

DEPARTAMENT D'INFORMÀTICA
UNIVERSITAT JAUME I

II31 PROYECTOS INFORMÁTICOS

INGENIERÍA INFORMÁTICA
Curso 2007--2008

Memoria Técnica del Proyecto

RTStatBasket - Gestión y retransmisión online de estadísticas de partidos de baloncesto a tiempo real

Proyecto presentado por el Alumno

Raúl Martín Félez

Dirigido por ***Raúl Montoliu Colás***

Castellón, a 30 de Enero de 2008

Resumen

Este proyecto trata de generar un sistema multiplataforma que permita la introducción de las acciones que suceden en los partidos de baloncesto en tiempo real y a pie de cancha. El sistema debe ser capaz a su vez de extraer las estadísticas completas del partido, de cada jugador y de los equipos involucrados en el partido a partir de esas acciones, y permitir visualizarlas a tiempo real tanto en el propio sistema a pie de cancha como de forma online, es decir, a través de una web para su rápida difusión a un mayor conjunto de personas. El formato web contendrá además de las estadísticas obtenidas una narración fácilmente legible de las acciones que van sucediendo a lo largo del partido.

El sistema debe proveer una forma de introducir las acciones ágil, cómoda, intuitiva y eficaz para evitar que quien está utilizando el sistema pierda demasiado tiempo el contacto visual con el terreno de juego. Para ello, se genera un nuevo lenguaje que consiste en un conjunto de comandos sencillos e intuitivos cuya combinación permite introducir a través del teclado cualquier acción que suceda en un partido. Se dota al usuario de la capacidad para introducir varias acciones simultáneamente, situación muy frecuente en un partido de baloncesto, pero que los sistemas similares que se utilizan en la actualidad no contemplan.

Palabras Clave

Baloncesto, Estadísticas, Tiempo real, Online

INDICE

CAPÍTULO 1	1
1. Introducción.....	2
1.1 <i>Objetivos</i>	14
1.2 <i>Descripción general del entorno</i>	15
1.3 <i>Planificación</i>	16
1.4 <i>Organización del documento</i>	24
CAPÍTULO 2	26
2. Descripción del proyecto	27
2.1 <i>Análisis</i>	27
2.1.1 Arquitectura del sistema	27
2.1.2 Definición de usuarios y roles	32
2.1.3 Tecnologías empleadas.....	33
2.2 <i>Diseño</i>	42
2.2.1 Definición del nuevo lenguaje de comandos	42
2.2.2 Diseño de las clases	55
2.2.3 Diseño de la base de datos.....	64
2.2.4 Diseño de las interfaces de usuario	67
CAPÍTULO 3	74
3. Experimentación: pruebas y resultados.....	75
CAPÍTULO 4	77
4. Conclusiones	78
CAPÍTULO 5	82
5. Posibles mejoras y ampliaciones	83
CAPÍTULO 6	85
6. Referencias	86
ANEXO 1	89
Manual de usuario	90
<i>Aplicación de escritorio</i>	90
Gestión acciones.....	91
Estadísticas jugadores	97
Estadísticas equipos	99
<i>Aplicación web</i>	101
ANEXO 2	105
Procedimiento de instalación.....	106
Java JDK.....	107
XAMPP básico	109
Add-ons Tomcat para XAMPP	118

INDICE DE FIGURAS

CAPÍTULO 1	1
• Figura 1: Ejemplo de planilla tradicional para tomar estadísticas de los partidos de baloncesto	7
• Figura 2: Pantalla principal de la aplicación de escritorio desde la que se introducen las acciones del partido	8
• Figura 3: Pantalla de la aplicación de escritorio desde la que se visualizan las estadísticas completas de los jugadores	9
• Figura 4: Pantalla de la aplicación de escritorio desde la que se visualizan las estadísticas completas de los equipos	10
• Figura 5: Pantalla principal de la aplicación web donde se visualiza la retransmisión de las acciones que están sucediendo a lo largo del partido.	11
• Figura 6: Pantalla de la aplicación web desde la que se visualizan las estadísticas completas de los jugadores	12
• Figura 7: Pantalla de la aplicación web desde la que se visualizan las estadísticas completas de los equipos	13
• Figura 8: Calendario del proyecto	17
• Figura 9: Diagrama de Gantt (11 Julio – 30 Agosto)	20
• Figura 10: Diagrama de Gantt (30 Agosto – 15 Octubre)	21
• Figura 11: Diagrama de Gantt (15 Octubre – 3 Diciembre)	22
CAPÍTULO 2	26
• Figura 12: Arquitectura Cliente- Servidor de la aplicación de escritorio	29
• Figura 13: El modelo tradicional para las aplicaciones Web (izq.) comparado con el modelo de AJAX (der.).	30
• Figura 14: El patrón de interacción sincrónica de una aplicación Web tradicional (arriba) comparada con el patrón asincrónico de una aplicación AJAX (abajo).....	31
• Figura 15: Arquitectura de la aplicación web.....	32
• Figura 16: Conexión de una aplicación Java a una base de datos mediante JDBC	35
• Figura 17: MDD (Máquina Discriminadora Determinista)	46
• Figura 18: Relación entre los componentes del patrón MVC.....	55
• Figura 19: Diagrama de clases de la aplicación de escritorio	62
• Figura 20: Diagrama de clases de la aplicación web	63
• Figura 21: Boceto inicial de interfaz para la aplicación de escritorio.....	67
• Figura 22: Diseño final de la aplicación de escritorio (Pestaña 1)	70
• Figura 23: Diseño final de la aplicación de escritorio (Pestaña 2)	71
• Figura 24: Diseño final de la aplicación de escritorio (Pestaña 3)	71
• Figura 25: Diseño final de la aplicación web (Pestaña 1).....	72
• Figura 26: Diseño final de la aplicación web (Pestaña 2).....	73
• Figura 27: Diseño final de la aplicación web (Pestaña 3).....	73

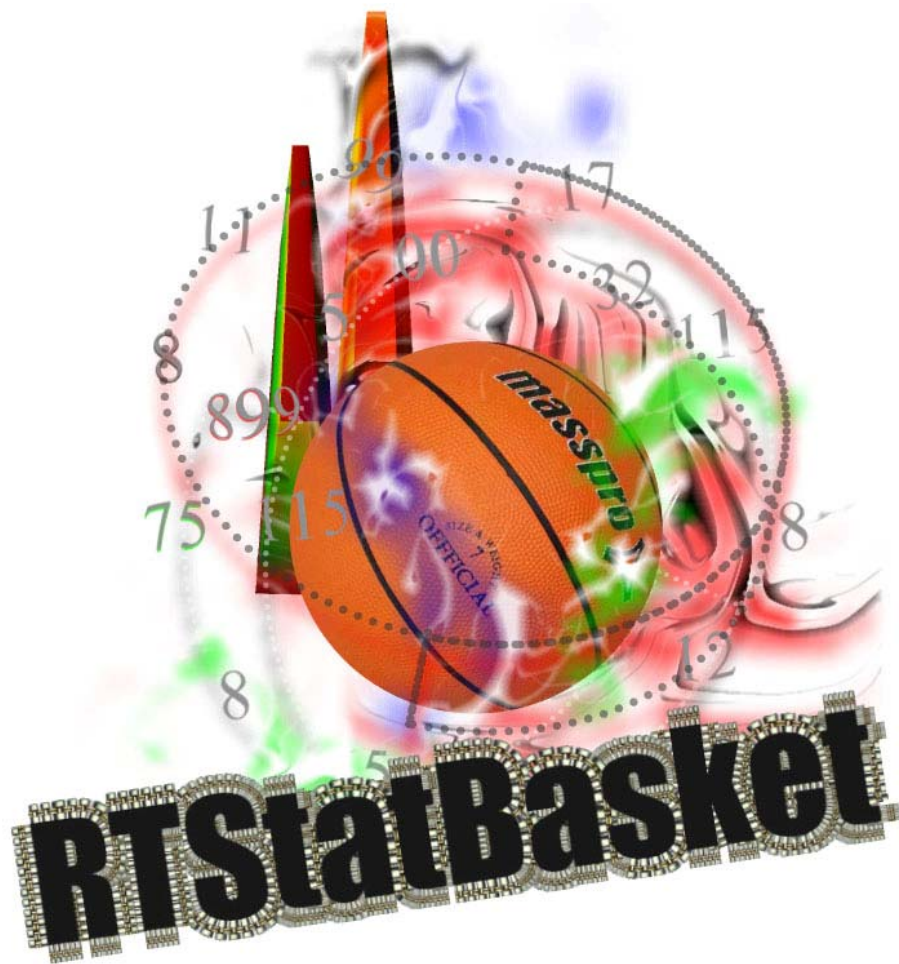
ANEXO 1 89

- **Figura 28:** Pantalla de conexión al servidor de bases de datos de la Aplicación de escritorio 90
- **Figura 29:** Identificación de las principales zonas de la 1ª pestaña de la aplicación de escritorio 91
- **Figura 30:** Ejemplo de envío de varias acciones simultáneas intercalando mayúsculas y minúsculas..... 93
- **Figura 31:** Ejemplo de detección de un error léxico..... 94
- **Figura 32:** Ejemplo de detección de error sintáctico 95
- **Figura 33:** Ejemplo de detección de un error semántico..... 96
- **Figura 34:** Ejemplo del estado de la 1ª pestaña de la aplicación de escritorio en el tercer cuarto de un partido..... 97
- **Figura 35:** Ejemplo del estado de la 2ª pestaña de la aplicación de escritorio en el tercer cuarto de un partido..... 98
- **Figura 36:** Ejemplo del estado de la 3ª pestaña de la aplicación de escritorio en el tercer cuarto de un partido..... 99
- **Figura 37:** Ejemplo del estado de la 1ª pestaña de la aplicación de escritorio al finalizar un partido..... 100
- **Figura 38:** Ejemplo del estado de la 1ª pestaña de la aplicación web en el tercer cuarto de un partido 101
- **Figura 39:** Ejemplo del estado de la 2ª pestaña de la aplicación web en el tercer cuarto de un partido 103
- **Figura 40:** Ejemplo del estado de la 3ª pestaña de la aplicación web en el tercer cuarto de un partido 104

ANEXO 2 105

- **Figura 41:** Instalación JDK Java (Paso 1) 108
- **Figura 42:** Instalación JDK Java (Paso 2)..... 108
- **Figura 43:** Instalación JDK Java (Paso 3)..... 108
- **Figura 44:** Instalación JDK Java (Paso 4)..... 109
- **Figura 45:** Instalación JDK Java (Paso 5)..... 109
- **Figura 46:** Proceso de instalación de XAMPP básico (Paso 1) 110
- **Figura 47:** Proceso de instalación de XAMPP básico (Paso 2) 111
- **Figura 48:** Proceso de instalación de XAMPP básico (Paso 3) 111
- **Figura 49:** Proceso de instalación de XAMPP básico (Paso 4) 112
- **Figura 50:** Proceso de instalación de XAMPP básico (Paso 5) 112
- **Figura 51:** Panel de control de XAMPP básico 114
- **Figura 52:** Pantalla de selección de idioma de XAMPP 114
- **Figura 53:** Pantalla de bienvenida a la configuración de XAMPP 115
- **Figura 54:** Pantalla de configuración de MySQL (phpMyAdmin) 115
- **Figura 55:** Pantalla para importar un archivo en MySQL 116
- **Figura 56:** Pantalla de visualización de la base de datos Basket..... 116
- **Figura 57:** Pantalla de configuración de privilegios y permisos de los usuarios de MySQL..... 117
- **Figura 58:** Modificación de la contraseña del usuario root 117
- **Figura 59:** Proceso de instalación de la extensión Tomcat para XAMPP (Paso 1) 118
- **Figura 60:** Proceso de instalación de la extensión Tomcat para XAMPP (Paso 2) 119
- **Figura 61:** Proceso de instalación de la extensión Tomcat para XAMPP (Paso 3) 119
- **Figura 62:** Proceso de instalación de la extensión Tomcat para XAMPP (Paso 4) 120
- **Figura 63:** Inicialización del servidor Tomcat desde el menú de Inicio..... 119

*A mi padre, que desde el cielo va a ver como su esfuerzo
me ha permitido llegar a ser Ingeniero.*



Introducción

1.1 Objetivos	14
1.2 Descripción general del entorno.....	15
1.3 Planificación	16
1.4 Organización del documento	24

1. INTRODUCCIÓN

Este proyecto trata sobre la realización de un sistema que permita la gestión de las estadísticas de partidos de baloncesto en tiempo real, y a su vez permita visualizar esas estadísticas de forma online, es decir, a través de la web. Se le ha asignado el nombre **"RTStatBasket"**, siglas correspondientes a "*Real Time Stat Basket*", que se podría traducir como Estadísticas de Baloncesto a Tiempo Real.

En primer lugar se realiza una aproximación para conocer "a grosso modo" los conceptos claves de este proyecto.

¿Qué es el baloncesto?

El **baloncesto** (del inglés *basketball*) es un deporte de equipo que consiste básicamente en introducir una pelota en un aro, del que cuelga una red, lo que le da un aspecto de cesto. En algunas regiones se llama *básquet*, al castellanizar el término inglés para la palabra cesto. Se juega con dos equipos de cinco personas, durante 4 periodos o cuartos de 10 minutos cada uno. Al finalizar el segundo cuarto, se realiza un descanso, normalmente de 10 a 20 minutos según la reglamentación propia del campeonato al cual el partido pertenezca [1].

Durante cada cuarto se suceden una serie de acciones como son los distintos tipos de canasta (que un jugador introduzca el balón en el aro contrario) que puede ser de 1, 2 o 3 puntos, tiros fallados de 1, 2 o 3 puntos, rebotes, asistencias, faltas, etc. que se definirán con más detalle más adelante.

¿Qué es la Estadística?

La **Estadística** es una ciencia matemática referida a la recolección, análisis, interpretación y presentación de datos. Tiene implicación en una amplia variedad de disciplinas académicas desde las ciencias sociales y físicas hasta las de humanidades, tanto como en el comercio, el gobierno, la medicina y la industria, y comenzando con el rol fundamental que juegan las estadísticas en los deportes en general y en el baloncesto en particular, temática que nos ocupa en este proyecto [2].

¿Qué relación tienen el baloncesto y la estadística?

Entre la numerosa información que se debe gestionar en un partido de baloncesto, la más destacada y visible para los aficionados y profesionales es la información estadística, es decir, la información relativa a todos los números que se producen durante un partido de baloncesto motivados por las acciones que van sucediendo (resultados, anotadores, faltas personales, rebotes, tapones...). Se trata de cifras que interesan mucho a los medios de comunicación y a los aficionados, pero que también tienen un gran valor para los entrenadores, que las emplean para conocer mejor a sus rivales o a sus futuros fichajes, y por supuesto para descubrir las virtudes y defectos de sus propios jugadores de forma que puedan fomentar esas virtudes, corregir esos defectos y poder atacar los defectos de los jugadores rivales.

¿Por qué es importante el uso de la estadística?

- Aporta conocimiento sobre la situación actual (estadísticas del partido: resultados, anotadores, faltas personales, rebotes, tapones, etc. en cada momento que se consulten).
- Permite detectar posibles necesidades y deficiencias (Permite conocer mejor a los rivales o futuros fichajes, y descubrir las virtudes y defectos de los propios jugadores).

- Apoya la toma de decisiones (Permite fomentar las virtudes de los propios jugadores, corregir sus defectos y poder atacar los defectos de los jugadores rivales desde el conocimiento adquirido gracias a la estadística).
- Influye en la motivación de los jugadores.

¿Cómo se recogen las estadísticas en los partidos de baloncesto?

En las grandes ligas de baloncesto o campeonatos a nivel continental o mundial (NBA, ACB, FIBA, etc.), se poseen medios audiovisuales y sistemas informáticos que permiten obtener las estadísticas de los partidos fácilmente, de forma correcta y cómoda. Una de las aplicaciones informáticas que suelen utilizar para ello es MBT SmartStats [7].

Sin embargo, cuando se trata de equipos pequeños o de equipos de cantera, únicamente se dispone de las actas de los partidos y de las estadísticas que recoge para cada equipo una persona correspondiente al cuerpo técnico denominada *estadígrafo*. Los estadígrafos deben tener la habilidad de sumar, restar y dividir mentalmente (no hay tiempo para calculadoras) y por otra parte también deben tener capacidad de escribir números y letras en forma muy legible, ya que todos los componentes del cuerpo técnico leen su trabajo. También deben tener las habilidades de controlar, verificar y examinar nuevamente su trabajo en todo momento, comunicándose con el resto del cuerpo técnico a efectos de evitar errores, ya que no existe una persona que pueda registrar estadísticas de forma estrictamente correcta. Por tanto, habitualmente pese a su trabajo, siempre se deben esperar los resultados del análisis a posteriori por medio del vídeo para tener las estadísticas reales y exactas de cada partido.

Objetivo principal del proyecto

Con este proyecto, se pretende crear una herramienta informática que permita al estadígrafo tomar a pie de cancha estadísticas de forma sencilla, intuitiva, ágil, sin realizar ninguna operación matemática, y sin necesidad de esperar a obtener los resultados del análisis por vídeo, ya que con este sistema tendrá total garantía de que las estadísticas han sido bien tomadas y sin ningún error de cálculo. Además, estas estadísticas se publicarán a tiempo real a través de la web, por lo que no sólo el estadígrafo y el cuerpo técnico tendrán acceso a las mismas durante la disputa del partido, sino que cualquier medio de comunicación, aficionado, familiar, amigo, etc. las podrá conocer a tiempo real de forma online. Así pues, las tradicionales planillas de papel (ver *Figura 1:*) quedarán relegadas por este sistema que ofrece al cuerpo técnico una manera fácil y eficaz de recopilar toda la información que necesitan en formato digital.

