Acceso a Datos - Proyecto

Como repaso final de la asignatura de Acceso a Datos, vamos a crear un servicio web.

Este servicio web se trata de API que servirá datos estadísticos sobre jugadores de la NBA, comprendidos entre las temporadas 1999/2000 y 2007/2008.

Para ello se presenta la siguiente información la cual tendremos que incluir en la base de datos y posteriormente realizar los endpoints correspondientes para poder obtener la información que se pida.

1. Tenemos un servidor de base de datos disponible llamado add-dbms. Levantamos este contenedor que nos servirá de servidor de bases de datos.
2. Creamos a través de la línea de comandos la base de datos a partir del script: nba_2022-02-02.sql
3. Crearemos una carpeta que incluirá un script python por cada tabla para rellenar su información.
    a. Estos scripts deberán conectarse a la base de datos.
    b. Leer el fichero correspondiente a su tabla.
    c. Ejecutar todos los INSERTS necesarios para dar de alta toda la información.
4. Partiendo de la base para crear proyectos Symfony que se facilitó en la Práctica 6, tendremos que crear una carpeta a partir de la carpeta ***sf-app-provisioning***.
    a. Una vez creada la carpeta llamada api-nba, con el contenido de ***sf-app-provisioning*** creamos el contenedor de Docker.
5. Con el contenedor listo, crearemos un proyecto Symfony llamado ***api-nba***.
6. Una vez creado el proyecto, recordad que tenéis que copiarlo todo al directorio principal del proyecto.
7. A través de Doctrine, creamos las Entities
8. Creamos los repositorios asociados a cada Entity

9. Creamos los siguientes endpoints de los cuales, a continuación, se especifica la información que tienen que devolver. La respuesta de cada endpoint tiene que mostrarse en formato JSON:

    a. `/equipos`

    Listará para cada equipo, toda la información que tiene.

    b. `/equipos/{nombre}`

    Listará para un equipo dado su nombre, toda la información que tiene.

    c. `/equipo/jugadores`

    Listará para cada equipo todos sus jugadores.

    d. `/equipo/jugadores/{nombre}`

    Listará para cada equipo dado su nombre, todos sus jugadores.

    e. `/jugadores`

    Listará para cada jugador, toda la información que tiene.

    f. `/jugadores/{nombre}`

    Listará para un jugador dado su nombre, toda la información que tiene. En caso de que el nombre del jugador lleve espacios, en la url al escribir el nombre, **sustituid el espacio por el código '%20'**. *Las comillas no forman parte del código*.

    g. `/jugador/fisico/{nombre}`

    Listará para un jugador dado su nombre, altura en cm y peso en kg y su posición.

    La **altura** está en pies y pulgadas. 1 pie = 12 pulgadas; 1 pulgada = 2,54 cm

    El **peso** está en libras. 1 libra = 0.453592 kg

    En caso de que el nombre del jugador lleve espacios, en la url al escribir el nombre, **sustituid el espacio por el código '%20'**. *Las comillas no forman parte del código*.

h. **`estadisticas/jugador/{nombre}`**

Listará para un jugador dado todas sus estadisticas que tenga registradas en la base de datos. Mostraremos un array con la siguiente información:

```
nombre {
      temporada {
            puntos_por_partido,
            asistencias_por_partido,
            tapones_por_partido,
            rebotes_por_partido
      }
      ...
}
```

i. **`estadisticas/jugador/{nombre}/avg`**

Listará para un jugador dado su nombre, la media de todas sus estadísticas que tenga registradas en la base de datos. Mostraremos un array con la siguiente información:

```
nombre {
      temporada {
            media_puntos_por_partido,
            media_asistencias_por_partido,
            media_tapones_por_partido,
            media_rebotes_por_partido
      }
}
```

j. **`partidos/resultados/local/{nombre}`**

Listará para un equipo dado su nombre, todos los resultados en los que ha jugado como local

k. **`partidos/resultados/visitante/{nombre}`**

Listará para un equipo dado su nombre, todos los resultados en los que ha jugado como visitante

l. **`partidos/resultados/media/local/{nombre}`**

Listará para un equipo dado su nombre, la media de puntos recibidos como local.

m. **`partidos/resultados/media/visitante/{nombre}`**

Listará para un equipo dado su nombre, la media de puntos recibidos como visitante.

NOTA: Cada acción asociada a una url, estará asociada a su propio Controller. Información asociada a equipos → EquiposControler, etc…

ALTERNATIVA 1

Alternativamente, si alguien quiere implementar otra API que le sirva como trampolín para el proyecto de final de curso, estoy abierto a sugerencias, pero deberá cumplir los requisitos anteriormente descritos desde el punto 1 al punto 8.

ALTERNATIVA 2

Alternativamente, si no queréis usar DQL como hemos visto hasta ahora en las prácticas y queréis usar queries con lenguaje SQL, aquí adjunto dos ejemplos:

La respuesta a este post de StackOverflow:

How to get the raw query from EntityRepository from the controller - Stack Overflow

Enlace a la documentación oficial de Doctrine para usar queries en formato raw.

Native SQL

Entrega

Generáis un documento con la siguiente información:

1. Todos los comandos a ejecutar para cargar la información.
2. Todos los comandos que habéis ido ejecutando para crear el proyecto y las entidades
3. Enlace para cada endpoint con una captura pantalla con la información que devuelve.

Para la subida del proyecto, creamos un zip en el cual **se excluye la carpeta vendor** del proyecto.

Nomenclatura: ApellidosNombrePADD.zip **(0,5 pts)**