Ejercicio PostgreSQL: Simulador La Volta

La organización de **La Volta** nos ha pedido crear un juego de simulación online. Como era de esperar la simulación estará basada en el mundo del ciclismo. El sistema tendrá que ser accesible desde diferentes dispostivos (iphone, android, ipad, mac, pc, …). Debido a esto hemos decidido integrar la mayor parte de la lógica del programa en la base de datos (es decir utilizaremos **transacciones, disparadores y funciones almacenadas**).

Empezaremos con una versión reducida y de demostración del simulador. Si a los organizadores les pareciese correcto e interesante acabaríamos de crear el programa más adelante.

Esta versión del juego no tendrá interfaz gráfica, sólo dispondremos como interfaz de juego de los procedimientos almacenados, así que será parecido a una aventura conversacional.

El simulador representará a dos Equipos de Corredores. Cada equipo tendrá un nombre y estará dirigido por un Director de equipo que, en la **versión de demostración**, controla a tres corredores.

De un Corredor vamos a necesitar conocer:

- El rol que juega en el equipo (gregario o jefe de filas).
- Su velocidad máxima (numero de metros que es capaz de recorrer compitiendo al 100% de esfuerzo en un sólo turno). Este valor se guarda en un almacén llamado `MetrosMaximosPorTurno`.
- Reservas energéticas en forma de glucógeno, grasa y agua. Estas reservas van a tener un valor máximo para cada corredor. Este valor máximo en la **versión de demostración** lo fijaremos en 100, pero con el tiempo debería poder variar en función del entrenamiento (subiría) y del nivel de esfuerzo en carrera (bajaría). Estas reservas de energía o depósitos van a estar llenos hasta cierto nivel, este nivel que variará en función del transcurso del juego, nunca superará el valor máximo propio del corredor. En esta **versión de demostración** se empezará la partida con los depósitos siempre en su valor máximo.

Durante la partida podremos modificar el nivel de esfuerzo de un corredor. Este valor variará entre 0% de esfuerzo y 100% de esfuerzo. Al empezar la partida el nivel de esfuerzo se situará en el 50% para todos los corredores. Para conseguirlo lo podremos hacer de dos formas:

- Por decisión del corredor (llamando a la función `corredorSetEsfuerzo(valor)`)
- Por decisión del diretor de equipo (llamando a la función `directorSetEsfuerzo(valor)`). Cuando la decisión la toma el director de equipo todo el equipo pasa a trabajar en el mismo nivel de esfuerzo.

Tambien necesitaremos saber la distancia recorrida por el Corredor. Cada corredor es capaz de desarrollar un velocidad máxima que se guardará en un almacén llamado `MetrosMaximosPorTurno`, con el tiempo debería poder variar en función del entrenamiento (subiría) y del nivel de esfuerzo en carrera (bajaría). En la **versión de demostración** dispondremos de una velocidad de 1.000 metros por turno para cada equipo que se tendrá que repartir entre los diferentes corredores antes de empezar la partida (esta regla no hace falta modelarla en la **versión de demostración**, se supone que el usuario la va a cumplir y no hace falta que lo controlemos).

Al empezar la etapa todos los corredores estarán en la línea de salida, es decir, habrán recorrido 0 metros. Cada vez que se ejecute la función `pasarTiempo()` se recalculará la distancia recorrida y el nivel de los depósitos de energía en función del nivel de esfuerzo exigido a los corredores. (más adelante acabaremos de especificar este punto).

Cada equipo dispondrá de una bolsa de alimentos. En un futuro tendremos un

catálogo de alimentos compuestos cada uno de ellos de una carga de puntos de glucosa, de grasa y de agua que darán forma a esa bolsa, pero **por ahora, en la versión demo** la bolsa de alimentos contendrá directamente el total de puntos de glucosa, grasa y agua disponibles para disputar la etapa. Estos puntos se podrán suministrar a los corredores durante la carrera. Antes de empezar la etapa se tendrá que cargar esa bolsa repartiendo, por ejemplo, 500 puntos de alimentos entre las tres categorias, (para ello usaremos la función `EquipoSetBolsa(puntosGlucosa, puntosGrasa, puntosAgua)`). Una vez más no hace falta controlar el reparto de los 500 puntos en esta **versión de demostración**, se supone que el usuario la va a cumplir y no hace falta que lo controlemos.

El director de equipo podrá dar la orden de comer a todo el equipo o a un corredor en concreto llamando a `directorOrdenComer(puntosGlucosa,puntosGrasa, puntosAgua` o a `directorOrdenComerCorredor(idCorredor,puntosGlucosa, puntosGrasa,puntosAgua)`) Cuando se da la orden de comer los puntos que se asignan a los corredores se restan de la bolsa del equipo.