Estado del arte

La idea de desarrollar una herramienta de este tipo no es nueva, y como ya se ha comentado anteriormente, las grandes ligas y campeonatos a nivel de selecciones ya poseen sistemas capaces de realizar esta adquisición de estadísticas a tiempo real junto con su exportación a través de la web. Entonces, ¿cuál es la novedad que aporta este proyecto? Pues algo tan básico pero a su vez extremadamente importante en este contexto como es la introducción de los datos en el sistema.

Algunas de las aplicaciones que actualmente ya realizan el objetivo de este proyecto son MBT SmartStat [7] que permite gestionar las estadísticas a pie de campo y a su vez exportarlas a través de la web, Basketgest 3.0. [8], Planilla 3000 [9], Kstatistics Basket [10], Turbo Stats 7.0 for Basketball [11], y StatsNow [12] que sólo permiten gestionar las estadísticas a tiempo real pero no exportan las mismas a través de la web en tiempo real. Todas estas aplicaciones, salvo BasketGest y Planilla 3000, poseen versiones especializadas para dispositivos móviles.

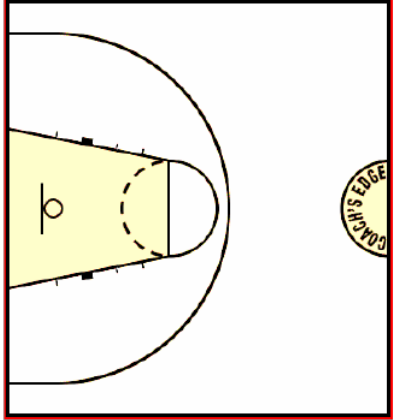
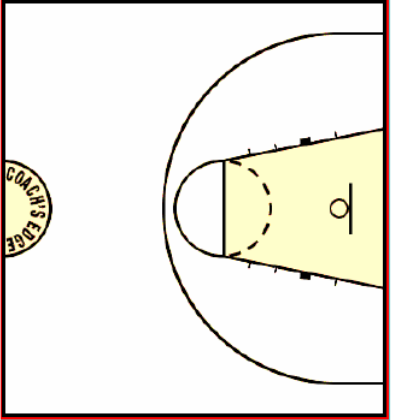
Independientemente de que las anteriores aplicaciones exporten o no las estadísticas que han calculado a través de la web (uno de los objetivos de este proyecto), todas las aplicaciones que se acaban de nombrar poseen un mismo problema, que es la forma de introducir los datos. Los datos se introducen mediante el ratón, y si la persona que introduce los datos a su vez debe estar observando el partido a pie de pista, su trabajo resulta incómodo e ineficaz, ya que pierde constantemente la visión del partido si se pone a introducir datos. Así pues, el objetivo principal del proyecto es diseñar una forma de introducir datos cómoda, ágil, sencilla e intuitiva, que permita a la persona que introduce los datos perder el menor tiempo posible el contacto visual con el terreno de juego.

Para alcanzar este objetivo se plantearon varias alternativas, unas más viables que otras:

- Obtener las acciones del partido mediante análisis de vídeo a tiempo real utilizando técnicas de visión por ordenador. Hoy en día esta alternativa es inviable porque no existen técnicas ni tecnología capaz de tratar con varios tipos de iluminación, tres árbitros interactuando, etc.
- Obtener las acciones del partido mediante reconocimiento de voz. Sería una alternativa viable, pero también muy costosa y de difícil aplicación.
- Introducir las acciones del partido manualmente utilizando el ratón. Es una alternativa viable que utilizan la mayoría de los sistemas de estas características.
- Introducir las acciones del partido manualmente utilizando el teclado. Es la alternativa viable que se empleará en este proyecto. Consiste básicamente en crear un conjunto de comandos sencillos e intuitivos que permitan de forma cómoda y rápida, y sin tener que mirar el ordenador apenas introducir las acciones que suceden a lo largo del partido. Así se consigue que la persona que introduce los datos apenas tenga que perder el contacto visual con el partido.



Lugar: _____ vs _____ Categoría: _____
 Fecha: _____ Horas: _____ Competición: _____

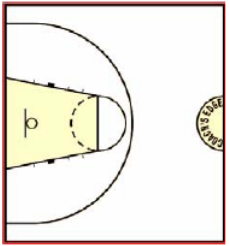
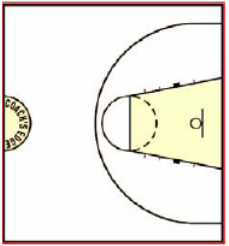
ENTR.	TIEMPO	TIROS LIBRES	TIEMPO	ENTR.
				
	1º cuarto	2º cuarto	3º cuarto	4º cuarto y p. extras

ERRORES DEFENSIVOS					OBSERVACIONES
REBOTES	DEFENSIVOS	1º	2º	3º	
	OFENSIVOS	1º	2º	3º	
BALONES	PERDIDOS	1º	2º	3º	
	RECUPERADOS	1º	2º	3º	
	ASISTENCIAS	1º	2º	3º	
	TAPONES	1º	2º	3º	
	LUCHAS	1º	2º	3º	
FALTAS RECIBIDAS					
VIOLACIONES	INDIVIDUALES	1º	2º	3º	
	8" 24" 5"	1º	2º	3º	
	3 SEGUNDOS	1º	2º	3º	

JUGADORA	SI	Nº	TIEMPO DE JUEGO				
		4					
		5					
		6					
		7					
		8					
		9					
		10					
		11					
		12					
		13					
		14					
		15					
		16					

FALTAS PROPIAS					FALTAS CONTRARIAS				
4					4				
5					5				
6					6				
7					7				
8					8				
9					9				
10					10				
11					11				
12					12				
13					13				
14					14				
15					15				
B					B				

PARCIALES 1 TIEMPO	PARCIALES 2 TIEMPO	TANTEO FINAL
/	/	/

ENTR.	EQUIPO CONTRARIO				ENTR.
	TIEMPO	T.L.	T.L.	TIEMPO	
					

TIEMPOS MUERTOS					TIEMPOS MUERTOS				

REBOTES DEFENSIVOS					ARBITROS				
REBOTES OFENSIVOS					ANOTADORES				
BALONES PERDIDOS					ESTADÍSTICOS				
BALONES RECUPERADOS									

Anotación Propia																																		Anotación contraria:																																	
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100				

Figura 1: Ejemplo de planilla tradicional para tomar estadísticas de los partidos de baloncesto

Presentación previa de la aplicación

A forma de presentación visual del trabajo realizado en este proyecto, se muestran a continuación algunas capturas de pantalla del resultado final de la aplicación.

En la *Figura 2*: se muestra la interfaz principal de la aplicación de escritorio. Desde ella se introducen las acciones que suceden a lo largo del partido. Además muestra las últimas acciones que se han enviado, un histórico de acciones del partido, la puntuación de los cuartos y la puntuación en ese instante del partido. Para cada equipo se puede observar cuál es el quinteto que está jugando en ese instante, cuál es el banquillo en ese instante y cuál es la situación de cada jugador respecto a puntos anotados y faltas personales realizadas. También se muestra la información del partido respecto a la fecha y lugar en que se está disputando.

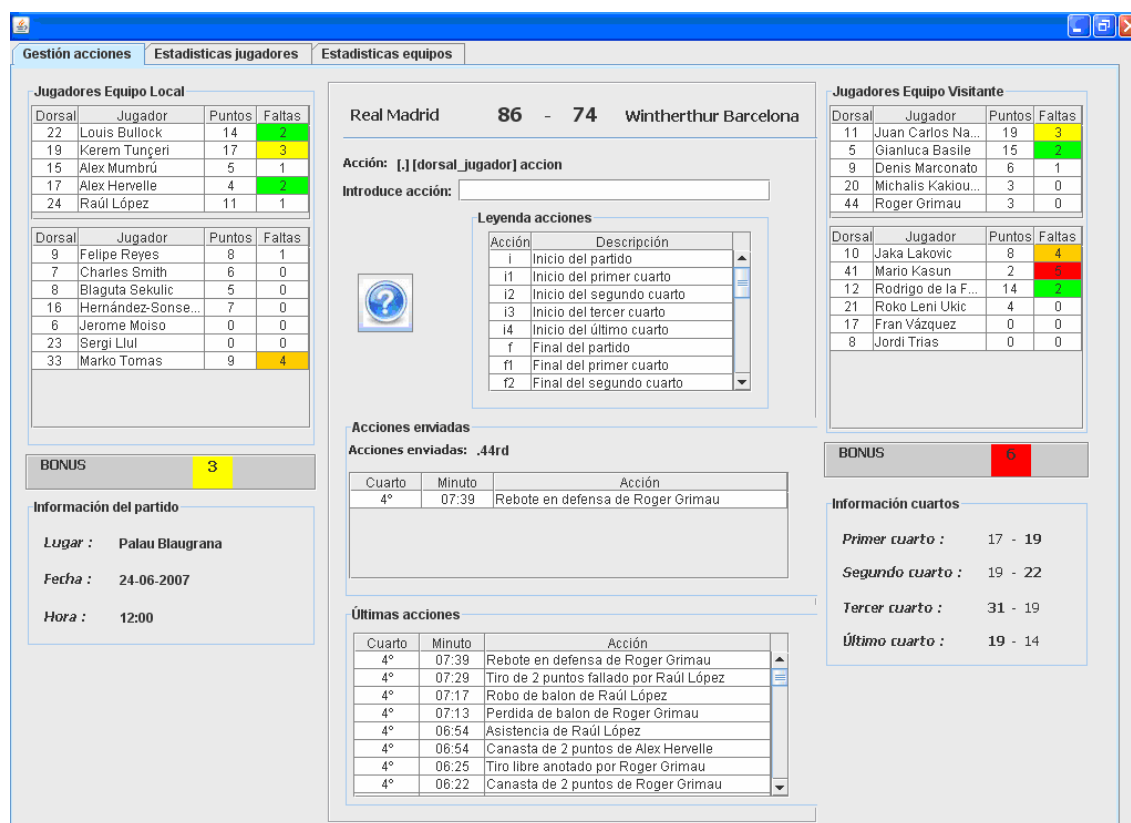


Figura 2: Pantalla principal de la aplicación de escritorio desde la que se introducen las acciones del partido

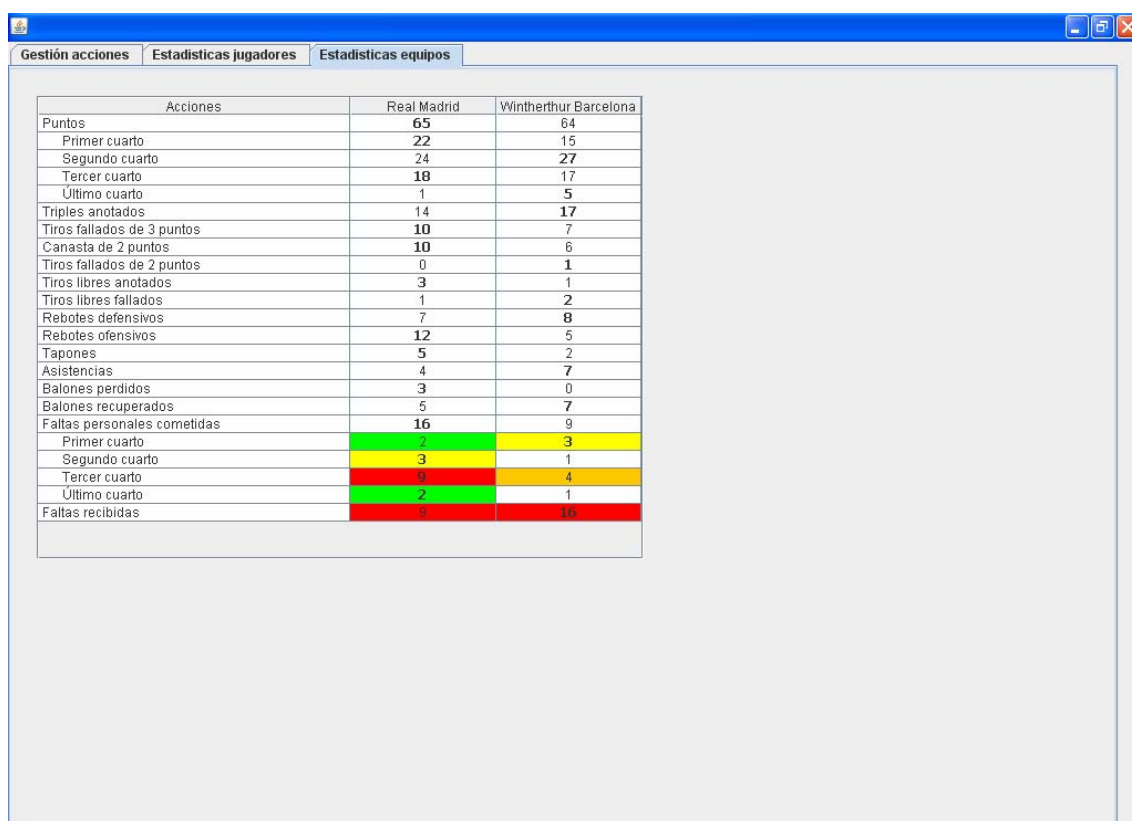
En la *Figura 3*: se muestra la interfaz correspondiente a la segunda pestaña de la aplicación denominada “Estadísticas jugadores”. Como su nombre indica, muestra las estadísticas completas de todos los jugadores de los dos equipos que están disputando el encuentro, incluida su valoración, que más adelante se detallará cómo se calcula.

En la *Figura 4*: se muestra la interfaz correspondiente a la tercera pestaña de la aplicación denominada “Estadísticas equipos”. Como su nombre indica, muestra las estadísticas completas de los dos equipos que están disputando el encuentro, desmenuzando por cuartos la puntuación y las faltas realizadas por cada equipo, ya que son los dos aspectos más importantes.

Estadísticas jugadores equipo local																	
Real Madrid																	
Dorsal	Nombre	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
22	Louis Bullock	12	12	2	2	1	3	4	0	3	0	1	3	2	5	0	2
19	Kerem Tunçeri	3	6	1	1	1	1	1	1	4	2	0	0	4	2	2	1
15	Alex Mumbrú	-5	3	0	2	1	2	1	2	0	0	2	1	1	3	1	0
17	Alex Hervelle	8	9	2	0	1	0	1	0	2	1	1	0	2	0	3	0
9	Felipe Reyes	14	14	0	1	7	4	0	3	5	4	4	2	3	2	0	0
7	Charles Smith	-21	0	0	3	0	3	0	2	1	1	0	0	1	0	5	0
8	Blaguta Sekulic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	Hdez-Sonseca	-1	9	0	1	1	3	7	2	0	2	1	1	2	3	4	0
6	Jerome Moiso	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	Raúl López	-1	0	0	0	0	4	0	3	3	1	0	5	2	5	2	0
23	Sergi Llul	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	Marko Tomas	-10	8	2	4	1	5	0	2	3	4	2	0	4	1	0	0

Estadísticas jugadores equipo visitante																	
Wintherthur Barcelona																	
Dorsal	Nombre	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
10	Jaka Lakovic	4	12	2	1	3	3	0	2	1	3	1	0	1	2	3	0
11	JC.Navarro	9	15	2	2	4	2	1	3	2	0	0	4	3	4	1	1
5	G.Basile	18	13	2	0	3	0	1	0	1	4	2	2	3	1	2	0
9	D. Marconato	-3	11	1	4	1	1	6	2	3	0	0	0	2	0	0	1
41	Mario Kasun	-4	0	0	1	0	4	0	1	2	2	0	1	0	3	0	0
12	R. de la Fuente	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	M. Kakiouzis	2	5	1	0	0	0	2	0	1	2	2	0	4	0	4	0
21	Roko L. Ukic	2	5	1	1	0	1	2	0	0	1	1	2	1	2	5	2
17	Fran Vázquez	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	Roger Grimau	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	Jordi Trias	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 3: Pantalla de la aplicación de escritorio desde la que se visualizan las estadísticas completas de los jugadores



Acciones	Real Madrid	Wintherthur Barcelona
Puntos	65	64
Primer cuarto	22	15
Segundo cuarto	24	27
Tercer cuarto	18	17
Ultimo cuarto	1	5
Triples anotados	14	17
Tiros fallados de 3 puntos	10	7
Canasta de 2 puntos	10	6
Tiros fallados de 2 puntos	0	1
Tiros libres anotados	3	1
Tiros libres fallados	1	2
Rebotes defensivos	7	8
Rebotes ofensivos	12	5
Tapones	5	2
Asistencias	4	7
Balones perdidos	3	0
Balones recuperados	5	7
Faltas personales cometidas	16	9
Primer cuarto	2	3
Segundo cuarto	3	1
Tercer cuarto	0	4
Ultimo cuarto	2	1
Faltas recibidas	3	16

Figura 4: Pantalla de la aplicación de escritorio desde la que se visualizan las estadísticas completas de los equipos

En la *Figura 5*: se muestra la interfaz principal de la aplicación web. En ella se puede observar la retransmisión del partido en un formato legible y muy vistoso, con iconos descriptivos y descripciones de las acciones que están sucediendo. Además, al igual que en la interfaz principal de la aplicación de escritorio (ver *Figura 2*:), se muestran la puntuación de los equipos en cada cuarto y la puntuación en ese instante del partido. Por otra parte, se indica para cada equipo quién es su entrenador, cuál es el quinteto que está jugando en ese instante, cuál es el banquillo en ese instante y cuál es la situación de cada jugador respecto a puntos anotados y faltas personales realizadas. También se muestra la información del partido respecto a la fecha y lugar en que se está disputando.

