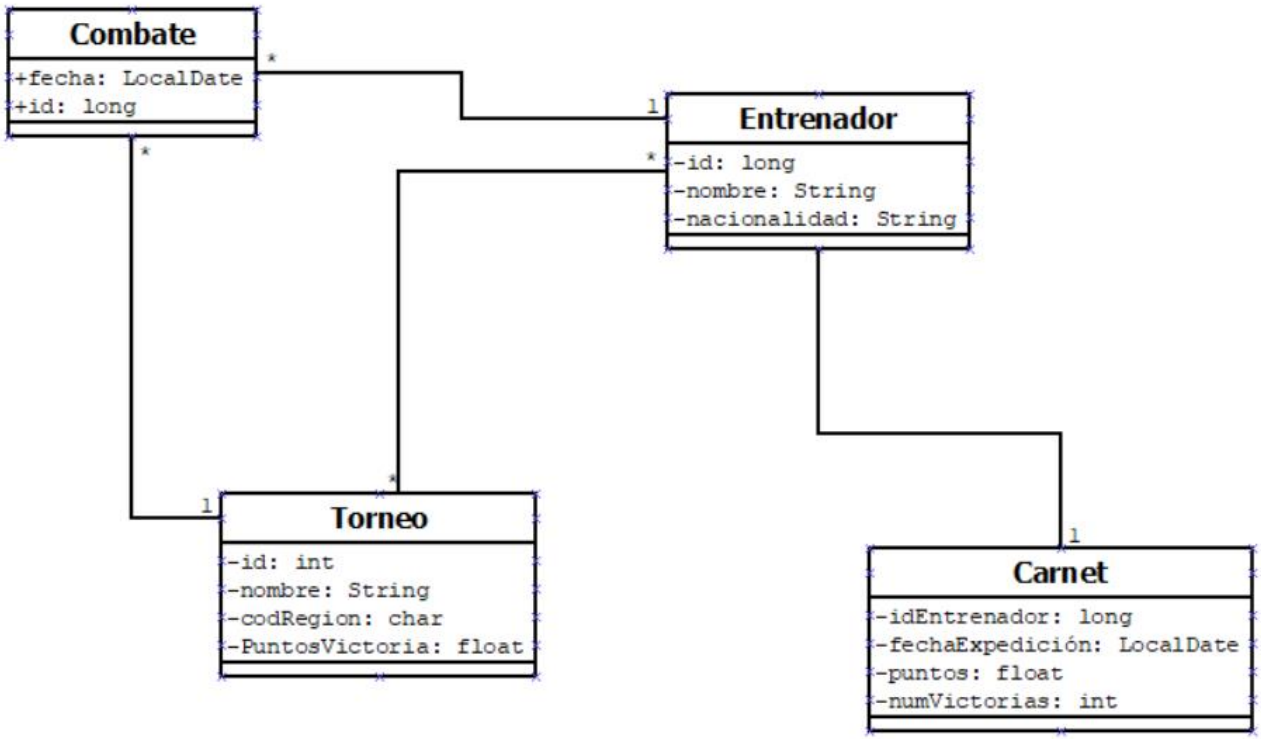
Centro Integrado de Formación Profesional
AVILÉS
Principado de Asturias