Cuando los jugadores hayan acabado de dar de comer a sus corredores y hayan configurado el nivel de esfuerzo de cada uno de ellos se llamará a la función `pasarTiempo()`

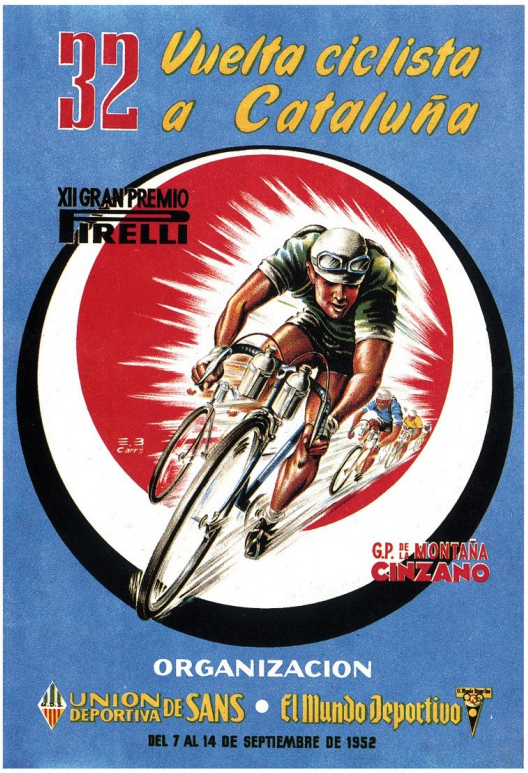
¿Qué hace la función `pasarTiempo()`? La función `pasarTiempo()`es la encargada de hacer evolucionar la partida. Cuando se llama a esta función se va a actualizar la distancia recorrida por cada corredor en función del nivel de esfuerzo. Para recorrer esa distancia habrá hecho un gasto de reservas de glucosa, grasa y agua que también va en función del nivel de esfuerzo desempeñado.

Veamos las reglas para calcular la distancia recorrida y el gasto de reservas en función del nivel de exigencia que está desarrollando el corredor:

- Si un corredor tiene el nivel de alguno de los depósitos a 0 no se mueve.
- La distancia se calcula con la siguiente fórmula:
  - **`Distancia = %Esfuerzo * MetrosMaximosPorTurno.`**
  - `Distancia = 0,5 * 300metros = 150metros.`
- El gasto de agua se calcula con la siguiente fórmula:
  - **`Gasto Agua = (%Esfuerzo * 100)`**
  - `Gasto Agua = (0,0 * 100) = decrementar 0 puntos`
  - `Gasto Agua = (0,5 * 100) = decrementar 50 puntos`
  - `Gasto Agua = (1,0 * 100) = decrementar 100 puntos`
- El gasto de grasa se calcula con la siguiente fórmula:

- **Gasto Grasa = 100 -(%Esfuerzo * 100)**
- Gasto Grasa = 100 - (0,5 * 100) = decr. 50 puntos
- Gasto Grasa = 100 - (0,0 * 100) = decr. 100 puntos
- Gasto Grasa = 100 - (1,0 * 100) = decr. 0 puntos

- El gasto de azucar se calcula con la siguiente fórmula:
  - **Gasto Azucar = %Esfuerzo * 10**
  - Gasto Azucar = 0,5 * 100 = decr. 50 puntos
  - Gasto Azucar = 0,0 * 100 = decr. 0 puntos
  - Gasto Azucar = 1,0 * 100 = decr. 100 puntos



Hoja de ruta sugerida y normas a seguir:

- Dibuja el diagrama de clases UML representativo de esta aplicación.
- Decide dónde vas a implementar como mínimo una transacción y un disparador. (se evaluará).
- Los nombres de las funciones/métodos tienen que reflejar a que clase pertenecen, como postgreSQL no nos ofrece nada mejor lo que haremos será seguir un protocolo a la hora de crear las funciones/métodos: El nombre de las funciones empieza con el nombre de la clase a la que pertenecen + "_" + el nombre de la función/método (Por ejemplo: `jefeDeEquipo_DarOrden()`)
- Crea las clases y atributos en postgresSQL con la jerarquía de clases definida en el diagrama UML.
- Crea un par de consultas/funciones que permitan conocer:
  - El estado de la carrera: Todos los corredores de un equipo ordenados por metros recorridos y estado de sus depósitos de energia
  - Información de equipo: Muestra el estado de los corredores y de la bolsa de alimentos del equipo.
- Crea después los métodos y la transacción y disparador que has decidido hacer en el segundo punto y que son el mínimo imprescindible para evaluar la actividad.
- Una vez configurados los equipos (bolsa de alimentos cargada, velocidades máximas de los corredores definidas, ...) haz una partida con un compañero. Para hacer más interesante la partida se pueden definir, una vez configurados los equipos, diferentes opciones en las que os pondréis de acuerdo mutuamente (no hay que modelarlo). Por ejemplo:
  - Distancia de la etapa, definida en metros o turnos.
  - ¿Quién gana? El que llegue antes a meta, pero... ¿con un corredor?, ¿con dos?, ¿con el equipo completo?
- Para facilitar el trabajo podéis usar el interfaz de administración gráfica de PostgreSQL si os parece adecuado.