Retransmisión partido Real Madrid - Wintherthur Barcelona

Real Madrid 65 Wintherthur Barcelona 64

Retransmisión partido | Estadísticas jugadores | Estadísticas equipos

Dorsal	Nombre	Faltas	Puntos
22	Louis Bullock	3	20
19	Keren Tunçeri	1	3
9	Felipe Reyes	0	7
7	Charles Smith	0	10
24	Raül López	2	5
15	Alex Mumbri	0	2
17	Alex Hervelle	3	6
8	Blaguta Selkovic	0	0
16	Hider-Soneca	1	4
6	Berona Moiso	0	0
23	Sergi Llul	0	0
33	Marico Tomas	1	8
BONUS		2	
Entrenador: Joan Plaza			

Cuarto	Tiempo	Acción	Descripción
4º	03:23	🕒	Tiempo muerto
4º	03:15	🚫	Falta personal de Raül López
4º	03:07	🔄	Se produce un cambio: Entra Raül López y sale Alex Hervelle
4º	02:31	🏀	Triple de Roko L. Ukic
4º	02:13	🚫	Falta personal de Alex Hervelle
4º	01:57	🏀	Rebote en ataque de Alex Hervelle
4º	01:11	🏀	Canasta de 2 puntos de Roko L. Ukic
4º	01:11	🏀	Asistencia de Roger Grimau
4º	00:43	🏀	Tiro libre fallado por Charles Smith
4º	00:40	🏀	Tiro libre anotado por Charles Smith
4º	00:27	🏀	Falta recibida por Charles Smith
4º	00:22	🚫	Falta personal de G.Basile
		🏀	Inicio del ultimo cuarto
3º	03:13	🏀	Final del tercer cuarto

Dorsal	Nombre	Faltas	Puntos
10	Jaka Latovic	4	11
5	G.Basile	1	6
41	Mario Kasun	0	12
21	Roko L. Ukic	0	8
44	Roger Grimau	0	5
11	JC Navarro	3	9
9	D. Marconato	1	6
12	R. de la Fuente	0	7
20	M. Katicic	0	0
17	Fran Vazquez	0	0
8	Jordi Trias	0	0
BONUS		1	
Entrenador: Dasko Ivanovic			

CUARTOS	
Primer cuarto:	22 - 15
Segundo cuarto:	24 - 27
Tercer cuarto:	18 - 17
Último cuarto:	1 - 5

INFORMACIÓN PARTIDO

Pabellón: Palacio de Vistalegre

Fecha: 24-06-2007

Hora: 12:00 horas

Figura 5: Pantalla principal de la aplicación web donde se visualiza la retransmisión de las acciones que están sucediendo a lo largo del partido

En la *Figura 6*: se muestra la interfaz correspondiente a la segunda pestaña de la aplicación web denominada "Estadísticas jugadores". Esta interfaz realiza el mismo cometido que la *Figura 3*., que es la interfaz análoga de la aplicación de escritorio. Las únicas mejoras con respecto a ésta es que se muestra la foto de cada jugador si se dispone de ella, y su demarcación. Además en la parte superior sigue apareciendo el resultado del partido para que mientras se consultan las estadísticas de los jugadores, se conozca el resultado del partido sin tener que cambiar de pestaña.


En la *Figura 7*: se muestra la interfaz correspondiente a la tercera pestaña de la aplicación web denominada "Estadísticas equipos". Esta interfaz realiza el mismo cometido que la *Figura 4*: sin ningún cambio reseñable.

Retransmisión partido Real Madrid - Wintherthur Barcelona - Windows Internet Explorer

http://localhost:8084/BasketWeb2/index.jsp


Retransmisión partido Real Madrid - Wintherthur Barc...

Página Herramientas



73


61















Real Madrid


Wintherthur Barcelona

Retransmisión partido Estadísticas jugadores Estadísticas equipos



Estadísticas de los jugadores del Real Madrid

Foto	Dorsal	Nombre	Demarcación	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
	22	Louis Bullock	Escolta	12	12	2	2	1	3	4	0	3	0	1	3	2	5	0	2
	19	Kerem Tunçeri	Base	3	6	1	1	1	1	1	1	4	2	0	0	4	2	2	1
	15	Ales Mumbri	Alero	-5	3	0	2	1	2	1	2	0	0	2	1	1	3	1	0
	17	Alex Herveille	Ala-Pivot	8	9	2	0	1	0	1	0	2	1	1	0	2	0	3	0
	9	Felipe Reyes	Pivot	14	14	0	1	7	4	0	3	5	4	4	2	3	2	0	0
	7	Charles Smith	Escolta	-21	0	0	3	0	3	0	2	1	1	0	0	1	0	3	0
	8	Blaguta Sekulic	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	Hdez-Sonseca	Pivot	-1	9	0	1	1	3	7	2	0	2	1	1	2	3	4	0
	6	Jerome Moiso	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	24	Raül López	Base	11	12	2	0	2	4	2	3	3	1	0	5	2	5	2	0
	23	Sergi Llul	Base	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	33	Marko Tomas	Escolta	-10	8	2	4	1	5	0	2	3	4	2	0	4	1	0	0



Estadísticas de los jugadores del Wintherthur Barcelona












Foto	Dorsal	Nombre	Demarcación	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
	10	Jaka Lakovic	Base	4	12	2	1	3	3	0	2	1	3	1	0	1	2	3	0
	11	JC Navarro	Base	9	15	2	2	4	2	1	3	2	0	0	4	3	4	1	1
	5	G Basile	Escolta	18	13	2	0	3	0	1	0	1	4	2	2	3	1	2	0
	9	D. Marconato	Pivot	-3	11	1	4	1	1	6	2	3	0	0	0	2	0	0	1
	41	Mario Kasun	Pivot	-4	0	0	1	0	4	0	1	2	2	0	1	0	3	0	0
	12	R. de la Fuente	Escolta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	M. Kakiozis	Ala-Pivot	2	5	1	0	0	0	2	0	1	2	2	0	4	0	4	0
	21	Roko L. Ukić	Base	2	5	1	1	0	1	2	0	0	1	1	2	1	2	3	2
	17	Fran Vázquez	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	44	Roger Grimau	Escolta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	Jordi Trias	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 6: Pantalla de la aplicación web desde la que se visualizan las estadísticas completas de los jugadores

Retransmisión partido Real Madrid - Wintherthur Barcelona - Windows Internet Explorer

http://localhost:8080/index.jsp

Retransmisión partido Real Madrid - Wintherthur Barc...

Real Madrid 65 64 Wintherthur Barcelona

Retransmisión partido Estadísticas jugadores Estadísticas equipos

Estadísticas de los equipos


Acción	Real Madrid	Wintherthur Barcelona
		
Puntos	65	64
Primer cuarto	22	15
Segundo cuarto	24	27
Tercer cuarto	18	17
Último cuarto	1	5
Triples anotados	14	17
Tiros fallados de 3 puntos	10	7
Canasta de 2 puntos	10	6
Tiros fallados de 2 puntos	0	1
Tiros libres anotados	3	1
Tiros libres fallados	1	2
Rebotes defensivos	7	8
Rebotes ofensivos	12	5
Taques	5	2
Asistencias	4	7
Balones perdidos	3	0
Balones recuperados	5	7
Faltas personales cometidas	16	9
Primer cuarto	2	3
Segundo cuarto	3	1
Tercer cuarto	9	4
Último cuarto	2	1
Faltas recibidas	9	16

Figura 7: Pantalla de la aplicación web desde la que se visualizan las estadísticas completas de los equipos

1.1 OBJETIVOS

Con este proyecto, se pretende crear una herramienta informática que permita registrar a pie de cancha las acciones que suceden en un partido de baloncesto de forma sencilla, intuitiva y ágil, y que sea capaz de realizar los cálculos necesarios con ellas para obtener estadísticas completas del partido, de cada jugador y de los equipos involucrados en el partido.

Además, estas estadísticas se publicarán en tiempo real en una web, en la que también aparecerá una narración fácilmente legible de las acciones que están sucediendo a lo largo del partido.

La forma de introducir las acciones que suceden a lo largo de un partido en la herramienta debe ser cómoda, ágil, sencilla e intuitiva, de forma que permita a la persona que introduce los datos perder el menor tiempo posible el contacto visual con el terreno de juego. Por tanto, se realizará manualmente utilizando el teclado con un conjunto de comandos sencillos e intuitivos que se definirán más adelante. Otra de las grandes ventajas de este nuevo sistema frente a los ya existentes, es la posibilidad de introducir un conjunto de acciones de forma simultánea, ya que el hecho de que en un partido de baloncesto sucedan varias acciones prácticamente al mismo tiempo es algo muy usual. Por ejemplo, al mismo tiempo se puede haber producido una asistencia, un tiro fallado por un jugador, y un tapón realizado por otro jugador.

La herramienta almacenará, gestionará y organizará toda la información estadística generada por un equipo de baloncesto. Esta información consiste principalmente en los datos de los jugadores, los datos de los equipos y las estadísticas de los partidos que estos jugadores y equipos disputen. Esta información se podrá utilizar por el equipo para obtener otro tipo de estadísticas, como podrían ser las estadísticas de los jugadores durante una liga, o las estadísticas del equipo a lo largo de una liga, pero ambas opciones son posibles mejoras que no alcanzan los objetivos de este proyecto. Sin embargo, el diseño de la base de datos ya debe contar con esta posibilidad.

El objetivo final de este proyecto es que la herramienta generada tenga una aplicación en el mundo real.

1.2 DESCRIPCIÓN GENERAL DEL ENTORNO

Este proyecto surge para subsanar un interés particular del Director de Proyecto, D. Raúl Montoliu Colás, quién entrena un equipo de cantera que disputa la liga federada de determinado municipio. Así pues, nos encontramos en un entorno escolar, en el que los potenciales usuarios de la aplicación, más en concreto de su parte web, van a ser los padres de los jugadores que en este caso son niños, y a todo caso algún profesor o amigo. Esta aplicación web también puede llegar a utilizarse para conocer las estadísticas del equipo contrario en el encuentro, y utilizar esta información para encuentros futuros.

En este entorno de equipos pequeños o equipos de cantera, ya se había comentado anteriormente que la única información de los partidos de que se dispone, son las actas de los partidos y las estadísticas que recogen para cada equipo una persona correspondiente al cuerpo técnico denominada *estadígrafo*. Así pues, esta aplicación la va a utilizar una única persona para introducir los datos (estadígrafo) y varias personas para consultar las estadísticas vía web.

La experiencia en este campo del interesado, consiste en la utilización de la aplicación MBT SmartStat [7], con la que no está satisfecho debido a que su interfaz se basa en el uso del ratón y ello le imposibilita registrar todas las estadísticas de un partido. Hasta tal punto llega esa incomodidad, que es necesaria una persona auxiliar que realice el dictado de las acciones del partido. De esta forma, esta persona no pierde el contacto visual con el partido, y la otra persona se puede dedicar en exclusiva a introducir acciones por medio del ratón. Aún así, sólo da tiempo de introducir las acciones de un solo equipo, porque se deben introducir las acciones una por una, no pudiendo introducir varias acciones de forma simultánea, con los retrasos que ello conlleva.

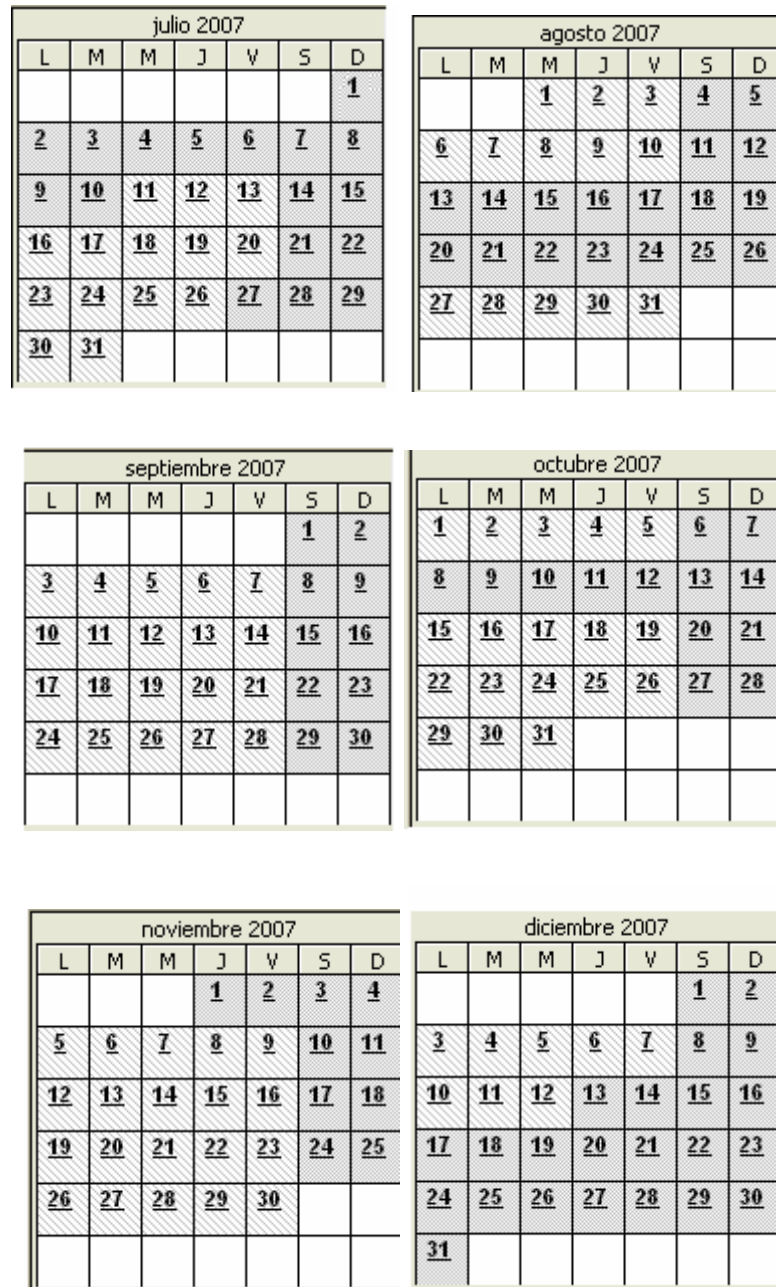
1.3 PLANIFICACIÓN

En las Figuras Figura 9:, Figura 10: y Figura 11: se muestra un Diagrama de Gantt, particionado por fechas para su mejor visualización, en dónde se pueden observar las tareas que se deben realizar para la elaboración de este proyecto, así como su duración y dependencias (tareas relacionadas en que una debe ser obligatoriamente predecesora de otra).

El proyecto comenzó a gestarse el día 11 de Julio de 2007, y la previsión es que finalice el 3 de Diciembre. De esta forma, se tendrá un margen de tiempo suficiente para poder solucionar los imprevistos que surjan en la planificación y que provoquen una variación en la fecha de finalización del proyecto. Marcados estos hitos temporales, y habiendo realizado un estudio de las tareas a realizar, se distribuyen éstas en el tiempo teniendo en cuenta el siguiente “calendario laboral”.

- *Julio*: 4 horas/ día
- *Agosto*: 4 horas / día. Última semana 8 horas/día
- *Septiembre*: 8 horas/día. Última semana 4 horas/día.
- *Octubre*: 2 horas/día
- *Noviembre*: 2 horas/día
- *Diciembre*: 2 horas/día

De esta forma, el tiempo para realizar el proyecto se reduce a 86 días laborables y a 346 horas.

**Figura 8:** Calendario del proyecto

Como se puede observar en la *Tabla 1:* o en las *Figuras Figura 9:, Figura 10: y Figura 11:* se han jerarquizado las tareas. En primera instancia se han jerarquizado las tareas según se encuentren en el Proceso de Planificación o en el Proceso de Desarrollo, y dentro del Proceso de Desarrollo se han jerarquizado en las 4 fases que todo buen proceso de desarrollo de un sistema de información debe tener, es decir, Análisis, Diseño, Implementación y Evaluación.

En el *Proceso de Planificación*, se analizan las tareas a realizar y se distribuyen temporalmente, siempre con un riesgo de equivocación ya que nunca se sabe cuando finalizará una tarea exactamente, pero sí se marca una fecha que se debe intentar cumplir.

Dentro del *Análisis*, las tareas que más importancia y duración presentan son la comprensión de los requisitos, en la cual se realizarán varias entrevistas con el cliente para conocer las cualidades del sistema que desea, y el análisis y estudio de las tecnologías y el software a utilizar. También se establece la arquitectura sobre la que se va a desarrollar el sistema y los diferentes usuarios y roles que harán uso del mismo. Por otra parte, dentro de esta fase también se analiza el estado del arte, buscando y analizando sistemas similares al que se debe realizar.

La fase de *Diseño* se dedica principalmente al diseño de la base de datos, de las clases con que se modela el sistema, de la especificación del nuevo lenguaje que se debe definir para la aplicación (conjunto de comandos con que se introducen las acciones del partido) y de la interfaz del sistema, presentando al cliente al final de esta fase un prototipo no funcional sobre el que podrá realizar proposiciones de mejora de la interfaz.

La fase de *Implementación* corresponde al período en que se desarrolla la aplicación. En este período se presenta un prototipo funcional al cliente para que lo evalúe antes de entregarle el prototipo final. En esta fase también se elabora un manual de usuario con el que el cliente pueda conocer el funcionamiento de la aplicación sin necesidad de realizar cursos de

formación. También se le proporciona un procedimiento de instalación para que sea capaz de instalar todo el software que necesita la aplicación.

