Bachelor's degree in Computer Science and Engineering
Tecnologías informáticas para la web
2023-2024

Práctica 1

"Plantilla de un equipo de fútbol"

Luis Daniel Casais Mezquida - 100429021
Pablo Montero Poyatos - 100429163
Javier Moreno Yébenes - 100428998
Alberto Urbano Ballesteros - 100429046

Professor:
Jesús Hernando Corrochano

Tabla de contenidos

1 Introducción

La consiguiente práctica consistirá en una evolución de la aplicación desarrollada enla anterior entrega en la cual la información requerida se guardaba en una base de datos volátil consistente unicamente en una variable de cadena a la que accedían los servlets. Como pequeño recordatotio se volverá a definir los ebjetivos funcionales de la misma así como su estructura, que ha variado levemente respecto a la version anterior.

La aplicación web realizada está basada en la posibilidad de agregar, editar y eliminar jugadores que podrían formar parte de la plantillas de los clubes más prestigiosos de España (eg: Atlético de Madrid, *er* Beti, Rayo Vallecano, F.C. Barcelona... etc). Esta vez la gestión de la plantilla de jugadores queda almacenada en bases de datos basadas en el uso tanto de JPA como de JDBC.

La estructura principal que sigue el proyecto está dividido en dos .war, uno para la implementación y el uso de base de datos que permite JPA y otro con el que permite JDBC. Ambas se componen de una carpeta principal (carpeta `src/main/`) que se divide en dos, una sección relacionada a la programación en **Java** (`java/`) y otra a cargo de las **WebApp** (`webapp/`). Respecto a la parte de **Java** es principalmente donde se incluyen los servlets, beans y utils además del sql encargado de crear la base de datos. Por otro lado el **WebApp** sigue siendo el más relacionado a la propia página web en sí y es el que más a cambiado respecto de la versión anterior, teniendo archivos JSP para distintas finalidades, como por ejemplo un *form* para editar el jugador y otro para insertarlo, un *header* y un *footer*, o el *home* y la página de error.

Lo más detacable de esta segunda versión es la migración de nuestros datos a una base de datos real, montada en MySQL lo que nos permite una mejor gestion de los datos, menos manual y más intuitiva y escalable.

2 JPA y JDBC

Dividido en dos partes principales, siendo la primera la parte técnica implementada con Java (compuesto por `entities/` para JPA y `beans/` para JDBC, `servlets/` y `utils/`) y la parte visual usando JSP (compuesta por los formularios y estructura de la página web dentro de la carpeta webapp). A continuación se analizan más en profundidad.

2.1 Entities (`entities/`) y Beans (`beans/`)

Incluye los archivos java que representan a los jugadores y a las posiciones disponibles. En el caso de las entidades, éstas están configuradas para usarse con la JPA.

2.1.1 Jugador (`Jugador.java`)

Se trata del archivo donde se define el objeto `Jugador` y se define el tipo de relación que va a tener dentro de la base de datos, en este caso es "many to one" (o n-1) lo que significa que vario jugadores pueden tener una posición (por ejemplo en un equipo hay varios mediocentros). Los atributos, todos strings, son `nombre`, `apellidos`, `dni`, `alias` y `posición`. Además se definen los *getters* y los *setters* de todos sus atributos.

2.1.2 Posicion (`Posicion.java`)

Se trata del archivo donde se define el objeto `Posiciones` y se definen estas mismas.
El archivo comienza definiendo el número máximo de cada posición junto a un *getter* del nombre de la posición como de el número máximo por posición o del número de jugagores actuales.
Se definen los métodos que añaden y borran jugadores con sus correspondiente gestión de errores (más judadores de la cuenta, posiciones no reconocidas o número negativos).
Se utiliza una función propia llamada `isMax()` que comprueba si la posición ha alcanzado su número límite de jugadores posibles en la plantilla. También se implementan las funciones de `addJugador()` y `removeJugador()` para insertar o eliminar el número de jugadores de su posición correspondiente.

2.2 FrontController.java (`FrontController.java`)

El servlet que constituye la implementación de las funciones principales.

Destacar la importancia del *ServletContext*, que es el que permite acceder a la información sobre el entorno. será necesario para poder ejecutar correctamente las funciones solicitadas en el enunciado. En el momento en el que el usuario interactúa con los botones de las funcionalidades principales (Insertar Jugador, Eliminar y Editar), estas acuden al `doGet`. Allí mediante el uso de un "switch" son redirigidas a su función correspondiente y se muestra el resultado de la operación usando `RequestDispacher.forward()` hacia la JSP. A continuación se analizan con mayor profundidad.

2.2.1 InsertJugador

Como su nombre indica, esta función inserta nuevos jugadores a la base de datos como filas a las tablas de SQL. Para la implementación con JPA, se hace uso de los recursos EntityManager (para buscar, editar y guardar los valores de los jugadores implementados) y UserTransaction (para conectar con la base de datos). En cambio, para la implementación de JDBC y su inserción dentro de las tablas, las *queries* se usan de manera directa como consultas SQL.

2.2.2 DeleteJugador

Encargada de eliminar los jugadores ya insertados en las tablas de la base de datos. Haciendo uso de los recursos *EntityManager* (para buscar, editar y guardar los valores de los jugadores implementados) y UserTransaction (para conectar con la base de datos). En cambio, para la implementación de JDBC y a la hora de se usan de manera directa las consultas de SQL. En cambio, para la implementación de JDBC y su edición dentro de las tablas, las querys se usan de manera directa las consultas de SQL.

