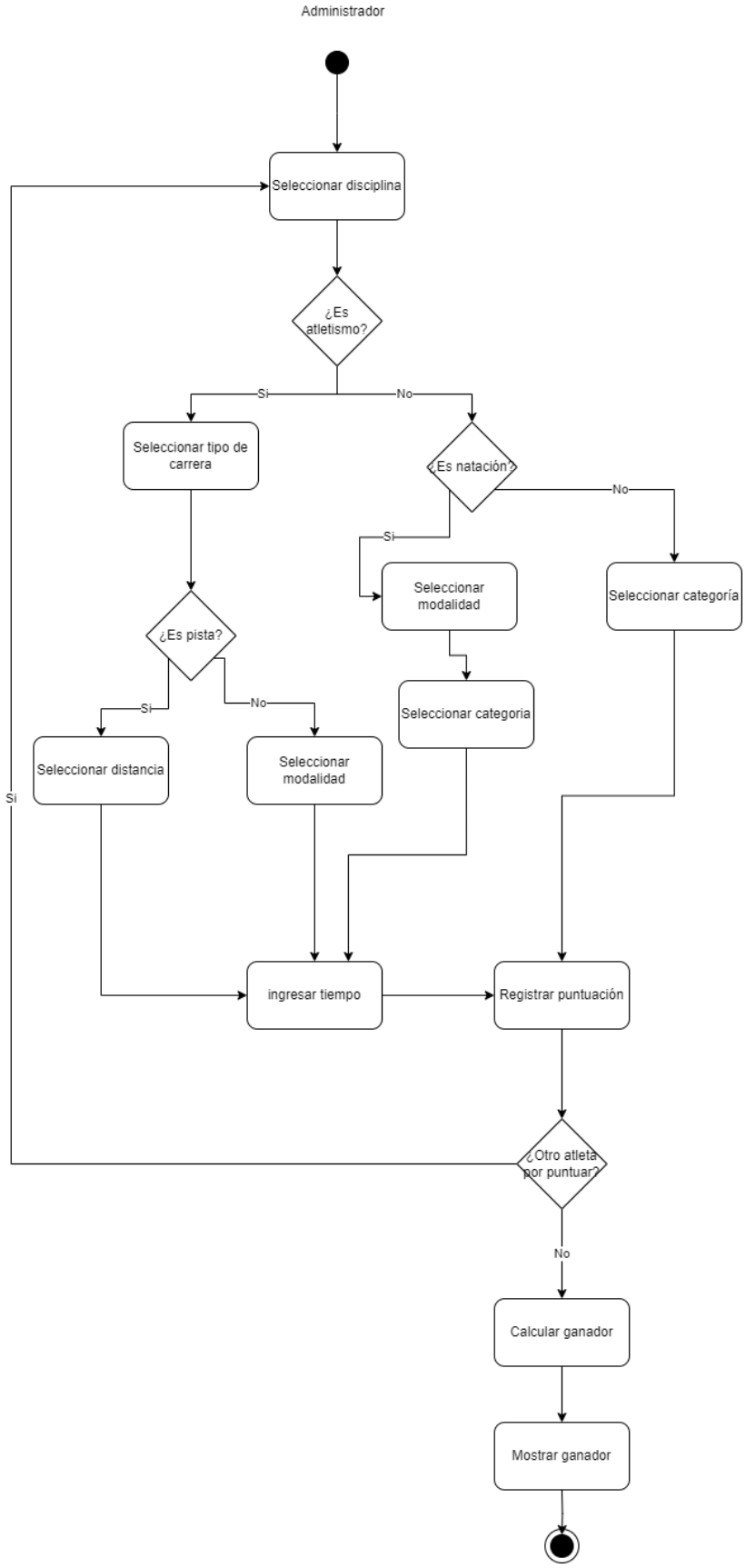


Justificación

Un diagrama de clases permite organizar de manera estructurada las entidades relacionadas en el sistema. Esto permite comprender el sistema en su conjunto y las relaciones entre sus componentes. Estas relaciones ayudan a comprender cómo interactúan los componentes del sistema entre sí.