ACCESO a DATOS – 2º DAM

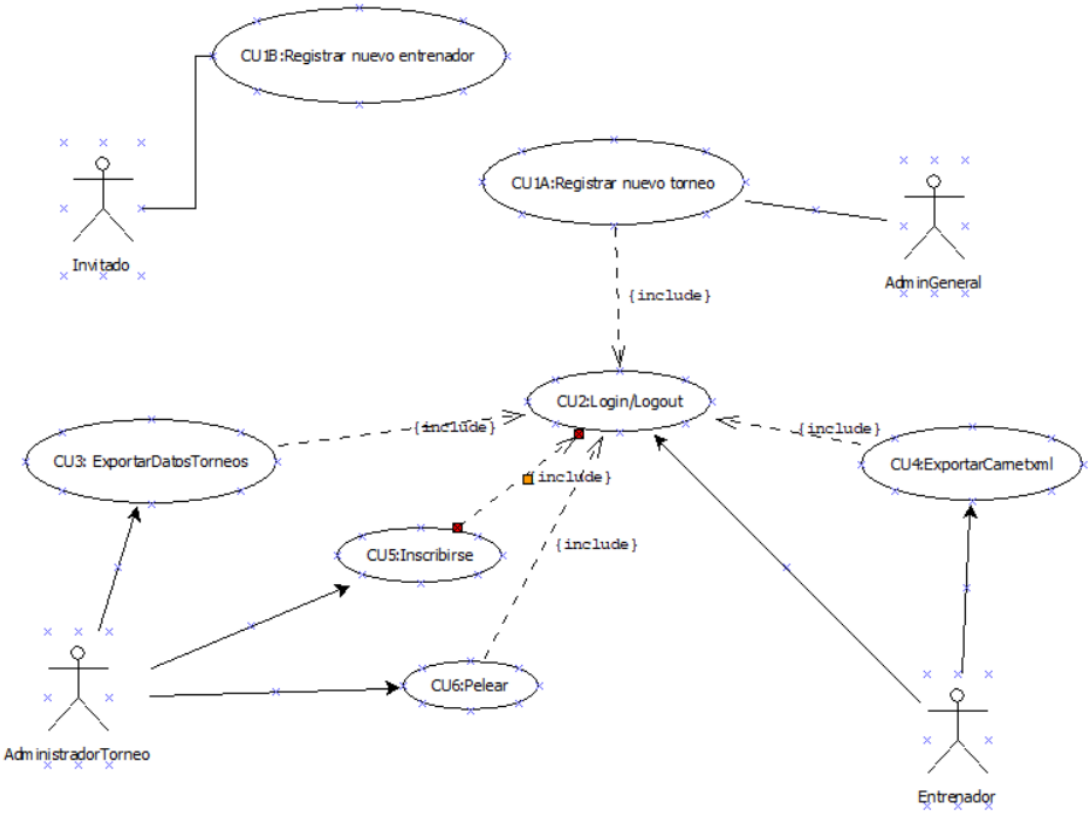
TAREA 3: Herramientas de mapeo

DESCRIPCIÓN:

DESCRIPCIÓN:
En esta tarea se van a trabajar las mismas especificaciones que para las tareas anteriores, en relación al programa de gestión de rutas, pero se va a cambiar la tecnología tanto de la creación/lanzamiento del proyecto como del acceso/persistencia a/de los datos del mismo. A partir de ahora se trabajará con el framework Spring de Java, de manera que se genere y se configure automáticamente con Maven, se arranque directamente con Spring Boot y se tenga como tecnología de acceso a datos mediante JPA implementado a partir del motor de persistencia Hibernate.
El modelo de clases de la aplicación es el siguiente:



Se quiere tener, además, un sistema de autenticación y de sesión por perfiles de usuario que otorgarán el acceso a diferentes funcionalidades del sistema. El diagrama de casos de uso inicialmente sería el siguiente:


CU1B:Registrar nuevo entrenador
Invitado
CU1A:Registrar nuevo torneo
AdminGeneral
{include}
CU2:Login/Logout
{include}
CU3: ExportarDatosTorneos
{include}
CU4:ExportarCametxml
{include}
CU5:Inscribirse
{include}
CU6:Pelear
AdministradorTorneo
Entrenador

De momento sólo se tendrán 4 perfiles de usuario:

- **Invitado**: es el perfil que no se autentica, por lo que sus funcionalidades son limitadas.
- **Administrador de torneo**: es la persona encargada de un torneo. Registralos datos de los combates de los entrenadores en el torneo del que es responsable.
- **Entrenador**: es la persona que pelea en los torneos, realizando diferentes combates.
- **Administrador General**: es único en el sistema, podrá llevar a cabo tareas de administración trasintroducir sus credenciales de acceso.

De momento sólo se estiman las siguientes funcionalidades:

- **CU1: Registrar:** es el proceso de alta en el sistema para un nuevo torneo para nuevo Entrenador.
    A. **Nuevo Torneo**: el sistema permitirá la introducción de un nuevo torneo sólo al usuario Administrador general. Deberá dar el nombre y la región del nuevo torneo, así como el usuario de tipo administrador de torneo que será responsable del mismo (nombre y contraseña). El sistema comprobará que no exista ya ese torneo, informando en tal caso. Si todo va bien, el nuevo torneo quedará añadido al sistema habiéndole asignado un responsable administrador del mismo. También se crearán 3 combates asociados a dicho torneo con las fechas que se pidan por pantalla y se asignará la puntuación por ganar el torneo.
    B. **Nuevo entrenador**: Un usuario invitado podrá, desde un torneo concreto, introducir sus datos (nombre y nacionalidad), para darse de alta en el sistema. Éste comprobará si ya existen esos datos, informando al usuario en tal caso. Si no, el invitado quedará dado de alta en el sistema y se le asignará un carné propio y único con la fecha de expedición actual y su primer torneo asociado.
- **CU2: Login/Logout:** es el proceso de autenticación/cierre de sesión en el sistema. El usuario introducirá sus credenciales de acceso al sistema y éste comprobará que son correctas (indicando error en caso contrario) para dar posteriormente acceso a las funcionalidades del sistema que le corresponda en función del perfil del usuario autenticado. De esta forma se guarda la sesión del usuario hasta el momento que cierra su sesión (Logout), cuando pasa a tener el perfil de invitado nuevamente.
- **CU3: Exportar datos torneo:** el responsable de un torneo podrá exportar los datos de los combates y entrenadores de su torneo. Se reflejarán los datos del torneo (id, nombre, región) la lista de los combates celebrados, así como los entrenadores que participaron con su nombre, id y si gano o no (Se mostrará por pantalla y se dará la opción de sacar por fichero).
- **CU4: Exportar Carnet en XML:** cada entrenador, tras autenticarse, podrá visualizar y exportar los datos de su propio carnet. Esto incluye: el id de entrenador, nombre y nacionalidad, fecha de expedición del carnet, puntos conseguidos y la colección de torneos (nombre y región) en los que ha participado hasta el día actual, indicando en qué fecha se realizó combates y si los gano o no. Esta información se exportará en formato XML, con la estructura del ANEXO I.

