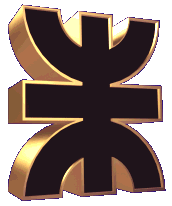
|  |
| --- |
| **Nombre y Apellido** |
| Cappellini Darío |
| Carella Germán |
| Centurion Damian |
| Gonzalez Santiago |

|  |  |
| --- | --- |
| **Fecha De Presentación:** | 24/09/2014 |
| **Fecha de Devolución:** |  |
| **Calificación** |  |
| **Firma Profesor** |  |



**GRUPO N° 7**

TP Anual: 4ta Entrega

***Diseño de Sistemas 2014***

***Miércoles mañana***

**Introducción**

Para esta entrega se solicitó diseñar la interfaz de usuario para tres casos de uso: generar equipos, buscar jugadores y visualizar los datos de un jugador.

Elegimos una interfaz de usuario web utilizando el framework Wicket para aplicaciones web. Esta aplicación tendría una arquitectura distribuida, mediante la cual el usuario que organiza los partidos puede ingresar desde cualquier browser a la aplicación para realizar sus tareas. Queda claro que, para una arquitectura de este estilo, es necesario un mecanismo de seguridad (por ejemplo que el usuario que organiza los partidos se identifique con nombre de usuario y contraseña como alguien autorizado para hacer eso) pero no se implementó ya que está fuera del alcance del trabajo práctico.

Elegimos una aplicación web porque nos pareció una experiencia útil considerando la prevalencia que tiene esta configuración en el mercado actual.

**Arquitectura**

Para los casos de uso “generar equipos” y “buscar jugadores” aplicamos el patrón MMVC. Cada uno de estos casos de uso tiene un objeto de dominio intermedio Application Model que trata con particularidades de la aplicación pero no depende de la tecnología de la UI:

* Para generar equipos, tenemos un GeneradorDeEquipos que se encarga de hablar con el partido para generar los equipos, y de crear los criterios de ordenamiento y de selección necesarios para llevar a cabo esta operación.
* Para buscar jugadores, tenemos un BuscadorDeJugadores que maneja los criterios de búsqueda y la lista de resultados.

El caso de uso que se refiere a la visualización de los casos del jugador es más sencillo, ya que no hace ninguna operación especial sino que sólo muestra datos. En este caso no hay un objeto de dominio intermedio, sino que los componentes visuales bindean directamente con el objeto jugador.

**Reutilización de código**

La consigna destaca los siguientes requerimientos para la interfaz de usuario, que deben ser resueltos sin repetir código:

* Los jugadores que tienen más de puntos de hándicap se deben mostrar en azul en todos los casos de uso.
* Al hacer click en un jugador debo poder acceder a la pantalla con su información. Esta pantalla debe estar definida en un solo lugar.

Para resolver esto creamos una clase LinkJugador que concentra ese comportamiento. Tenemos definida la página que muestra la información del jugador que recibe al jugador por parámetro, y por lo tanto está definida una sola vez. LinkJugador es un componente visual que hereda de XLink. Al ser creado recibe como parámetro el Wicket ID que necesita el XLink, el jugador al que se refiere, y la página que lo está creando.

Con estos datos setea su onClick, que tiene como resultado abrir una nueva página de información del jugador, utilizando el jugador como parámetro para poder armar la página correcta. También se le pasa a la nueva página la página que creó el XLink, para que sepa a dónde regresar con su botón de volver. Finalmente, como el XLink conoce al jugador, le pregunta su nivel de hándicap y si es mayor a 8 se pinta de azul.

Al utilizar el LinkJugador como componente visual en lugar de Label en todos los casos de uso que muestran jugadores, hacemos que cada mención a un jugador sea un hipervínculo que linkea a los datos de ese jugador y que aparece en azul cuando es necesario.