Por último, aunque no en orden cronológico, la fase de Evaluación engloba la duración de todo el proyecto ya que en ella se incluyen las reuniones con el cliente que comienzan el primer día para establecer los objetivos del proyecto, y otras reuniones para establecer los requisitos de la nueva aplicación, para revisar el análisis de la nueva aplicación, y para presentarle y que evalúe diversos prototipos de la aplicación. En total se realizan 8 reuniones con el cliente. En esta fase también se fija una prueba en un entorno real de la nueva aplicación para el día 8 de Noviembre, y se elaboran las propuestas de ampliaciones y mejoras del proyecto, se obtienen las conclusiones del mismo y se elabora la presentación que se expondrá el día en que se presente este proyecto.

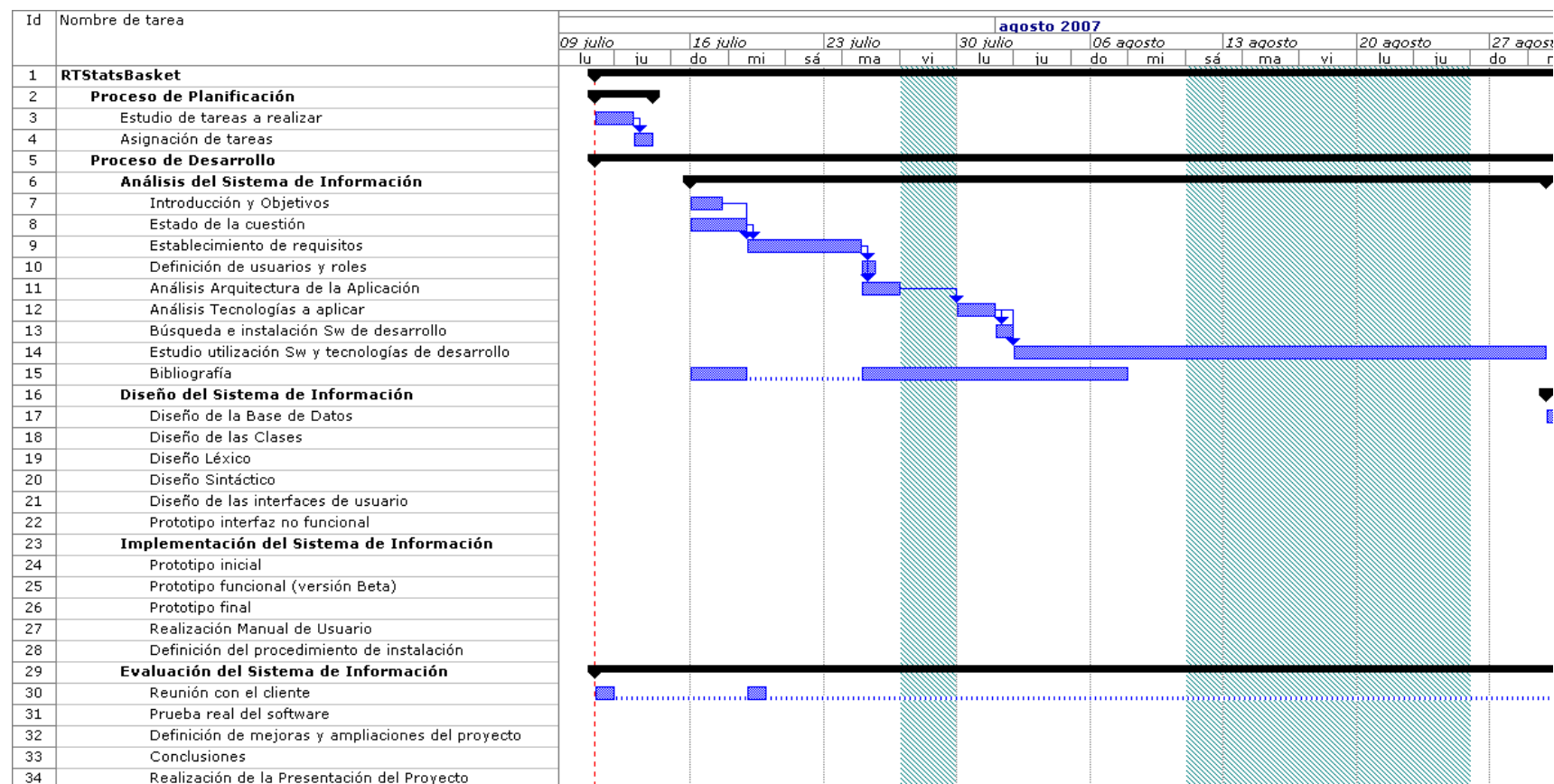


Figura 9: Diagrama de Gantt (11 Julio – 30 Agosto)

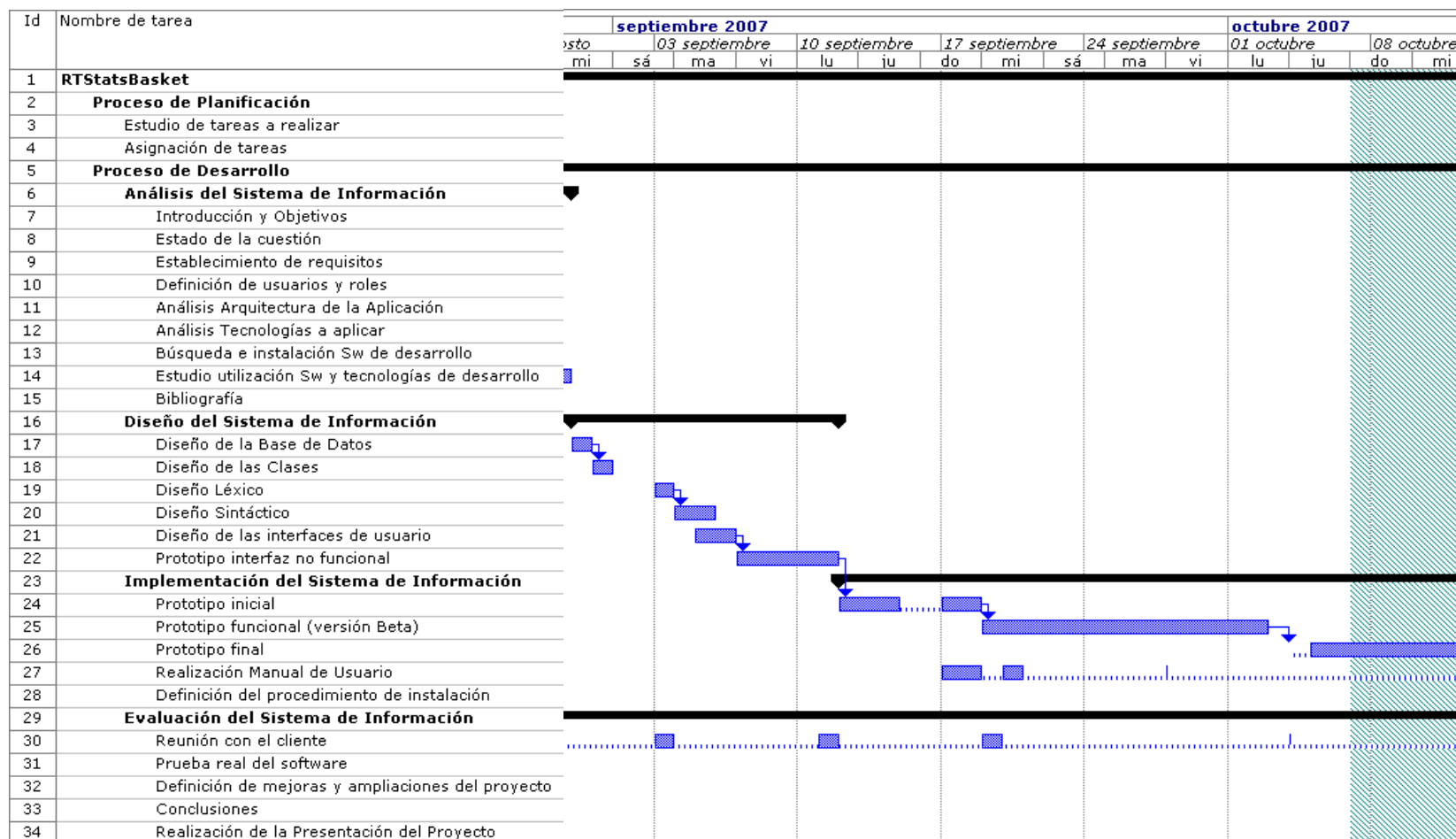


Figura 10: Diagrama de Gantt (30 Agosto – 15 Octubre)

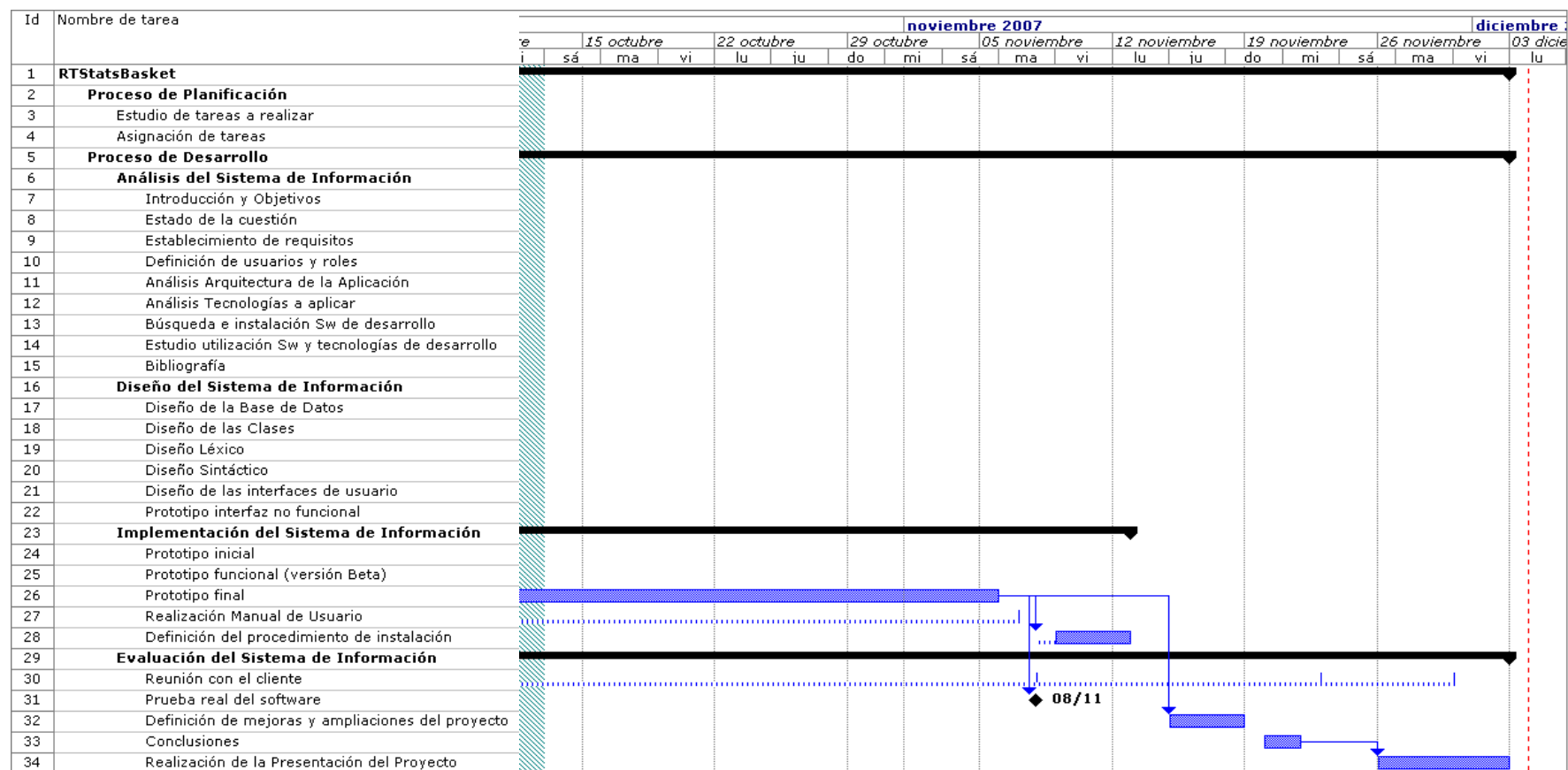


Figura 11: Diagrama de Gantt (15 Octubre – 3 Diciembre)

RTStatBasket	346h	11/07/07	03/12/07
1. Proceso de Planificación	9h	11/07/07	13/07/07
Estudio de tareas a realizar	5h	11/07/07	12/07/07
Asignación de tareas	4h	13/07/07	13/07/07
2. Proceso de Desarrollo	338h	11/07/07	03/12/07
2.1 Análisis del Sistema de Información	97h	16/07/07	29/08/07
Introducción y Objetivos	3h	16/07/07	17/07/07
Estado de la cuestión	8h	16/07/07	18/07/07
Establecimiento de requisitos	13h	19/07/07	24/07/07
Definición de usuarios y roles	2h	25/07/07	25/07/07
Análisis Arquitectura de la Aplicación	5h	25/07/07	26/07/07
Análisis Tecnologías a aplicar	7h	30/07/07	31/07/07
Búsqueda e instalación Sw de desarrollo	3,5h	01/08/07	01/08/07
Estudio utilización Sw y tecnologías de desarrollo	50h	02/08/07	29/08/07
Bibliografía	5,5h	16/07/07	07/08/07
2.2 Diseño del Sistema de Información	68h	30/08/07	11/09/07
Diseño de la Base de Datos	8h	30/08/07	30/08/07
Diseño de las Clases	8h	31/08/07	31/08/07
Diseño Léxico	8h	03/09/07	04/09/07
Diseño Sintáctico	8h	04/09/07	05/09/07
Diseño de las interfaces de usuario	14h	05/09/07	06/09/07
Prototipo interfaz no funcional	22h	07/09/07	11/09/07
2.3 Implementación del Sistema de Información	128h	12/09/07	13/11/07
Prototipo inicial	32h	12/09/07	18/09/07
Prototipo funcional (versión Beta)	46h	19/09/07	03/10/07
Prototipo final	32h	04/10/07	06/11/07
Realización Manual de Usuario	12h	17/09/07	07/11/07
Definición del procedimiento de instalación	6h	08/11/07	13/11/07

2.4 Evaluación del Sistema de Información	44h	11/07/07	03/12/07
Reunión con el cliente	18h	11/07/07	30/11/07
Prueba real del software	2h	08/11/07	08/11/07
Definición de mejoras y ampliaciones del proyecto	6h	15/11/07	19/11/07
Conclusiones	6h	20/11/07	22/11/07
Realización de la Presentación del Proyecto	12h	26/11/07	03/12/07

Tabla 1: Descripción detallada de las tareas del proyecto

1.4 ORGANIZACIÓN DEL DOCUMENTO

Con el fin de documentar este proyecto, se ha dividido su memoria en 6 capítulos y 2 anexos ordenados cronológicamente según su realización.

La estructura que presenta esta memoria es la siguiente:

- Capítulo 1 - INTRODUCCIÓN: Es el capítulo actual, en el que se realiza una breve introducción al proyecto, indicando la motivación, objetivos y entorno para el que se va a realizar, así como el estado del arte o aplicaciones ya existentes que cubren las necesidades que propone este proyecto. También contiene la planificación temporal de las actividades a llevar a cabo para la realización del mismo.
- Capítulo 2 – DESCRIPCIÓN DEL PROYECTO: En este capítulo se detallarán global y específicamente todas las actividades realizadas para llevar a cabo este proyecto.
 - Análisis: Se realiza un análisis de la arquitectura sobre la que funcionará la nueva aplicación, se definen los distintos usuarios y roles del sistema, y, se realiza un completo estudio de las tecnologías utilizadas.
 - Diseño: Se detallará el diseño de la base de datos incluyendo el diagrama entidad-relación y el diseño lógico. También se expondrá el diseño de las clases, la definición del nuevo lenguaje de comandos para introducir acciones así como el diseño de su nivel léxico, sintáctico y semántico, y también los diseños de la interfaces de usuario.

- Capítulo 3 – EXPERIMENTACIÓN: PRUEBAS Y RESULTADOS: Se expondrá en este capítulo el resultado de las distintas pruebas a las que ha sido sometida la aplicación.
- Capítulo 4 - CONCLUSIONES: Se presentarán las conclusiones alcanzadas tras la realización del proyecto.
- Capítulo 5 – POSIBLES MEJORAS Y AMPLIACIONES: Se analizarán las posibles mejoras y/o ampliaciones que se podrán realizar a la aplicación en el futuro.
- Capítulo 6 - REFERENCIAS: Bibliografía formada por referencias a las distintas direcciones URL utilizadas durante la realización de este proyecto.
- Anexo 1 – MANUAL DE USUARIO: Como anexo a la documentación se incluirá un manual de usuario que facilite la utilización de la nueva aplicación.
- Anexo 2 – PROCEDIMIENTO DE INSTALACIÓN: Se especificará en este capítulo el software que necesariamente deberá estar instalado en la máquina donde se desee implantar la aplicación y también se expondrá cómo ejecutar la aplicación.

Descripción del proyecto

2.1	Análisis	27
2.1.1	Arquitectura del sistema	27
2.1.2	Definición de usuarios y roles	32
2.1.3	Tecnologías empleadas	33
2.2	Diseño	42
2.2.1	Definición del nuevo lenguaje de comandos	42
2.2.2	Diseño de las clases	55
2.2.3	Diseño de la base de datos.....	64
2.2.4	Diseño de las interfaces de usuario.....	67

2. DESCRIPCIÓN DEL PROYECTO

2.1 ANÁLISIS

2.1.1 Arquitectura del sistema

La arquitectura del sistema implementado en este proyecto se puede dividir en 2 partes: la arquitectura de la aplicación de escritorio y la arquitectura de la aplicación web.