- **CU5: inscribirse**: un administrador de torneo recibe a un entrenador que llega a su torneo. Le pide su carnet de entrenador. Se asignarán los combates para ese entrenador en ese torneo (2 combates) y se rellenarán los datos de cada combate, actualizando los datos del entrenador. Si el torneo esta lleno se informara al entrenador de que no puede inscribirse.
- **CU6: pelear**: Un administrador de torneo ejecutara pelear. Se comprobará que el torneo tenga todos sus combates rellenos con entrenadores, si es así se resolverán los combates de manera aleatoria y el ganador del torneo (el que más combates gane, en caso de empate también aleatorio) se actualizarán los datos relativos a puntos conseguidos y al nº total de victorias de cada entrenador.

TAREA:

Será necesario crear un nuevo proyecto que sea generado a partir de Maven y que disponga del sistema de arranque automático Spring Boot. Este proyecto trabajará con el framework Spring de Java con la capa de persistencia mediante JPA+Hibernate, por lo que será imprescindible cargar todas las dependencias necesarias en el fichero *pom.xml* para que éstas se descarguen automáticamente al importar el proyecto. Se recuerda que la entrega debe incluir todos los mecanismos necesarios para su ejecución directa, sin que sea necesaria ninguna configuración extra del proyecto ni del entorno de ejecución.

- En primer lugar, deberán implementarse todas las entidades del diagrama de clases y cualesquiera otras que se entiendan necesarias para el manejo de las funcionalidades expresadas en el diagrama de casos de uso. Estas entidades deberán estar correctamente anotadas mediante JPA y, a través de las propiedades expresadas en el fichero *application.properties* de Spring del proyecto, se creará automáticamente el esquema de la base de datos relacional con la que se trabajará. Sólo será imprescindible que aparezcan implementadas las relaciones entre entidades que sí están marcadas en el diagrama de clases aportado.

- Se crearán clases del tipo *@Repository* que contendrán los métodos necesarios para el acceso a los datos en la base de datos relacional. Sólo estas clases podrán realizar esas operaciones. En relación al mecanismo de consultas a la BD se podrá utilizar cualquiera de los siguientes: JPQL/HQL, JPA Data, Criteria Query o una mezcla de ellos.

- Se crearán clases del tipo *@Service* que contendrán los métodos necesarios para aportar las funcionalidades disponibles para cada entidad. Estas clases inyectarán directamente a las clases de tipo repositorio para poder trabajar con los métodos de éstas. Así mismo, la/s clase/s de tipo controlador inyectarán directamente a estas clases de servicio para poder trabajar con sus métodos.

- Se preparará el proyecto para que disponga de las clases de configuración necesarias, tales que permitan arrancar el programa a partir de una función/clase principal que vaya inyectando directamente las clases del proyecto que constituyan Beans de Spring para su posterior manejo.

- Se implementaran los dos casos de uso que faltan Inscribirse y pelear.

NOTA: Se deberá trabajar a partir de lo implementado para el proyecto de la tarea anterior, estructurando adecuadamente el nuevo código fuente, refactorizando el que ya había para que funcione todo correctamente.

ENTREGA:

La entrega se realizará a través del aula virtual del módulo en Educastur, y consistirá en un archivo comprimido (.zip o .rar) con el proyecto Java y el fichero SQL de la solución que deberá estar correctamente preparado para su despliegue y ejecución directamente, sin la necesidad de configuraciones añadidas, con una correcta compilación y ejecución del programa. Se tendrán en cuenta todos los mecanismos de seguridad que se añadan a la solución, con el fin de evitar cierres inesperados del programa.

En primer lugar, deberá diseñarse un modelo relacional de tablas para persistir y recuperar la información relativa a todas las entidades anteriores, con sus correspondientes tipos de datos, valores por defecto, claves primarias y foráneas, así como otras restricciones. Deberá incluirse en la entrega el script sql con la creación de dicho modelo relacional, para poder ser creada una BD relacional MySQL de nombre `bdentrenadores_<nombreestudiante>`. Recuérdese que el proyecto Java ha de trabajar con el generador de proyectos Maven (a través de la inclusión de las dependencias que se estime necesarias en el fichero pom.xml).

