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 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).

La estructura principal que sigue el proyecto (carpeta `src/main/`) 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. Por otro lado el **WebApp** es donde aparecen los archivos JavaServer Pages (los que permiten dar el formato HTML a nuestra web) junto a una hoja de estilos (`styles/style.css`) para poder editar fuentes y todo lo relacionado con la apariencia de la aplicación web en posibles actualizaciones.

2 Java

Dividido en tres carpetas principales (`beans/`, `servlets/` y `utils/`) con sus respectivos archivos y divisiones. Es la carpeta madre de todos los archivos Java del proyecto.

2.1 Beans (`beans/`)

Incluye los objetos que serán usados.

2.1.1 Jugador (`Jugador.java`)

Se trata del archivo donde se define el objeto `Jugador` y se comprueba si los valores intruducidos al constructor son correctos (no hay campos vacíos) y se llama al validador del DNI para comprobar si este es correcto.
Los atributos, todos strings, son `nombre`, `apellidos`, `dni`, `alias` y `posición`.

2.1.2 Posiciones (`Posiciones.java`)

Se trata del archivo donde se define el objeto `Posiciones` y se definen estas mismas.
El archivo comienza definiendo el numero inicial en cada posición que obviamente es 0 ya que la base de datos está vacia, y el número máximo de jugadores por posición.
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 negativo).
Cuenta con un *Enum* `Posicion` el cual se encarga de validar que las posiciones sean las correctas.
Los atributos, todos ints, son `numDelanteros`, `numDefensas`, `numMedios`, `numPorteros`, `maxDelanteros`, `maxDefensas` y `maxMedios`, `maxPorteros`.

2.2 Servlets (`servlets/`)

Carpeta conteniente de todos los archivos Java en relación a los servlets que nuestra aplicación utiliza.
Destacar la importancia del *ServletContext*, que el es encargado de mantener la consistencia entre todos los servlets y es el encargado de almacenar datos cuando por ejemplo un jugador se edita entre otras. Este servlet es la interfaz entre todos ellos y el que permite el correcto funcionamiento de los servicios que proveen.

2.2.1 AñadirJugadorServlet (`AñadirJugadorServlet.java`)

Archivo en el que se define el servet encargado de añadir un jugador a la base de datos. Este servlet actualiza los valores de las posiciones y introduce a la base de datos la información introducida por el usuario.
Para realizar este cometido se añade a la base de datos utilizando el método `append()` (ya que es un str), se sobrescribe en el *ServletContext* con `setAttribute()`, por último redirigimos al usuario con la correspondiente gestión de errores.
Tanto este como el formulario de editar jugador hacen uso de la request tipo POST.

2.2.2 EditarJugadorServlet (`EditarJugadorServlet.java`)

Archivo en el que se define el servet encargado de modificar la información de un jugador a la base de datos. Este servlet modifica en la base de datos la información introducida por el usuario.
Con el índice del jugador (que es al fin y al cabo la forma de localizarlo) se actualiza en la base de datos utilizando el metodo `set()`,se sobrescribe en el *ServletContext* con `setAttribute()`, por último redirigimos al usuario a la pagina principal (`index.jsp`) con la correspondiente gestión de errores.
Tanto este como el formulario de añadir jugador hacen uso de la request tipo POST.

2.2.3 EliminarJugadorServlet (`EliminarJugadorServlet.java`)

Archivo en el que se define el servet encargado de borrar un jugador de la base de datos. Este servlet actualiza los valores de las posiciones y elimina de la base de datos la información introducida por el usuario.
Este formulario hace uso de la request tipo GET.

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 CreateDBServlet (`CreateDBServlet.java`)

Archivo encargado de la gestión que implica crear la base de datos de cero. Importa numerosas funcionalidades de `javax.servlet` ya que muchas de las funciones de estos están estrechamente relacionadas a la base de datos.
Es relevante destacar que en este caso la base de datos se trata simplemente de un string en el que almacenamos los datos y no una base de datos al uso por tanto siempre que se mencione la misma nos referiremos a esta forma de almacenamiento.

2.3.2 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).

3 WebApp (`webapp/`)

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

3.1 `index.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 aparecerá el array 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.

3.2 `AñadirJugador.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 el array, presentando los campos nombre, apellidos, DNI (con formato modificado), alias y posición, donde exclusivamente se pueden seleccionar delantero, defensa, medio y portero.
Su conexión con `AñadirJugadorServlet/` y la *Bean* `Jugador.java` permite mostrar por pantalla los errores pertinentes si no se cumple uno de los requisitos en cuestión dentro

del formulario. Además, existe la opción de cancelar la operación y así volver a la página principal sin haber realizado ninguna agreagación a la base de datos.

3.3 `EditarJugador.jsp`

La página que permite al usuario modificar al jugador que se haya seleccionado desde el array principal del index.
Presenta la misma estructura que el formulario para añadir a los jugadores con cada uno de sus subapartados y el botón para poder guardar los cambios realizados. Ésta es una de las funcionalidades que no son estrictamente necesarias o solicitadas en el enunciado de la práctica, pero para futuros trabajos y actualizaciones sobre esta aplicación web se entiende que será necesaria y de ahí su implementación extraordinaria.
Además, existe la opción de cancelar la operación y así volver a la página principal sin haber realizado ninguna modificación en los jugadores existentes en la base de datos.

3.4 `ErrorPage.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.

4 Conclusiones

La realización de esta práctica a pesar de no suponer una gran dificultad en primera instancia ha sido muy constructiva, ya que es el primer contacto directo con un proyecto de este tipo y en el cual se tratan muchas cosas nuevas (especialmente en la conexión a servidores como de java) al igual que se repasan otras tantas como el trabajo con html y las hojas de estilos. Siendo estas ultimas algo fundamental ya que se trata de la cara del producto la cara del producto y la aplicación web que se quiere dar a conocer.

Ha sido cuanto menos interesante la utilización primeriza de Eclipse y Payara para establecer las bases en las que el proyecto se desarrolla (servidores, HTML, CSS...) ya que nos da un enfoque más profesional de la forma de trabajar del mercado laboral.

Como conclusiones finales, cabe resaltar el interés que tenemos sobre como se pueden unir este tipo de recursos y herramientas con el resto que se han comentado en las clases para, de esta manera, acabar con un trabajo de mucha mayor escala y aplicable a problemas reales.