APLICACIÓN DE ESCRITORIO

La aplicación de escritorio que se ha desarrollado en este proyecto, utiliza una arquitectura cliente-servidor construida de tal modo que la base de datos puede residir en un equipo central, denominado servidor y ser compartida entre varios usuarios denominados clientes. En este caso, el cliente únicamente es uno, con rol de *estadígrafo*, que es el encargado de introducir las acciones que suceden en el partido a la base de datos del servidor. También recibe información procesada a partir de esos datos del servidor para actualizar su interfaz de usuario. Así pues, se utiliza una arquitectura cliente-servidor de dos capas donde los usuarios con rol *estadígrafo* ejecutan una aplicación en su equipo local, llamado cliente, que se conecta a través de la red con el servidor que ejecuta la base de datos e intercambian información (ver *Figura 12:*).

El hecho de tener los datos almacenados y administrados en una ubicación central ofrece varias ventajas:

- Todos los datos están almacenados en una ubicación central en donde todos los usuarios pueden trabajar con ellos. Así, los clientes de la aplicación web podrán obtener información coherente con la del estadígrafo.
- No se almacenan copias separadas del elemento en cada cliente, lo que elimina los problemas de hacer que todos los usuarios trabajen con la misma información, es decir, de actualizar esas copias para que sean coherentes.

- Las reglas de la organización y las reglas de seguridad se pueden definir una sola vez en el servidor para todos los usuarios. Esto se puede hacer en una base de datos mediante el uso de restricciones, procedimientos almacenados y desencadenadores.
- Los servidores de base de datos relacionales optimizan el tráfico de la red al devolver sólo los datos que la aplicación necesita.
- Los gastos en hardware se pueden minimizar. Como los datos no están almacenados en los clientes, éstos no tienen que dedicar espacio de disco a almacenarlos. Los clientes tampoco necesitan la capacidad de proceso para administrar los datos localmente y el servidor no tiene que dedicar capacidad de proceso para presentar los datos.
- El servidor se puede configurar para optimizar la capacidad de E/S de disco necesaria para obtener los datos y los clientes se pueden configurar para optimizar el formato y presentación de los datos obtenidos desde el servidor.
- El servidor puede estar situado en una ubicación relativamente segura y estar equipado con dispositivos como Sistemas de alimentación ininterrumpida (SAI), lo que resulta más económico que si se protegieran todos los clientes.
- Las tareas de mantenimiento como las copias de seguridad y restauración de los datos son más sencillas porque están concentradas en el servidor central.

Antes de realizar una petición al servidor, ésta es evaluada y procesada en el cliente, de forma que lo que se envía es una petición de inserción de datos, o una consulta optimizada. El servidor envía al cliente bloques de información que, una vez en el cliente son procesados para mostrar la información solicitada. Así pues, muy poca actividad lógica se desarrolla en el servidor. El cliente es el encargado de desarrollarla casi completamente, ocupándose así de la ejecución de la lógica correspondiente a las reglas de negocio y por otra parte de la interfaz de usuario. Por este hecho, algunos autores consideran que esta arquitectura Cliente –Servidor posee un cliente "grueso" y un servidor "delgado".



Figura 12: Arquitectura Cliente- Servidor de la aplicación de escritorio

APLICACIÓN WEB

El modelo o arquitectura clásica de aplicaciones Web funciona de la forma que se detalla a continuación. Cualquier usuario interacciona con las aplicaciones web a través de un navegador. Como consecuencia de la actividad del usuario, se envían peticiones HTTP al servidor, donde se aloja la aplicación que normalmente hace uso de una base de datos que almacena toda la información relacionada con la misma. El servidor procesa la petición y devuelve la respuesta HTML al navegador, que es el encargado de presentarla al usuario. Por tanto, el sistema se distribuye en tres componentes: el navegador (Cliente), que presenta la interfaz al usuario; la aplicación web (Servidor), que se encarga de realizar las operaciones necesarias según las acciones llevadas a cabo por éste; y la base de datos (Servidor), donde la información relacionada con la aplicación se hace persistente. Esta distribución se conoce como el modelo o arquitectura de tres capas [4] (ver *Figura 13:*).

Este modelo clásico tiene mucho sentido a nivel técnico, pero no lo tiene para una gran experiencia de usuario ya que mientras la aplicación en el servidor está procesando la petición del usuario, éste está esperando la respuesta. Las aplicaciones web no deberían hacer esperar a los usuarios si quieren hacer frente a las aplicaciones de escritorio.

Una aplicación web que utiliza AJAX [13] (tecnología utilizada que se explica en el siguiente apartado) elimina esa naturaleza “arrancar-frenar-arrancar-frenar” de la interacción en la Web cuando se sigue el modelo clásico. Para ello, introduce un intermediario (un motor AJAX) entre el usuario y el servidor (ver *Figura 13:*). El funcionamiento es el siguiente: una vez que se ha cargado una página web al inicio de la sesión, el navegador carga el motor AJAX. Este motor es el responsable de renderizar la interfaz que el usuario ve y de comunicarse con el servidor en nombre del usuario. El motor AJAX permite que la interacción del usuario con la aplicación en el servidor suceda asincrónicamente. Así el usuario nunca estará mirando una ventana en blanco del navegador ni un icono de reloj de arena esperando a que el servidor haga algo.

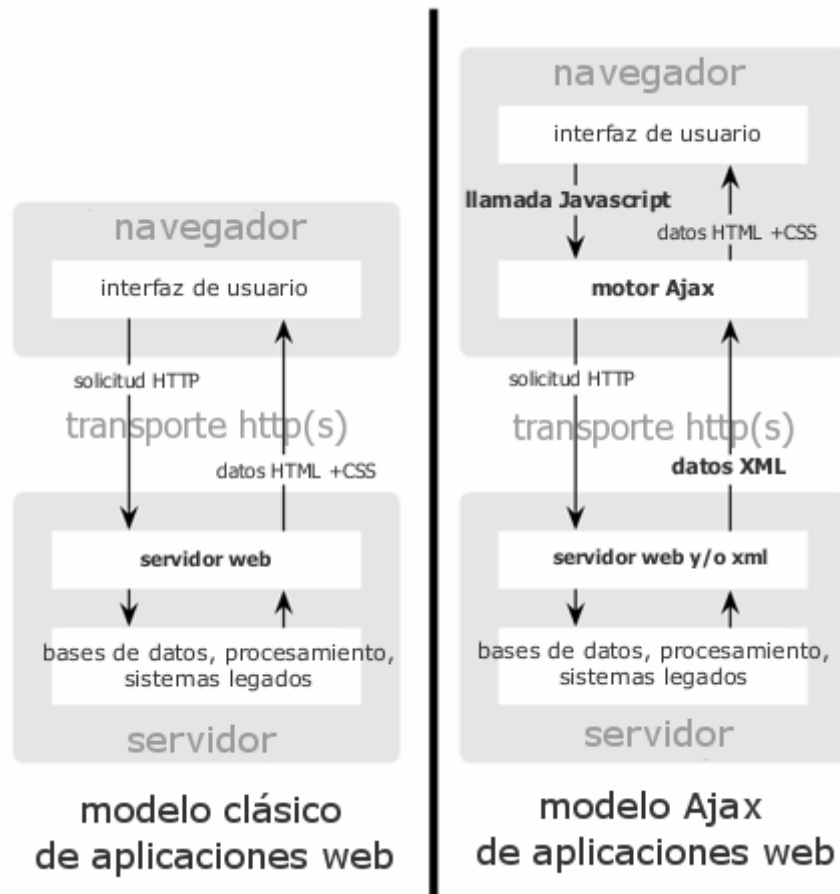


Figura 13: El modelo tradicional para las aplicaciones Web (izq.) comparado con el modelo de AJAX (der.).

Cada acción de un usuario que normalmente generaría una petición HTTP al servidor, toma la forma de una llamada JavaScript al motor AJAX en vez de esa petición. Cualquier respuesta a una acción del usuario que no requiera una comunicación con el servidor, como una simple validación de datos, es manejada por cuenta del motor AJAX. Si éste necesita algo del servidor para responder (sea enviando datos para procesar, cargar código adicional, o recuperar nuevos datos) hace esos pedidos asincrónicamente al servidor usando normalmente XML, pero sin frenar la interacción del usuario con la aplicación (ver *Figura 14:*).

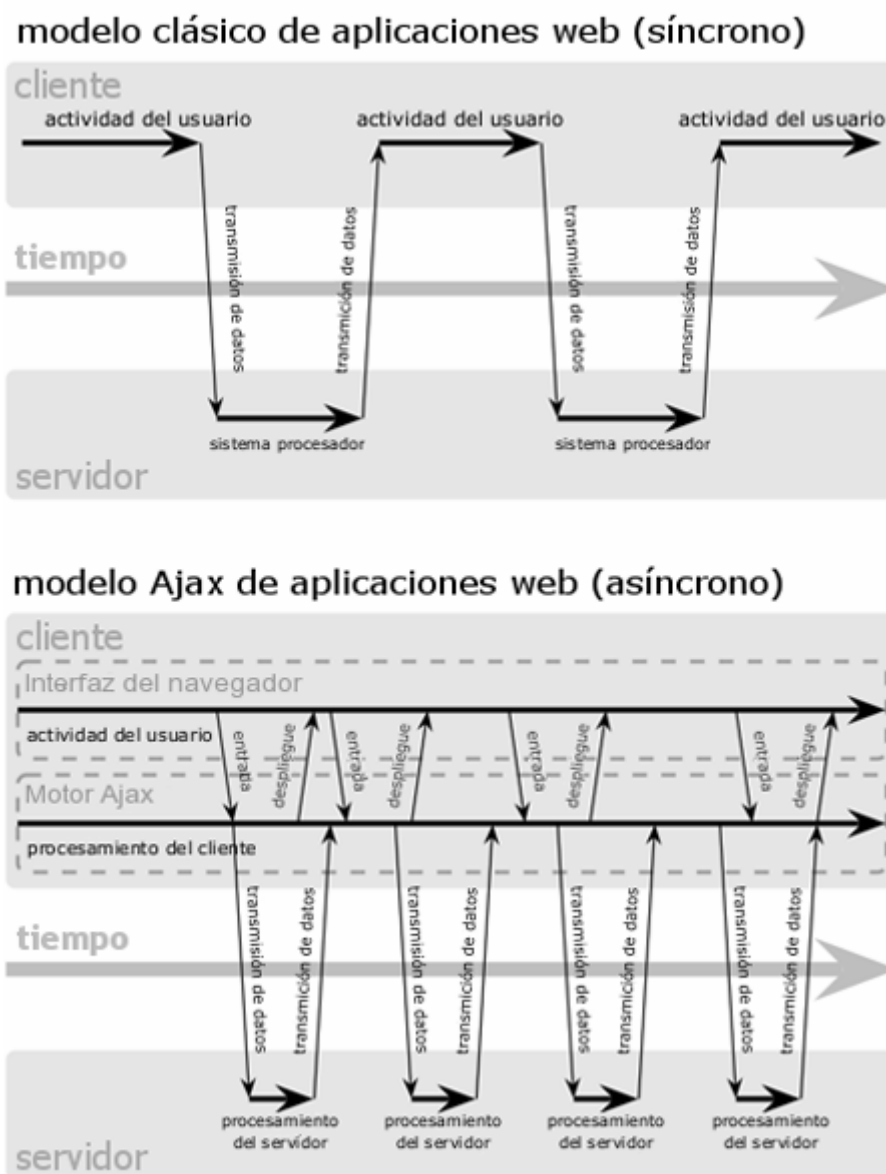


Figura 14: El patrón de interacción sincrónica de una aplicación Web tradicional (arriba) comparada con el patrón asincrónico de una aplicación AJAX (abajo).

Así pues, el modelo o arquitectura pasa a tener 4 capas, o visto de otra manera posee 3 capas pero el modelo del cliente se sitúa en un punto intermedio entre un modelo de cliente delgado (el navegador suele ser un mero presentador de información y no lleva a cabo ningún procesamiento relacionado con la lógica de negocio) y un modelo de cliente grueso (donde el cliente realiza el procesamiento de la información y el servidor sólo es responsable de la administración de datos). El procesamiento realizado en el cliente está relacionado con aspectos de la interfaz (como ocultar o mostrar secciones de la página en función de determinados eventos) y nunca con la lógica de negocio.

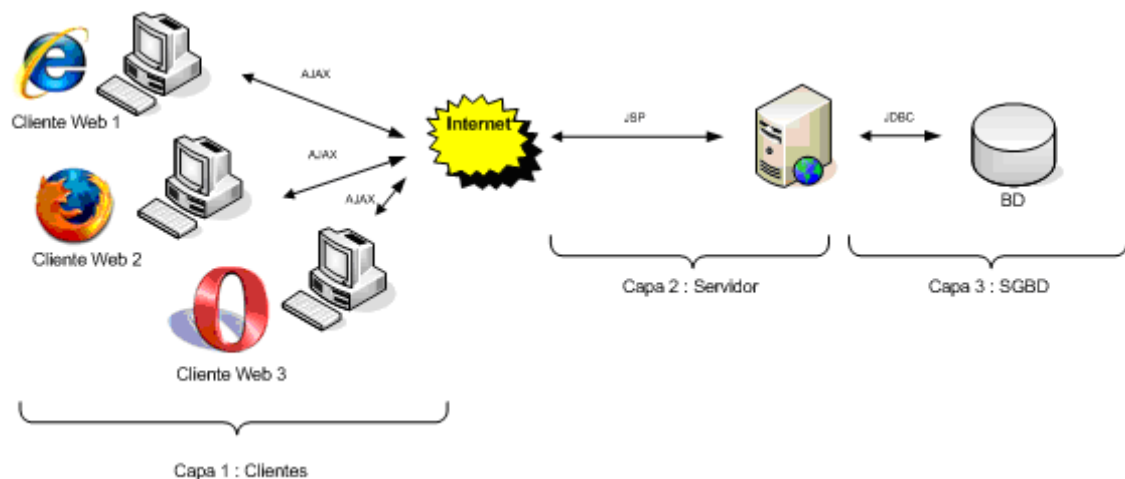


Figura 15: Arquitectura de la aplicación web

2.1.2 Definición de usuarios y roles

La aplicación de escritorio va a tener un único usuario con rol de *estadígrafo* que es el encargado de introducir las acciones que suceden a lo largo de un partido. Podría para un futuro ampliarse y establecer 2 roles, el estadígrafo que introduce las acciones, y el *entrenador*, que visualiza las estadísticas que han generado esas acciones, pero con este proyecto las podría visualizar a través de la web así que no tiene mucho sentido. Existe una persona que introduce las acciones, y el entrenador puede acercarse para visualizar las estadísticas, ya que cuando se use este proyecto en la realidad, serán personas del mismo cuerpo técnico.

La aplicación web tiene muchos usuarios todos con el mismo rol, el de *visitantes*, ya que visualizan las estadísticas que se van generando del partido a tiempo real a través de Internet. Se podría añadir otro rol de *administrador* si se quisiera generar una parte privada de la web en la que este usuario pudiese modificar su diseño, pero este caso sería una ampliación de este proyecto.

2.1.3 Tecnologías empleadas

APLICACIÓN DE ESCRITORIO

A la hora de plantear con qué tecnología implementar la aplicación que permite introducir las acciones de los partidos, en un primer momento surgió la duda de si realizar una aplicación de escritorio o una aplicación web. Se decidió realizar una aplicación de escritorio porque muchos de los pabellones donde van a jugar los pequeños equipos no disponen de Internet, y por tanto interesaría más una aplicación de escritorio que pudiese funcionar sin Internet. Esto implicaría crear una copia de la base de datos y posteriormente cargarla en el servidor, aunque para ello se necesitaría incrementar la funcionalidad de la aplicación que se va a llevar a cabo en este proyecto. Además, habitualmente son mucho más rápidas las aplicaciones de escritorio que las aplicaciones web, lo que supuso un factor determinante en esta decisión, ya que todo debe ir orientado a la agilidad a la hora de introducir acciones.

Una vez confirmada la decisión de crear una aplicación de escritorio, surgió la duda de qué lenguaje era más conveniente utilizar. La mayoría de las aplicaciones informáticas de gestión están implementadas con lenguajes de programación orientados a objetos, porque poseen una interfaz gráfica muy intuitiva y facilitan la interacción con el sistema. Por tanto, Java, .NET, Visual C++, Delphi, etc. eran algunas alternativas. De todas ellas se escogió Java [18], debido a que permite que la aplicación sea multiplataforma, y a que el lenguaje de servidor escogido para la parte web del proyecto también es Java, y por tanto se van a poder reutilizar clases y código.

La biblioteca gráfica de Java se denomina **Swing** [19], e incluye widgets para la interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas.

Swing existe desde la JDK 1.1 [23] como un agregado. Antes de la existencia de Swing, las interfaces gráficas de usuario en Java se realizaban a través de AWT (Abstract Window Toolkit), de quien Swing hereda todo el manejo de eventos. Normalmente, para toda componente AWT existe una componente Swing que la reemplaza, por ejemplo, la clase Button de AWT es reemplazada por la clase JButton de Swing (el nombre de todas las componentes Swing comienza con "J"). Por tanto, las componentes de Swing utilizan la infraestructura de AWT, incluyendo el modelo de eventos AWT, que establece cómo un componente reacciona a eventos tales como, eventos de teclado, ratón, etc.