También, cuando modelamos con un diagrama de clases podemos identificar de manera temprana las consideraciones que debemos tener cuando implementemos el sistema. Esto facilita la implementación misma, ya que tenemos una guía clara sobre cómo deben organizarse y relacionarse los componentes.

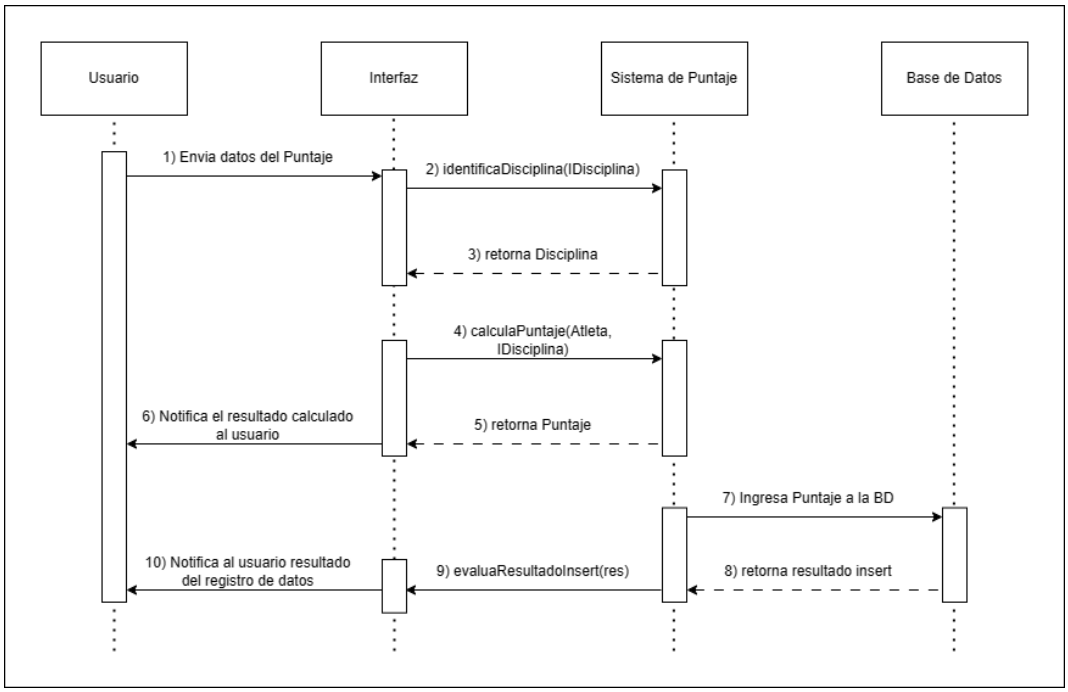
- Diagrama de actividad



Justificación

El diagrama de actividades ayuda a comprender el flujo de las mismas dentro del sistema de gestión de puntuaciones. Permite visualizar la secuencia de acciones que lleva el sistema, empezando con la selección de la disciplina, hasta el final mostrando los ganadores. El flujo de dichas acciones está representado mediante flechas, así como también la toma de decisiones que tiene el sistema. En este caso al ser automatizado, el diagrama de actividad cuenta con un rol de administrador, el cual se encarga de proporcionar la información necesaria para activar el proceso de puntuación y determinación de ganadores de cada competencia.