2.2.3 EditJugador

Encargada de editar la información de cada jugador tras haber comprobado si el jugador ya existe. Haciendo uso de los recursos EntityManager (para buscar, editar y guardar los valores de los jugadores implementados) y UserTransaction (para conectar con la base de datos). En cambio, para la implementación de JDBC y a la hora de se usan de manera directa las consultas de SQL. En cambio, para la implementación de JDBC y su eliminación dentro de las tablas, las querys se usan de manera directa las consultas de SQL.

Recordar que nuestro proyecto SQL está principalmente definido por dos tablas que son jugador y posición, con una realcion n-1.

2.3 Utils (`utils/`)

Contiene funcionalidades genéricas que pueden ser utilizadas una o varias veces en un proyecto y que es mejor encapsular para la simplicidad y escalado del proyecto.

2.3.1 ValidadorDNI (`ValidadorDNI.java`)

Archivo encargado de validar un valor que se introduce como DNI comprobando que la letra es correcta y coincide con el número correspondiente o sea un dni que no exista.
Para esta funcionalidad se utiliza una expresion regular (regex).

2.4 db_create.sql (`db_create.sql`)

Como se mencionó en la introducción se trata de un archivo relacionado a la creación de la base de datos, y como se indica en su cabecera se trata de un archivo generado con MySQL Workbench que es la herramienta con la que estamos gestionando la base de datos.

2.5 WebApp (`webapp/`)

Dividido en los diferentes **JavaServer Pages** y la carpeta de hojas de estilo posible.

2.5.1 `Home.jsp`

La página principal donde el usuario accede a la aplicación web. En ella ya se presenta la posibilidad de añadir jugadores a la plantilla de fútbol en cuestión. Además, en ella misma será donde aparecerán las filas implementadas en las tablas de la base de datos de sql con cada uno de los jugadores y los datos pertinentes de cada uno de ellos según se vayan añadiendo al equipo, acompañado de una lista visible de los jugadores que quedan por agregar de esa plantilla con los totales permitidos por cada posición. Junto a estos datos de cada jugador, aparecerá el botón para poder editar los datos de cada uno de ellos a gusto del usuario a la vez que poder eliminarlos de la base de datos. Todo el resto de las páginas redirigen a esta al ser el punto de partida de la funcionalidad principal, siendo esta la de añadir nuevos jugadores.

2.5.2 `InsertarJugadorForm.jsp`

La página desde la cual se pueden agregar los nuevos jugadores a la plantilla. Sigue una estructura basada en los datos que hay que incluir de cada uno de los jugadores para su correcto almacenamiento en formato fila dentro de la tabla de jugadores de SQL, presentando los campos nombre, apellidos, DNI (con formato modificado), alias y posición, donde exclusivamente se pueden seleccionar delantero, defensa, medio y portero. Archivo JSP que despliega en la página web el formulario necesario para añadir a aun jugador, dándo al ususario capacidad de acceso y modificación sobre la base de datos al igual que la funciónalidad editar.

2.5.3 `EditarJugadorForm.jsp`

Archivo JSP que despliega en la página web el formulario necesario para editar a aun jugador, dándo al ususario capacidad de acceso y modificación sobre la base de datos. Esta funcionalidad es tambien útil de cara a gestión de errores, no tanto de código si no puramente humanos, como inputs erroneos o inexactos que con esta funcionalidad pueden ser modificados.

2.5.4 `Header.jsp`

Es un JSP que representa la cabecera de página de en este caso el Atlético de Madrid pero cuyo objetivo es ser "generalizado" para poder ser reutilizado en distintas páginas (ya sea de otros equipos o de distintas funcionalidades dentro de un mismo equipo).

2.5.5 `Footer.jsp`

Al igual que el archivo anterior un JSP que representa el footer de página y cuyo objetivo es ser "generalizado" para poder ser reutilizado en distintas páginas.

2.5.6 `Error.jsp`

Esta página, como su propio nombre indica, es la destinada a mostrar errores y permitir lanzar los mensajes de error y excepciones pertinentes. Además, se incluye un botón que permite volver a la página principal para poder seguir navegando si es que ya estabas dentro de la propia aplicación cuando el error dió lugar.

3 Conclusiones

La realización de esta práctica ha supuesto un reto para el equipo y en ciertas ocasiones una ocasión perfecta para aprender a enfrentarnos a problemas reales que pueden aparecer en el desarrollo de una aplicación real de misma naturaleza. Creemos que el hecho de incluir una base de datos real y no un simple string nos ha ayudado a entender correctamente la forma en la que se trabaja en el mundo laboral con herramientas como MySQL workbench o los servidores Payara entre otras muchas cosas. La realización de esta segunda parte de la primera práctica ha supuesto más problemas que la primera parte que la compone. La mayor dificultad encontrada puede haber sido el ser consciente de las conexiones entre los formularios para insertar y editar nuevos jugadores junto con los *Front Controllers* y tener control de las consultas sql con las que se trabajaban.

Cabe destacar que el uso de archivos JDBC y JPA aun aumentando la dificultad de la práctica hace de nuestra aplicación un proyecto más consistente escalable y organizado y nos a proporcianado conocimientos muy útiles para el desarrollo de este tipo de herramientas. JPA tiene una implementación más sencilla y menos costosa frente a JDBC, que al ejecutar y tener una gestión de las propias "querys" manera precisa es más tedioso.

Destacar la dificultad añadida de las coincidencias con otros exámenes y proyectos que ha podido entorpecer el hecho de coincidir el equipo entero de desarrollo que en cambio ha sido capaz de sacar adelante el proyecto dividiendo el trabajo de forma eficiente y estando en constante contacto.