Tal y como se comentó en el apartado anterior de arquitectura, esta aplicación de escritorio se conecta a un servidor de bases de datos para añadir y extraer información de los partidos. De entre todos los SGBD que hay en el mercado [6] se debía elegir uno, y se selecciona **MySQL 5.0** [6] porque es libre, completo, disponible en la mayoría de alojamientos web, sencillo de configurar e instalar, y muy en boga actualmente. Además viene integrado en el kit XAMPP, como se explica en el ANEXO 2: PROCEDIMIENTO DE INSTALACIÓN, junto al servidor Apache y el servidor Tomcat, necesarios para el funcionamiento de la aplicación web.

Para manejar la conexión al SGBD MySQL y la manipulación de los datos desde la aplicación Swing se utiliza el API (Interfaz para el Programador de Aplicaciones) denominado **JDBC** (Java DataBase Connectivity) [20].

Un API es un conjunto de funciones que proveen los fabricantes de hardware y/o software para que los programadores no deban llegar a bajo nivel para relacionarse con sus productos y a cambio lo hagan a través de estas funciones. Actúan sobre los controladores o drivers (similares a los drivers para impresoras, cámaras, etc), que son los encargados de comunicarse con el sistema operativo y poner a disposición sus productos. Existe por lo menos un driver para cada producto.

Por tanto, para que la aplicación de escritorio se pueda comunicar con la Base de Datos debe estar habilitado el driver correspondiente.

JDBC [21] es un API específico para lenguaje Java. Consiste en un conjunto de clases que representan conexiones con bases de datos, sentencias SQL, conjuntos de datos y metadatos entre otras cosas. El API definido por JDBC permite a los programadores enviar sentencias SQL al SGBD y procesar los resultados.

Como en los otros casos se requiere de un driver o controlador, en este caso JDBC. Los drivers JDBC se han clasificado en tipos, según el gestor de bases de datos con el que se comunican. En este caso se selecciona el driver JDBC para MySQL.

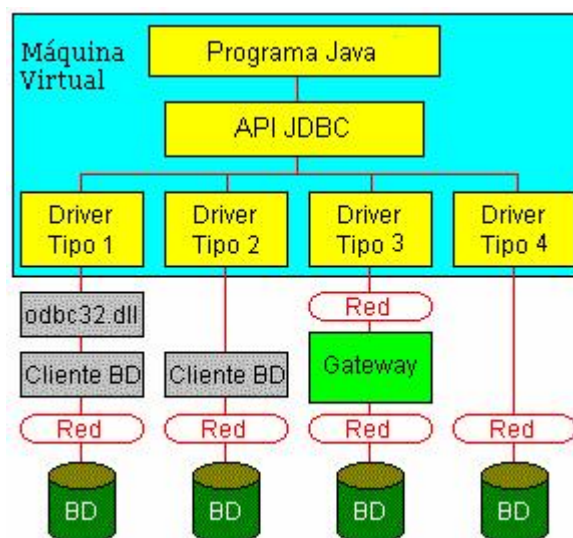


Figura 16: Conexión de una aplicación Java a una base de datos mediante JDBC

APLICACIÓN WEB

Para la aplicación web del proyecto es necesario un lenguaje de lado del servidor [5] ya que se debe cargar contenido dinámicamente. De entre los existentes: ASP (Active Server Pages) de Microsoft, PHP, Perl, CGI (Common Gateway Interface), y JSP (Java), se selecciona **JSP** [22] ya que como la aplicación de escritorio se realiza en Swing (Java), se podrán reutilizar clases y código. Además ya se ha trabajado con PHP en varias asignaturas de la carrera y se desea profundizar en el estudio de una nueva tecnología.

La principal ventaja de JSP frente a los otros lenguajes es que permite integrarse con clases Java (.class) lo que permite separar en niveles las aplicaciones web, almacenando en clases java las partes que consumen más recursos (así como las que requieren más seguridad) y dejando la parte encargada de formatear el documento HTML en el archivo JSP. La idea fundamental detrás de este criterio es la de separar la lógica del negocio de la presentación de la información.

Los servlets y JSPs (Java Server Pages) son dos métodos de creación de páginas web dinámicas de lado de servidor usando el lenguaje Java. Los JSPs son en realidad servlets: un JSP se compila a un programa en Java la primera vez que se invoca, y del programa en Java se crea una clase que se empieza a ejecutar en el servidor como un servlet. La principal diferencia entre los servlets y los JSPs es el enfoque de la programación: un JSP es una página web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página web.

Cada servlet o JSP se ejecuta en su propio hilo, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo (cargar programa + intérprete). Su persistencia le permite también hacer más eficiente la conexión a la base de datos y el manejo de sesiones.

Los JSPs y los servlets se ejecutan en una máquina virtual Java, lo que permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que tengan instalada una **máquina virtual Java**. En el ANEXO 2: PROCEDIMIENTO DE INSTALACIÓN se indica como instalarla y se adjunta con el proyecto [23]. Por otra parte, para ejecutar los servlets y las páginas JSP, se necesita un servidor Web con un contenedor Web que cumpla las especificaciones de JSP y de Servlet. **Tomcat** es una completa implementación de referencia que cumple estas especificaciones. También en el Anexo 2 hay una guía de cómo instalarlo como una extensión del kit XAMPP [25].

Una vez se ha seleccionado y se conoce todo lo necesario para crear una web dinámica, los requisitos de este proyecto hacen necesaria además de la utilización de JSP la utilización de AJAX, ya que aparte de cargar el contenido dinámicamente, se debe cargar a tiempo a real, es decir, la página debe actualizarse constantemente sin que el usuario intervenga. AJAX [13] no es una tecnología, sino la unión de varias tecnologías que juntas pueden lograr cosas realmente impresionantes como GoogleMaps, Gmail o algunas otras aplicaciones muy conocidas. AJAX en resumen, es el acrónimo para Asynchronous JavaScript + XML y el concepto en el que se basa es cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que se necesitan para actualizar la página y sólo re-renderizar las porciones de la página que se necesite, sin recargar la página completamente.

La utilización de AJAX en este proyecto se realiza a partir del empleo de 2 frameworks, uno para crear pestañas que permitan moverse entre distintas páginas web manteniendo una parte de la página constante, y otro para actualizar las porciones de la página a las que afectan los cambios sin intervención del usuario, de forma automática. Veamos primero qué son los framework, ya que es un concepto novedoso.

Un framework [14], que literalmente se traduce del inglés como marco de trabajo, es un conjunto de clases que proporcionan un soporte o estructura base sobre la que adaptar el desarrollo de otras aplicaciones. Se trata de un esqueleto de aplicación que abstrae un conjunto de herramientas y servicios que son comunes a multitud de aplicaciones.

Trasladado al caso de las aplicaciones web, un framework nos proporciona un conjunto de clases que resuelven tareas comunes a todas las aplicaciones web, como por ejemplo, la gestión de peticiones y respuestas entre el cliente y el servidor web.

El uso de frameworks permite centrar el esfuerzo del diseño en resolver el problema del dominio que se plantee, sin tener que volver a diseñar aspectos de las aplicaciones web que son comunes a todas, y que ya han sido resueltos.

Los frameworks se diferencian de las librerías en que estas últimas son un conjunto de clases a las que recurre una aplicación, mientras que en el caso de los framework, las clases formarán parte del corazón del diseño de la aplicación.

El uso de frameworks para el desarrollo de aplicaciones web está de moda hoy en día debido a que permiten reutilizar código, y por tanto ahorrar tiempo y minimizar la cantidad de errores entre otras ventajas.

Los 2 frameworks que se han utilizado en este proyecto son:

PROTOTYPE [15]

Para que la aplicación web se actualice de forma automática sin requerir que el usuario tenga que pinchar en el botón de actualización, se utiliza el framework *Prototype*. Así, cuando se registra una nueva acción en un partido aparece en la pantalla sin intervención del usuario, refrescándose todas las porciones de la página que sean necesarias; por ejemplo, si ha metido una canasta un jugador del equipo local, se debe actualizar todo menos la porción donde aparecen los jugadores del equipo visitante.

Prototype es un framework muy útil para facilitar el desarrollo sencillo y dinámico de aplicaciones web con JavaScript y AJAX. Su autor original es Sam Stephenson, aunque las últimas versiones incorporan código e ideas de muchos otros programadores. A pesar de que incluye decenas de utilidades, la librería es compacta y está programada de forma muy eficiente. En este proyecto su utilización es mínima, pero la importancia de sus resultados es máxima, ya que recargar la página de forma automática es un requisito básico de funcionalidad del proyecto, porque no tendría sentido que el visitante de la web tuviese cada 2 segundos que actualizar la página manualmente para conocer las nuevas acciones que han sucedido.

Esta funcionalidad la aporta una única función del framework *Prototype* y se denomina *Ajax.PeriodicalUpdater*. Los parámetros del constructor de la función son:

- ID del elemento HTML a rellenar.
- URL de la petición.
- Opciones específicas:
 - **frequency** (entero) - Tiempo en segundos entre peticiones.
 - **decay** (entero) - El valor de **frequency** se multiplica por éste cada vez que se recibe una respuesta sin modificaciones respecto a la anterior.

Así pues, esta función realiza cada *frequency* segundos una petición AJAX y rellena el elemento HTML indicado con la respuesta recibida.

SPRY [16]

Para poder cambiar entre las páginas web que muestran la retransmisión del partido, y las que muestran las estadísticas de los jugadores y las estadísticas de los equipos, simulando el comportamiento de la aplicación de escritorio, se decidió que lo más conveniente era navegar por estas páginas mediante el uso de pestañas. Sin embargo, la implementación de pestañas en la web no es sencilla, por lo que buscando apareció el framework *Spry*, que permite entre otras muchas aplicaciones, la utilización de pestañas junto al empleo de Ajax.

Este empleo de Ajax produce que se pueda navegar entre las distintas páginas web sin necesidad de recargar la página entera, sólo cambiando la pestaña seleccionada que contiene cada página web. Ésto es muy ventajoso para este proyecto ya que se va a poder seguir viendo el resultado del partido en la parte superior de la página a la vez que se va cambiando de pestaña consultando las estadísticas de los jugadores, la retransmisión del partido o las estadísticas de los equipos.

Por tanto, se utiliza el widget de *Paneles de Pestañas* del framework Spry [17], que consiste en un conjunto de paneles que pueden almacenar contenido web en un espacio compacto. Los navegadores esconden o revelan el contenido almacenado en los paneles si se selecciona la pestaña del panel al que se quiere acceder, pero sólo uno de los paneles puede estar abierto al mismo tiempo.

En general, el framework Spry es una librería Javascript para diseñadores web a quienes proporciona funcionalidades para construir páginas web que provean una experiencia enriquecedora a sus usuarios. Fue diseñado para acoger Ajax en la comunidad de diseño web supliendo carencias en este campo que otros frameworks no cumplían. Es un framework centrado en HTML, y fácil de utilizar para usuarios con unos conocimientos básicos de HTML, CSS y Javascript.

Por último, hacer referencia a que la aplicación web, como ya se comentó en el apartado anterior de arquitectura del sistema, también debe acceder a una base de datos, de donde extraer la información que presenta dinámicamente en formato web. Esta base de datos va a ser la misma que utiliza la aplicación de escritorio, y por tanto, el sistema gestor de bases de datos que la controla es MySQL. Las conexiones desde JSP también se manejan gracias al API JDBC.

Una vez estudiadas las tecnologías utilizadas tanto en la aplicación web como en la aplicación de escritorio, se comentan también algunas herramientas que se han utilizado para la realización de este proyecto:

- *Netbeans 5.5* ➔ Herramienta CASE utilizada en el diseño e implementación de la aplicación Swing y de la aplicación web JSP.
- *Macromedia Dreamweaver* ➔ Herramienta utilizada en el diseño HTML de la aplicación web.
- *XAMPP* ➔ Kit de herramientas independiente de plataforma, software libre, que consiste principalmente en el gestor de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. También se utiliza una extensión consistente en el servidor Tomcat 6.0.13 que es capaz de servir páginas JSP.
- *Pacestar UML Diagrammer 5.08* ➔ Herramienta utilizada en el diseño UML de las clases que sustentan la aplicación de escritorio y la aplicación web.
- *DER Editor* ➔ Herramienta multiplataforma que permite realizar diseños de diagramas entidad-relación y generar su correspondiente esquema lógico en lenguaje SQL/92. El desarrollo de esta herramienta lo llevó a cabo un alumno de la EUPT (Escuela Universitaria Politécnica de Teruel) como defensa de su Trabajo Final de Carrera.

2.2 DISEÑO

2.2.1 Definición del nuevo lenguaje de comandos

La mayor innovación de este proyecto respecto al estado del arte en su ámbito, consiste, como ya se indicó anteriormente, en la forma de introducir las acciones que suceden en un partido de baloncesto. En este proyecto, se define una nueva forma de introducir las acciones a través del teclado a partir de un lenguaje que se define a continuación.

Este lenguaje está formado por un conjunto de categorías léxicas que se muestran en la *Tabla 2*: especificadas a partir de su correspondiente expresión regular. De estas categorías léxicas cabe destacar que las expresiones regulares que contienen caracteres de la 'a' la 'z', se han diseñado de forma flexible para que admitan éstos caracteres tanto en mayúsculas, como en minúsculas, como en una combinación de ambas. Así el comando `ftm`, `FTM` ó `fTm` tienen el mismo significado y esto proporciona al usuario una mayor comodidad, ya que el sistema no considerará como error un fallo al escribir un comando en mayúsculas o minúsculas. En otros ámbitos sí tiene sentido diferenciar entre mayúsculas y minúsculas, pero no es el caso de este proyecto. Para implementar esto, simplemente se transforma toda la entrada a minúsculas y se compara ésta con los comandos formados por todo minúsculas.

<i>Categoría léxica</i>	<i>Descripción</i>	<i>Expresión Regular</i>
blancos	Blancos	[\\t ' ']
num	Número jugador	[0-9] [1-9][0-9]
inicio	Inicio del partido	[i I]
inicio_1	Inicio del primer cuarto	[i I] 1
inicio_2	Inicio del segundo cuarto	[i I] 2
inicio_3	Inicio del tercer cuarto	[i I] 3
inicio_4	Inicio del último cuarto	[i I] 4
fin	Fin del partido	[f F]
fin_1	Fin del primer cuarto	[f F] 1
fin_2	Fin del segundo cuarto	[f F] 2
fin_3	Fin del tercer cuarto	[f F] 3
fin_4	Fin del último cuarto	[f F] 4
finTiempoM	Fin del tiempo muerto	[f F][t T][m M]
error	Error: Elimina la última acción	[e E]
asistencia	Asistencia	[a A]
reboteA	Rebote en ataque	[r R][a A]
reboteD	Rebote en defensa	[r R][d D]
cambioEntra	Cambio Entrante	[c C][e E]
cambioSale	Cambio Saliente	[c C][s S]
canasta1	Tiro libre convertido	[c C] 1
canasta2	Tiro convertido de 2 puntos	[c C] 2
canasta3	Triple convertido	[c C] 3
fallo1	Tiro libre fallado	[t T] 1
fallo2	Tiro fallado de 2 puntos	[t T] 2
fallo3	Triple fallado	[t T] 3
tacon	Tapón	[t T][a A]
tiempoM	Tiempo muerto	[t T][m M]
robo	Robo de balón	[b B][r R]
perdida	Pérdida de balón	[b B][p P]
faltaP	Falta personal	[f F][p P]
faltaR	Falta recibida	[f F][r R]
punto	Acción del equipo contrario	.

Tabla 2: Categorías léxicas del nuevo lenguaje definido

Todas las expresiones regulares vistas anteriormente, tienen asociada una acción que excepto para el caso de la categoría léxica **blancos** donde se omitirán o eliminarán, para el resto emitirán el correspondiente componente léxico. Para reflejar mejor estas expresiones regulares junto a sus acciones asociadas se emplea una máquina discriminadora determinista (MDD) que se puede ver en la *Figura 17*:

La implementación de este tipo de máquina, permite la segmentación empleando una *estrategia avariciosa* del conjunto de acciones correspondientes a un partido que se introducen como entrada. De esta forma, en cada momento se busca el prefijo más largo de la entrada no analizada que pertenezca a alguna categoría léxica. Si en un determinado momento no se ha transitado a ningún estado final está sucediendo un error. Así, por ejemplo, la entrada i12a, significaría inicio del primer cuarto del partido y asistencia del jugador del equipo local número 2, mientras que si no se siguiese esta estrategia, esa entrada significaría inicio del partido y asistencia del jugador 12 del equipo local.

Por tanto, el analizador léxico realiza una completa detección de errores léxicos. Cada error léxico se describe mediante un mensaje de error (en cursiva) que mostrará la aplicación de escritorio una vez implementada. Sólo se indican los caracteres en minúsculas para hacerlo más sencillo, pero también estarían involucrados los mismos caracteres en mayúsculas.

- Si se introduce cualquier carácter al principio que no sea un espacio, tabulador, un número del 0 al 9, o los caracteres 'i', 'f', 'c', 't', 'r', 'b', 'a', '.', ó 'e', que son los caracteres que permiten transitar del estado 0 de la máquina discriminadora a cualquier estado relacionado con el estado 0 ➔ *"Inicio de acción no valido: carácter x"*, donde x es el carácter erróneamente introducido.
- Si tras los caracteres 'ft' (estado 11) de la máquina discriminadora, se introduce cualquier carácter que no sea una 'm', se produce un error, porque no se puede transitar desde este estado. Por tanto, si que se ha pasado por el estado final de la f, pero se considera que al haber puesto la m no se quería indicar el final de partido, y se considera un error. ➔ *"Falta la m para finalizar la acción ftm"*.
- Si tras el carácter 'c' (estado 19) de la máquina discriminadora, se introduce cualquier carácter que no sea 'e', 's', '1', '2' ó '3', no se podrá transitar a ningún estado y no se habrá llegado a ningún estado final por lo que será un error. ➔ *"Acción desconocida: cx"*, donde x es el carácter erróneamente introducido.