Para esta tarea 3 se piden los siguientes apartados:

1. Preparar la BD de datos para que contenga, al menos, los siguientes datos iniciales:
   - 3 Torneos diferentes y 5 Entrenadores distintos (éstos con su correspondiente Carnet personal).
   - 1 Entrenador Con al menos 1 torneo terminado y las puntuaciones obtenidas.
   - 1 Torneo con todos los combates llenos listo para ejecutar pelear.
   - 1 Torneo con hueco para inscribir al menos un nuevo entrenador
   - Todos los datos que se manejen inicialmente deberán ser coherentes y no podrá haber inconsistencia en ellos. Esto incluye el disponer (de alguna forma) de un sistema de cálculo de la distancia entre 2 paradas para marcar ese dato en el Carnet del Peregrino. Incluir en la entrega esa estructura para las distancias con datos válidos.

2. Implementar el CU5: Inscribirse. El detalle de este caso de uso es el siguiente:

CU5: Inscribirse
Precondiciones:
El usuario se encuentra ya autenticado con el perfil de administrador de torneo
Postcondiciones:
Queda registrado en el sistema el entrenador para este torneo y se le asignarán dos combates, actualizando consecuentemente los datos de su carnet.
Acciones:

1) El usuario accede a la pantalla de "Inscribir entrenador". 2) El sistema carga directamente la información básica del torneo y la muestra por pantalla, entre esta información estará si el torneo tiene plazas libres, si no es así directamente nos dirá que no podemos inscribir a más entrenadores y se volverá al menú. 3) Se muestra una lista de los entrenadores que hay en el sistema. 4) El usuario introduce el identificador del entrenador que acaba de llegar al torneo y el sistema muestra los datos de éste. 5) El usuario confirma los datos para inscribir él entrenador. En caso de no confirmación se termina directamente el proceso. 6) Se muestran los datos de los combates que el entrenador va a realizar. 7) Se registra en el sistema los datos del nuevo entrenador, así como los combates que va a realizar. 8) El sistema informa del éxito de la operación de inscripción del entrenador en el torneo.
Excepciones
1) Si el identificador de entrenador es inválido se avisa al usuario para que vuelva a introducirlo. 2) Si hay algún problema durante el proceso de inscripción se revierte el proceso (rollback) avisando alusuario y se finaliza. 3) Si hay algún problema durante el registro de los combates se revierte el proceso (rollback) avisando al usuario y se finaliza.

3. Implementar el CU6: Pelear. El detalle de este caso de uso es el siguiente:

CU6: Pelear
Precondiciones:
El usuario se encuentra ya autenticado con el perfil de administrador de torneo El torneo esta completo.
Postcondiciones:
Queda registrado quien gana el torneo y se asignan los puntos al entrenador vencedor.
Acciones:

9) El usuario accede a la pantalla de "Combate".
10) El sistema carga directamente la información básica del torneo y la muestra por pantalla.
11) EL usuario confirma que se desarrollen los combates. Si el torneo no estaba lleno se informará que no se puede combatir aún.
12) Si todo ha ido bien, se mostrarán otra vez los combates con el ganador de cada uno de ellos y también el ganador del torneo.
13) Todo deberá de quedar registrado en la BD y el torneo se marcará como finalizado.

Nota:
El método por el cual cada entrenador gana o no cada combate será aleatorio y quedará a criterio del alumno/a su implementación. Cada entrenador disputa siempre dos combates, el ganador del torneo será quien gane sus dos combates, si hubiese mas de un entrenador con sus dos combates ganados, el ganador del torneo se decidiría aleatoriamente entre estos.

Excepciones

1) Si el identificador de entrenador es inválido se avisa al usuario para que vuelva a introducirlo.
2) Si el torneo esta cerrado se informara al usuario y se saldrá.
3) Si hay algún problema durante el proceso de combate se avisara alusuario y se finaliza.

NOTA: Se deberá trabajar a partir del proyecto de la tarea anterior, estructurando adecuadamente el nuevo código fuente, refactorizando el que ya había para que funcione todo correctamente.

ENTREGA:

La entrega se realizará a través del aula virtual del módulo en Educastur, y consistirá en un archivo comprimido (.zip o .rar) con el proyecto Java de la solución que deberá estar correctamente preparado para su despliegue y ejecución directamente, sin la necesidad de configuraciones añadidas, con una correcta compilación y ejecución del programa. Se tendrán en cuenta todos los mecanismos de seguridad que se añadan a la solución, con el fin de evitar cierres inesperados del programa.