- Diagrama de secuencia

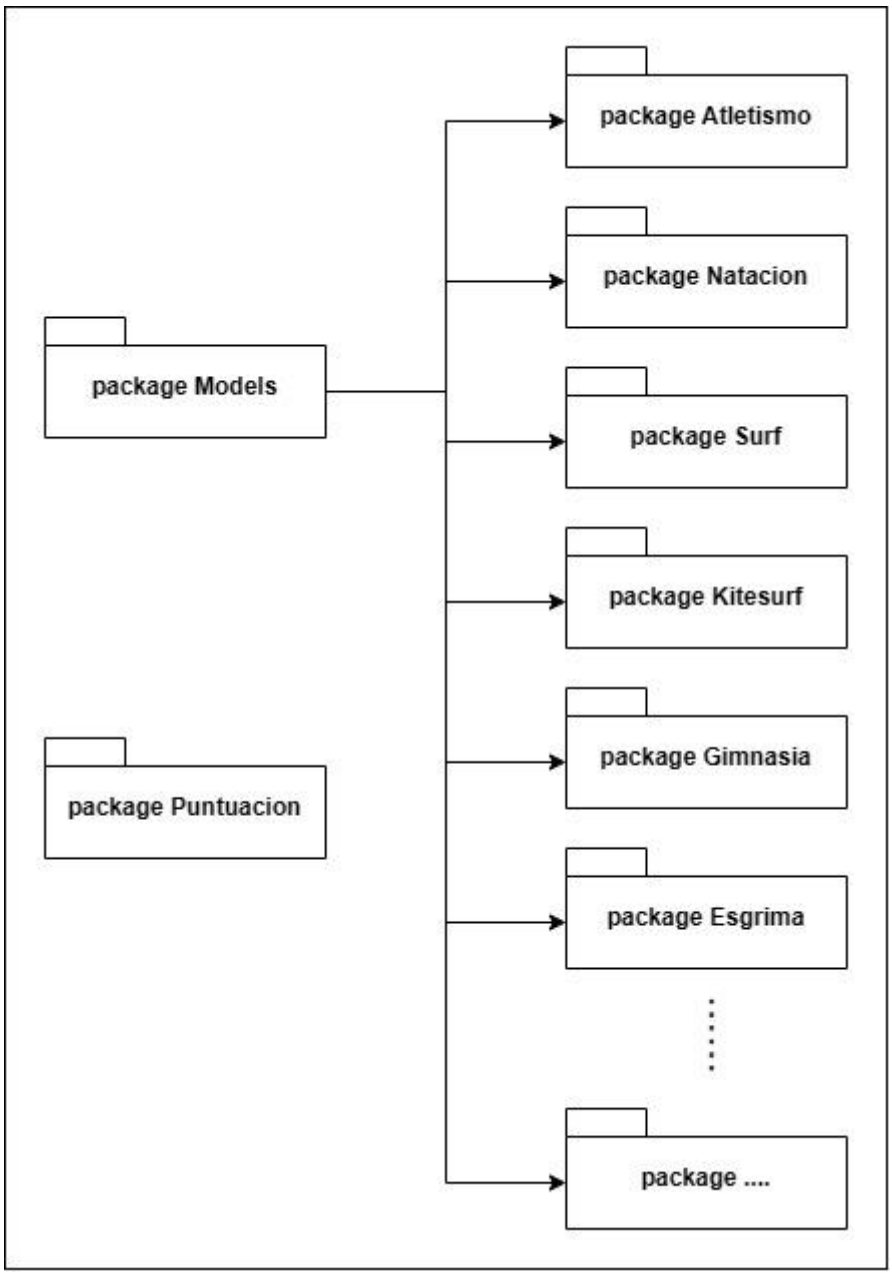


Justificación

El diagrama de secuencia nos ayuda a comprender como es la interacción de los componentes de nuestro sistema, además de brindar una mejor comprensión del mismo. También sirve como parte de documentación para desarrolladores de este, o futuros, y para personas que necesiten entender el funcionamiento en caso de integración a nuestro sistema.

En este caso se muestra la interacción con el sistema de puntajes externo, desde cuando el usuario carga los datos indicados hasta que los mismos son persistidos en la base de datos del sistema.

- Diagrama de Paquetes



Justificación

Este diagrama permite tener una idea global de cómo está estructurado el programa y la distribución de sus módulos, pero sin entrar en detalles de cómo será llevada a cabo la implementación en el código. También permite tener una idea de que paquetes depende cada módulo, lo que facilita organizar la planeación del proyecto, ya sabiendo de antemano la dependencia de cada módulo.