- Si tras el carácter 't' (estado 25) de la máquina discriminadora, se introduce cualquier carácter que no sea 'a', 'm', '1', '2' ó '3', no se podrá transitar a ningún estado y no se habrá llegado a ningún estado final por lo que será un error. ➔ "*Acción desconocida: tx*", donde x es el carácter erróneamente introducido.
- Si tras el carácter 'r' (estado 31) de la máquina discriminadora, se introduce cualquier carácter que no sea 'a', ó 'd' no se podrá transitar a ningún estado y no se habrá llegado a ningún estado final por lo que será un error. ➔ "*Acción desconocida: rx*", donde x es el carácter erróneamente introducido.
- Si tras el carácter 'b' (estado 34) de la máquina discriminadora, se introduce cualquier carácter que no sea 'p', ó 'r' no se podrá transitar a ningún estado y no se habrá llegado a ningún estado final por lo que será un error. ➔ "*Acción desconocida: bx*", donde x es el carácter erróneamente introducido.

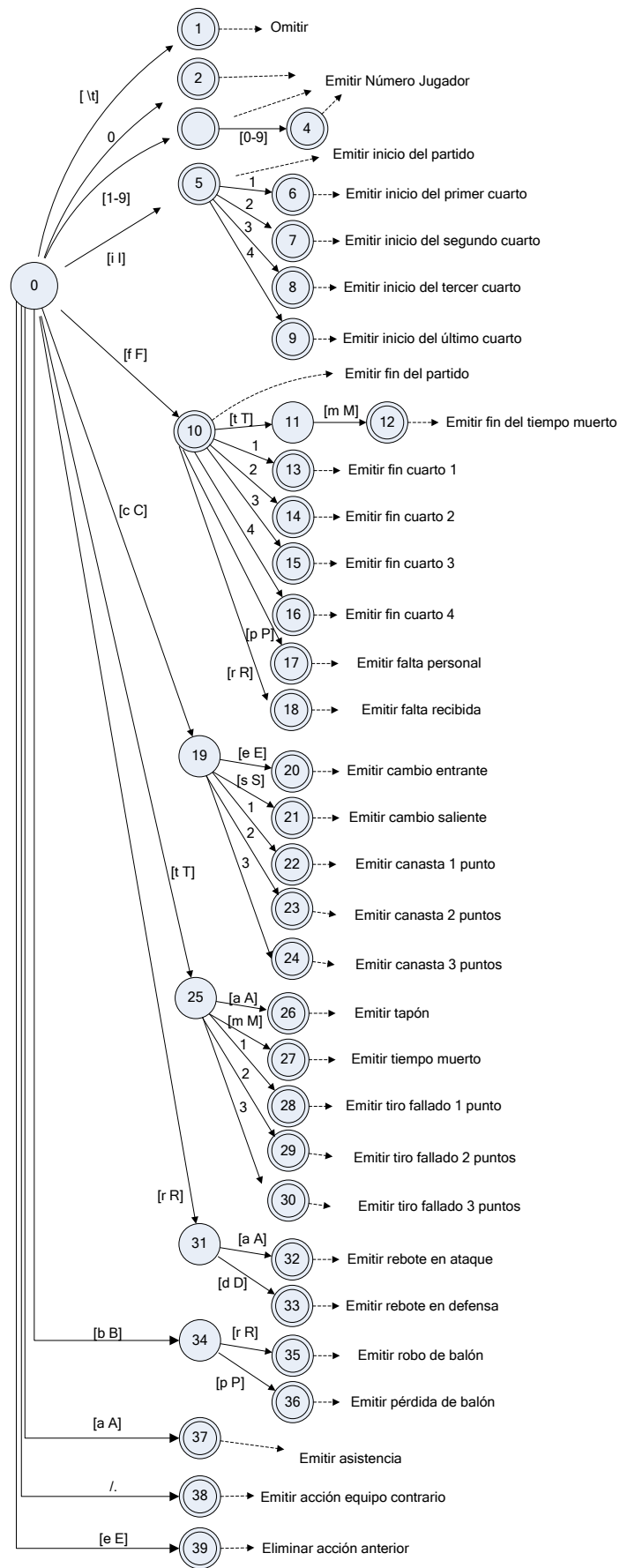


Figura 17: MDD (Máquina Discriminadora Determinista)

Para especificar este nuevo lenguaje se utiliza una gramática incontextual basada en los componentes léxicos ya mencionados anteriormente. En ella, los componentes léxicos se expresan en negrita, siendo los elementos terminales de la misma, y los elementos no terminales se expresan entre $\langle \rangle$.

```

<Partido> → inicio <Cuarto1><Cuarto2><Cuarto3><Cuarto4> fin
<Cuarto1> → inicio_1 <ListaAcciones> fin_1
<Cuarto2> → inicio_2 <ListaAcciones> fin_2
<Cuarto3> → inicio_3 <ListaAcciones> fin_3
<Cuarto4> → inicio_4 <ListaAcciones> fin_4
<ListaAcciones> → <Accion><ListaAcciones> | <Accion>
<Accion> → punto? num <Comando> <Accion> | punto? num <Comando> |
           error | tiempoM | finTiempoM
<Comando> → reboteA | reboteD | cambioSale punto? num cambioEntra |
           asistencia | canasta1 | canasta2 | canasta3 | fallo1 |
           fallo2 | fallo3 | tapon | robo | perdida | faltaP | faltaR

```

Esta gramática que acabamos de exponer es ambigua. Para eliminar su ambigüedad, se eliminan los prefijos comunes y la gramática incontextual queda de la siguiente forma.

```

<Partido> → inicio <Cuarto1><Cuarto2><Cuarto3><Cuarto4> fin
<Cuarto1> → inicio_1 <ListaAcciones> fin_1
<Cuarto2> → inicio_2 <ListaAcciones> fin_2
<Cuarto3> → inicio_3 <ListaAcciones> fin_3
<Cuarto4> → inicio_4 <ListaAcciones> fin_4
<ListaAcciones> → <Accion><LA>
<LA> → <ListaAcciones> | λ
<Accion> → punto? num <Comando> <A> | error | punto? tiempoM |
           punto? finTiempoM
<A> → <Accion> | λ
<Comando> → reboteA | reboteD | cambioSale punto? num cambioEntra |
           asistencia | canasta1 | canasta2 | canasta3 | fallo1 |
           fallo2 | fallo3 | tapon | robo | perdida | faltaP | faltaR

```

Esta gramática establece las siguientes reglas a cumplir por las acciones que formen este lenguaje:

- El partido se inicia cuando se introduce el componente **inicio** y no finaliza hasta que se introduce el componente **fin**.
- Entre el componente inicio y fin deben introducirse las acciones de los 4 cuartos de que consta un partido en orden. Es decir, primero el primer cuarto, que se inicia con el componente **inicio_1** y finaliza con el componente **fin_1**, después el segundo cuarto, y así sucesivamente hasta el último cuarto siguiendo una estructura análoga.
- Cada cuarto entre sus componentes de inicio y fin puede poseer un conjunto de acciones, y a su vez cada acción puede ser compuesta de varias acciones. Un ejemplo sería:

```
i1
  23ta
    .10t2
      12br8t39ra9c2
f1
```

En el primer cuarto se introducen 3 acciones, de las cuáles la tercera es compuesta. En la primera acción el jugador 23 del equipo local realiza un tapón. En la segunda acción el jugador 10 del equipo visitante falla un tiro de 2 puntos. En la tercera acción, que es compuesta, el jugador 12 del equipo local roba un balón, el jugador 8 del equipo local falla un triple, el rebote en ataque lo coge el jugador 9 del equipo local y este mismo jugador 9 del equipo local anota una canasta de 2 puntos.

- Cada acción por sí misma está compuesta de un posible componente **punto**, que si aparece indica que la acción es del equipo visitante y si no aparece indica que la acción es del equipo local, el dorsal del jugador que realiza la acción y un comando que representa la acción (asistencia, rebote en ataque, canasta de 2 puntos, etc.). Si la acción es un cambio, se obliga a que haya dos acciones seguidas: una para indicar el cambio saliente y otra para indicar el cambio entrante. Habrá que detectar que en ambos casos los jugadores sean del equipo local o del equipo visitante.

También una acción puede ser simplemente cualquiera de los componentes **error**, **tiempoM** o **finTiempoM** para los cuáles no haría falta indicar el jugador que los realiza ya que los realiza el equipo (tiempoM y finTiempoM) o el estadígrafo si se equivoca al introducir una acción (error) .

Una vez especificada la gramática que define el lenguaje, su implementación directa no fue posible. La causa es que durante un partido las acciones se van introduciendo poco a poco, no existe un archivo en el que ya estén reflejadas todas las acciones, por tanto, se conocen las acciones que han sucedido hasta un determinado momento, pero no las que van a suceder después. En este contexto, se puede estudiar si una nueva acción cumple la sintaxis conforme a lo existente hasta el momento, pero no se puede realizar un completo análisis sintáctico que garantice que todas las acciones del proyecto se han introducido de forma correcta, se ha de ir verificando acción por acción. Resumiendo, la especificación de la gramática permite que el usuario conozca en qué orden debe introducir las acciones, qué reglas debe seguir, pero también sirve para que, en la medida de lo posible, la aplicación detecte fallos sintácticos producidos por la entrada de nuevas acciones y le avise al usuario facilitándole su correcta introducción.

Algo que hay que destacar de esta gramática es que permite introducir acciones compuestas, es decir, que permite introducir de una sola vez varias acciones que han sucedido en el partido. Ésta circunstancia no la permite ninguno de los sistemas de estas características que existen hoy en día, ya que en ellos, la interfaz se basa en el uso del ratón en vez del teclado y se deben ir introduciendo las acciones una a una. Los sistemas actuales, en función de la acción introducida intentar ayudar al usuario a introducir la siguiente acción cuando es muy frecuente encontrar esas acciones seguidas. Ésto que en principio es una ayuda, muchas veces acaba convirtiéndose en un incordio, ya que a veces no es esa acción la que se quiere introducir, y por tanto hay que cancelar e introducir la nueva acción. Todo esto requiere un tiempo valiosísimo, en el que el estadígrafo va a perder el contacto visual con el partido y posiblemente va a dejar de introducir algunas acciones que sucedan porque no las ha visto.

Este proyecto, gracias a esta gramática, permite introducir varias acciones simultáneamente a través del teclado, con lo que consigue limitar el tiempo en que se pierde el contacto visual con el partido, y a su vez consigue una mayor rapidez a la hora de introducir acciones, permitiendo también introducir varias acciones simultáneamente.

Los errores sintácticos que pueden aparecer según esta gramática son los siguientes, identificados por su correspondiente mensaje de error (en cursiva).

- Si se introduce la acción 'i' y ésta ya había sido introducida → *"El partido ya ha sido iniciado y no puede volver a iniciarlo"*
- Si se introduce cualquier acción y todavía no se había introducido la acción 'i' → *"Todavía no se ha iniciado el partido. Introduzca el comando 'i'".*
- Si se introduce la acción 'i1' pero todavía no se ha recibido la acción de inicio del partido → *"Debe iniciar el partido mediante el comando 'i' antes de iniciar el primer cuarto".*
- Si ya se ha recibido la acción de inicialización de cualquier cuarto y se recibe la acción 'i1' → *"Actualmente se está disputando el cuarto x del partido, y no puede volver a iniciarse el primer cuarto",* siendo x el número de cuarto que se esté disputando en ese momento.
- Si se recibe la acción de inicialización del 2º cuarto 'i2' antes de haber recibido la acción de inicialización del primer cuarto 'i1' → *"Todavía no se ha jugado el primer cuarto y no se puede iniciar el 2º cuarto. Introduzca el comando 'i1' para iniciar el primer cuarto".*
- Si ya se han recibido las acciones de inicialización de al menos hasta el 2º cuarto y se recibe la acción 'i2' → *"Actualmente se está disputando el cuarto x del partido, y no puede volver a iniciarse el segundo cuarto",* siendo x el número de cuarto que se esté disputando en ese momento.
- Si se recibe la acción de inicialización del 3º cuarto 'i3' antes de haber recibido la acción de inicialización del segundo cuarto 'i2' → *"Todavía no se ha jugado el segundo cuarto y no se puede iniciar el tercer cuarto. Introduzca el comando 'i2' para iniciar el segundo cuarto".*

- Si ya se han recibido las acciones de inicialización de al menos hasta el 3º cuarto y se recibe la acción 'i3' → *"Actualmente se está disputando el cuarto x del partido, y no puede volver a iniciarse el tercer cuarto"*, siendo x el número de cuarto que se esté disputando en ese momento.
- Si se recibe la acción de inicialización del último cuarto 'i4' antes de haber recibido la acción de inicialización del tercer cuarto 'i3' → *"Todavía no se ha jugado el tercer cuarto y no se puede iniciar el último cuarto. Introduzca el comando 'i3' para iniciar el tercer cuarto"*.
- Si ya se han recibido las acciones de inicialización de todos los cuartos y se recibe la acción 'i4' → *"Actualmente ya se está disputando el último cuarto del partido, y no puede volver a iniciarse"*.
- Si se recibe la acción de finalización del primer cuarto 'f1' antes de haber recibido la acción de inicialización del primer cuarto 'i1' → *"Todavía no se ha iniciado ningún cuarto, y por tanto no es posible finalizar el primer cuarto"*.
- Si se recibe la acción de finalización del primer cuarto 'f1' si se está disputando el 2º, 3º o último cuarto → *"Actualmente se está disputando el cuarto x del partido, y por tanto no es posible finalizar el primer cuarto"*, donde x es el número de cuarto que se está disputando en ese momento.
- Si se recibe la acción de finalización del segundo cuarto 'f2' antes de haber recibido la acción de inicialización del primer cuarto 'i1' → *"Todavía no se ha iniciado ningún cuarto, y por tanto no es posible finalizar el segundo cuarto"*.
- Si se recibe la acción de finalización del segundo cuarto 'f2' si se está disputando cualquier cuarto que no sea el segundo → *"Actualmente se está disputando el cuarto x del partido, y por tanto no es posible finalizar el segundo cuarto"*, siendo x el número de cuarto que se está disputando en ese momento.
- Si se recibe la acción de finalización del tercer cuarto 'f3' antes de haber recibido la acción de inicialización del primer cuarto 'i1' → *"Todavía no se ha iniciado ningún cuarto, y por tanto no es posible finalizar el tercer cuarto"*.

- Si se recibe la acción de finalización del tercer cuarto 'f3' si se está disputando cualquier cuarto que no sea el tercero → *"Actualmente se está disputando el cuarto x del partido, y por tanto no es posible finalizar el tercer cuarto"*, siendo x el número de cuarto que se está disputando en ese momento.
- Si se recibe la acción de finalización del tercer cuarto 'f4' antes de haber recibido la acción de inicialización del primer cuarto 'i1' → *"Todavía no se ha iniciado ningún cuarto, y por tanto no es posible finalizar el último cuarto"*.
- Si se recibe la acción de finalización del último cuarto 'f4' si se está disputando cualquier cuarto que no sea el último → *"Actualmente se está disputando el cuarto x del partido, y por tanto no es posible finalizar el último cuarto"*, siendo x el número de cuarto que se está disputando en ese momento.
- Según la gramática, no puede finalizar el partido si todavía no se ha jugado y finalizado el último cuarto, sin embargo esta condición se rebaja y un partido puede finalizarse en cualquier momento ya que a veces hay circunstancias que obligan a ello y este sistema debe estar preparado para ello. Por tanto, no se considerará error introducir la acción final de partido en cualquier momento.
- Si se recibe una acción de tiempo muerto 'tm' o final de tiempo muerto 'ftm' antes de haber recibido la acción de iniciar el primer cuarto 'i1' → *"Todavía no se ha iniciado el primer cuarto. Introduzca el comando 'i1' para iniciarlo"*.
- Si una acción, independientemente de si comienza con el componente punto o no, es alguna de las acciones que realiza un jugador y no comienza con el dorsal del jugador sino que empieza con la acción → *"Una acción se debe iniciar con el dorsal del jugador que la ha realizado. De la siguiente forma => [.] dorsal_jugador accion"*.
- Si se introduce una acción cualquiera de las que realiza un jugador antes de recibir la acción de inicio del partido 'i' → *"Todavía no se ha iniciado el partido. Introduzca el comando 'i'"*.
- Si se introduce una acción cualquiera de las que realiza un jugador antes de recibir la acción de inicio del primer cuarto 'i1' → *"Todavía no ha inicializado ningún cuarto. Introduzca el comando 'i1'"*.

- Si se introduce una acción que no sea cambio entrante 'ce' si la última acción introducida era de cambio saliente 'cs' → *"Tras una acción de cambio saliente 'cs' debe aparecer una acción de cambio entrante 'ce'"*.
- Si se introduce una acción de cambio entrante 'ce' antes que una acción de cambio saliente 'cs', o se introducen los dorsales de los jugadores después de los comandos → *"La acción de cambio debe seguir la siguiente sintaxis: [.] num_jugador cs [.] num_jugador ce"*.

Algunas comprobaciones semánticas que no podían modelarse en la gramática incontextual, producen los siguientes casos de error.

- Si la última acción recibida es un tiempo muerto 'tm', y se vuelve a recibir una acción de tiempo muerto → *"El partido ya se encuentra en tiempo muerto y por tanto no se puede iniciar un nuevo tiempo muerto"*.
- Si se recibe la acción final de tiempo muerto 'ftm', y la anterior acción no es tiempo muerto 'tm' → *"El partido no se encuentra en tiempo muerto y por tanto no se puede finalizar el tiempo muerto"*.
- Si se introduce una acción cualquiera de las que realiza un jugador, pero el jugador que la realiza ya ha realizado 5 faltas personales y está todavía en el equipo titular → *"El jugador x ha sido expulsado del partido y no puede intervenir en ninguna acción. Cámbielo del equipo titular."*, siendo x el nombre del jugador que realiza la acción.
- Si se introduce una acción cualquiera de las que realiza un jugador, pero el jugador que la realiza no está disputando el partido en ese momento, es decir, está en el banquillo → *"El jugador x se encuentra en el banquillo y por tanto no puede efectuar ninguna acción"*, siendo x el nombre del jugador que realiza la acción.

- Si se introduce una acción cualquiera de las que realiza un jugador, pero el dorsal del jugador que la realiza no pertenece a ningún jugador del equipo local si en la acción no aparece el componente punto, o no pertenece a ningún jugador del equipo visitante si aparece el componente punto → *"El equipo x no posee ningún jugador con el dorsal z"*, siendo x el nombre del equipo local si no aparece el componente punto al principio de la acción, o el nombre del equipo visitante si aparece el componente punto al principio de la acción, y siendo z el número del dorsal del jugador que supuestamente realiza la acción.
- Si se introduce una acción de cambio entrante después de una acción de cambio saliente, y el jugador que va a entrar ya se encuentra disputando el partido → *"Se ha producido un error en el cambio de jugador. El jugador x ya está jugando"*, siendo x el nombre del jugador que supuestamente iba a entrar al campo.
- Si se introduce una acción de cambio entrante después de una acción de cambio saliente, y el jugador que va a entrar se encuentra en el banquillo pero posee 5 faltas personales → *"Se ha producido un error en el cambio de jugador. El jugador x posee 5 faltas y está expulsado del partido"*, siendo x el nombre del jugador que supuestamente iba a entrar al campo.
- Si se introduce una acción de cambio saliente y después de ella una acción de cambio entrante, pero el jugador que va a salir no se encuentra disputando el partido, es decir, está en el banquillo → *"Se ha producido un error en el cambio de jugador. El jugador x no está jugando y por tanto no puede ser cambiado."*, siendo x el nombre del jugador que supuestamente iba a entrar al campo.

Algunos de estos errores léxicos, sintácticos y semánticos simplemente se tratarán no procesando la última acción enviada, sin embargo otros requerirán la eliminación de la última acción tratada y la correspondiente actualización de otros datos en la base de datos.

2.2.2 Diseño de las Clases

Para el diseño de las clases que modelan el sistema se utiliza el patrón de arquitectura de software denominado **Modelo Vista Controlador (MVC)** [3], que separa en tres componentes distintos los datos de una aplicación, la interfaz de usuario, y la lógica de control. La ventaja más importante relacionada con el uso de este patrón es la ordenación de la estructura de la aplicación y la separación completa de la presentación (vista), de los datos (modelo), y de la lógica de negocio (controlador).

El patrón MVC se utiliza frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio.

Se analizan ahora más en detalle cada uno de los 3 componentes (ver *Figura 18:*):

- **Modelo:** Es la representación específica de la información con la que opera el sistema. La lógica de datos asegura la integridad de éstos y permite derivar nuevos datos.
- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente se conoce como *interfaz de usuario*.
- **Controlador:** Responde a eventos, normalmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

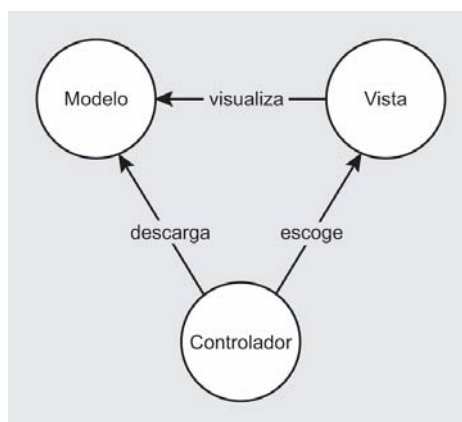


Figura 18: Relación entre los componentes del patrón MVC

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario (vista) de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.).
2. El controlador recibe (por parte de los objetos de la interfaz de usuario) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler).
3. El controlador accede al modelo, posiblemente actualizándolo de forma adecuada a la acción solicitada por el usuario.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejen los cambios del modelo. El modelo no debe tener conocimiento directo sobre la vista. El controlador no pasa objetos del modelo a la vista aunque puede dar la orden a la vista para que se actualice. En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Así pues, en este proyecto se diseñan las clases conforme a este patrón de arquitectura de software, con la intención de estructurar bien la aplicación, separar la gestión de los datos de la interfaz y de la lógica de negocio. Además este diseño permitirá reutilizar clases en ambas aplicaciones: la aplicación de escritorio y la aplicación web.

En el diagrama de clases que aparece en la *Figura 19:*, correspondiente al diseño de clases para la aplicación de escritorio, la **vista** es la clase denominada *Aplicación*. Sin embargo, para la aplicación web (ver *Figura 20:*), es la clase denominada *Aplicación Web*, que en realidad no es una clase, sino la página JSP *index.jsp*. Ambas tienen una relación con el modelo y el controlador.

La clase *Aplicación* posee gestores de eventos que actúan por ejemplo al presionar la tecla ENTER sobre el campo de texto que permite introducir las acciones de los partidos. Este gestor de eventos, recoge lo que ha enviado el usuario y lo pasa al controlador, quien lo procesa y envía el resultado al modelo, que es el encargado final de insertar las acciones del partido en la base de datos. Otros gestores de eventos menos importantes son los que detectan el foco sobre las distintas pestañas y cargan el contenido de cada una de ellas. Por la otra parte, la clase *Aplicación Web*, es decir, la página JSP *index.jsp*, posee un gestor de eventos que detecta cuando se recarga la página, y en base a ello, recupera la información de las acciones del partido. Esta información la envía al controlador, para que éste la procese actualizando las estadísticas del partido y devuelva esta información procesada a la interfaz, quien finalmente la muestra.

El **controlador** lo forman un conjunto de clases en ambos casos. Para el caso de la aplicación de escritorio, la clase *StringParser*, encargada de extraer las acciones que envía el estadígrafo a lo largo del partido, sería la clase principal del controlador, y ésta se encarga también de las relaciones con el modelo y con el resto de clases controladoras (Partido, Equipo, Jugador y Acción). La clase *StringParser* actúa para un partido en concreto, del que se registran las acciones y estadísticas que realizan 2 equipos en un lugar, una fecha y una hora concretos, dónde uno de los equipos actúa como local y el otro como visitante. Cada equipo está compuesto por un conjunto de jugadores, y esta clase almacena dinámicamente las estadísticas que está realizando ese equipo en ese partido. Cada jugador, por su parte, pertenece a un único equipo y también almacena las estadísticas que está realizando en ese partido. En la clase Jugador, no se necesitan guardar los puntos por cuarto ni las faltas por cuarto, sólo es necesario almacenar estas estadísticas para el partido completo. Sin embargo, para la clase Equipo, si se necesita guardar esa información separada por cuartos para conocer el tanteo de ambos equipos en cada cuarto, y para conocer el número de faltas de cada equipo en cada cuarto activando cuando corresponda el bonus.

Ambas clases almacenan dinámicamente la información de las estadísticas de cada jugador o cada equipo respectivamente. De esta forma, no es necesario almacenarlo en la base de datos.

Cada jugador puede realizar un conjunto de acciones en un partido, aunque también hay acciones en un partido que no son realizadas por ningún jugador (inicio del partido, error, tiempo muerto, etc.). Precisamente por esta causa, es necesario almacenar el equipo que realiza las acciones, porque por ejemplo, los tiempos muertos los realiza el equipo, no el jugador. Si no existieran acciones de este tipo, el equipo que ha realizado la acción no sería necesario almacenarlo, ya que podría obtenerse a partir del jugador que las ha realizado. El procesamiento de estas acciones, permite obtener las estadísticas de los jugadores y de los equipos, y de esta forma, la clase partido permite localizar al mejor jugador del encuentro mediante la operación *buscarMVP()*. Esta operación analiza qué jugador que ha disputado ese partido y tiene la mejor valoración. Cada objeto jugador permite calcular su valoración en ese partido a partir de la operación *getValoración()*, y ésta se calcula de la siguiente forma:

$$\begin{aligned}\textbf{Valoración jugador} = & \text{asistencias} + \text{rebotes_ataque} + \text{rebotes_defensa} \\ & + 2*\text{canasta2} + 3*\text{canasta3} + \text{canasta1} \\ & - 2*\text{tiros2} - 3*\text{tiros3} - \text{tiros1} + \text{balones_robados} \\ & - \text{balones_perdidos} + \text{tapones} - \text{faltas_personales} \\ & + \text{faltas_recibidas}\end{aligned}$$

El cálculo se basa en sumar las acciones “buenas” y restar las acciones “malas” que ha realizado el jugador a lo largo del partido. Los tiros fallados o canastas se ponderan según su valor.

El controlador para el caso de la aplicación web es prácticamente el mismo, pero no incluye la clase *StringParser*, ya que sólo hay que mostrar las acciones almacenadas en la base de datos que ya fueron procesadas por la aplicación de escritorio. Así, ahora la clase principal pasa a ser *Partido*, que debe englobar toda la información del mismo a través de sus relaciones con las otras clases (Jugador, Equipo y Acción) de la misma forma que lo hacía en el diseño de clases de la aplicación de escritorio. Únicamente se agrega la operación *reiniciarEstadísticas()* en las clases Equipo y Jugador, que cada vez que se recarga la página web, se ejecuta para que el sistema vuelva a calcular todas las estadísticas del partido y se las asigne a los equipos y jugadores. Esto es necesario debido a la existencia del comando "e", error, que elimina la última acción que el estadígrafo envió. Como esta acción puede englobar varias, como sería el caso de un cambio que engloba 2 acciones (actualizar jugador entrante y saliente), hay que recargar todas las acciones y calcular de nuevo todas las estadísticas, para asegurarse así de que no hay inconsistencias en la información.

El **modelo** gestiona una única conexión a la base de datos para todas las acciones (consulta, modificación, inserción o borrado) que se pretendan realizar sobre la misma. De esta forma, es más sencillo no dejar conexiones abiertas en la base de datos, y se ahorra tiempo de conexión para cada acción que le solicitan la aplicación de escritorio o la aplicación web. La aplicación de escritorio dispondrá de una única conexión, y cada cliente que se conecte mediante la aplicación web dispondrá de otra conexión. La implementación de esta clase Modelo se basa en instanciar un objeto de la clase **DriverManager** que se encarga de cargar inicialmente todos los drivers JDBC disponibles. Así se puede obtener un objeto de tipo conexión (**Connection**) con la base de datos, que aparece como el atributo *conn* de la clase Modelo. La conexión con la base de datos en particular, se establece siguiendo una sintaxis basada en la especificación más amplia de los URL, de la forma:

jdbc:subprotocolo//servidor:puerto/base de datos

Para este proyecto, al utilizar el sistema gestor de bases de datos MySQL, el nombre del subprotocolo será **mysql**. Esta URL se almacena también como atributo de la clase Modelo con el nombre *url*. Por otra parte, se obtiene la dirección IP del servidor y el puerto mediante el atributo *ip*, y el nombre de la base de datos desde el atributo *bd*. Una vez creado un objeto de tipo **Connection**, se pueden crear sentencias (**statements**) ejecutables. Cada una de estas sentencias puede devolver cero o más resultados dentro de un objeto de tipo **ResultSet** dependiendo de si es una consulta o una operación de inserción, borrado o modificación.

Tanto el statement como el resultset se almacenan también como atributos de la clase Modelo y se denominan *sentencia* y *resultado* respectivamente.

La operación *conectarBD()*, realiza básicamente la instanciación de estos atributos, preparando así la conexión a la base de datos. La operación contraria se denomina *cerrarBD()*, y su función es cerrar la conexión a la base de datos. El resto de operaciones se utilizan para recuperar, eliminar o insertar nueva información en la base de datos.

- *EliminarUltimaAccion()*, detecta y elimina la última acción que se introdujo en la base de datos. Se utiliza cuando el estadígrafo introduce el comando "e" y la última acción no fue un cambio. También actualiza el resto de datos que estén relacionados con esa acción.
- *EliminarAccionAnterior()*. Persigue el mismo objetivo que la anterior, pero en este caso elimina las 2 acciones anteriores, y es la operación utilizada cuando la última acción fue un cambio.
- *EliminarTodasEstadísticas()*. Realiza el borrado por completo de toda la tabla de acciones. Se utiliza porque este proyecto sólo gestiona un único partido, y cada vez que se inicia la aplicación, el partido no debe contener ninguna acción realizada.
- *insertarAccion()*. Dada una acción la introduce a la base de datos.
- *recuperarAcciones()*. Recupera todas las acciones que han sucedido en un partido hasta el momento en que se invoca la operación.

- *recuperarPartido()*. Recupera toda la información de un partido, incluyendo la información del partido, de los equipos, de los jugadores, pero no las acciones que han sucedido en el mismo.

Tanto para la aplicación web como para la aplicación de escritorio, la clase Modelo tiene los mismos atributos. Sin embargo, las operaciones de eliminación o inserción de información sólo se utilizan en la aplicación de escritorio. En la aplicación web no son necesarias ya que esta aplicación sólo se encarga de recuperar la información de la base de datos y presentarla en formato web a los visitantes.

Al ejecutar la aplicación web, se cargan los datos estáticos del partido (hora, fecha, lugar del partido, información de los equipos y jugadores, etc. pero no estadísticas) a través de la invocación de la operación *recuperarPartido()*. Posteriormente, cada vez que se recarga la página web, ésta información ya no es necesario recuperarla, porque no cambia, así que ya sólo es necesario recuperar la información de las acciones que van sucediendo a lo largo del partido a través de la operación *recuperarAcciones()*. Esta información de las acciones es procesada para generar la información estadística que se recoge en las clases Equipo y Jugador.

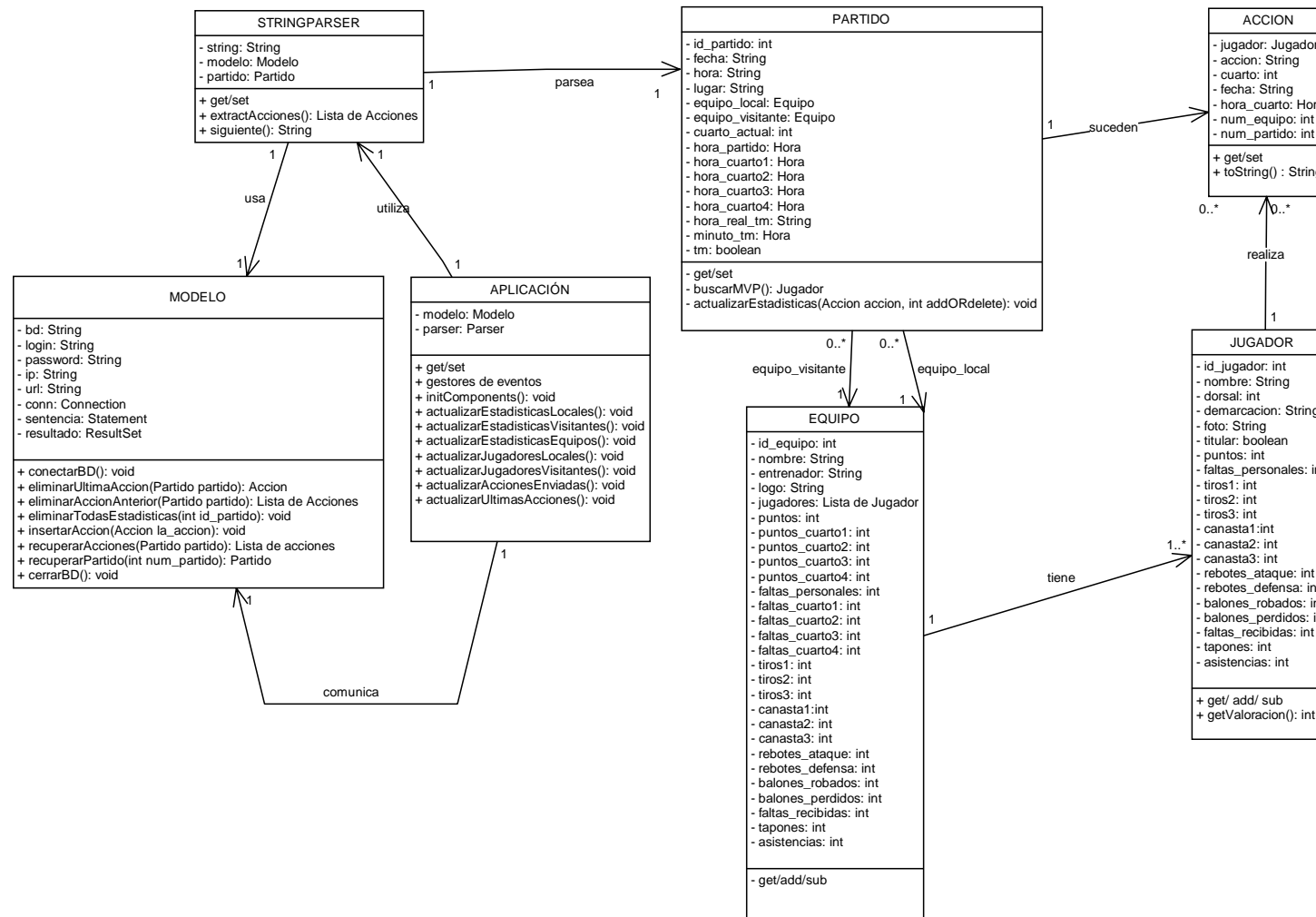


Figura 19: Diagrama de clases de la aplicación de escritorio

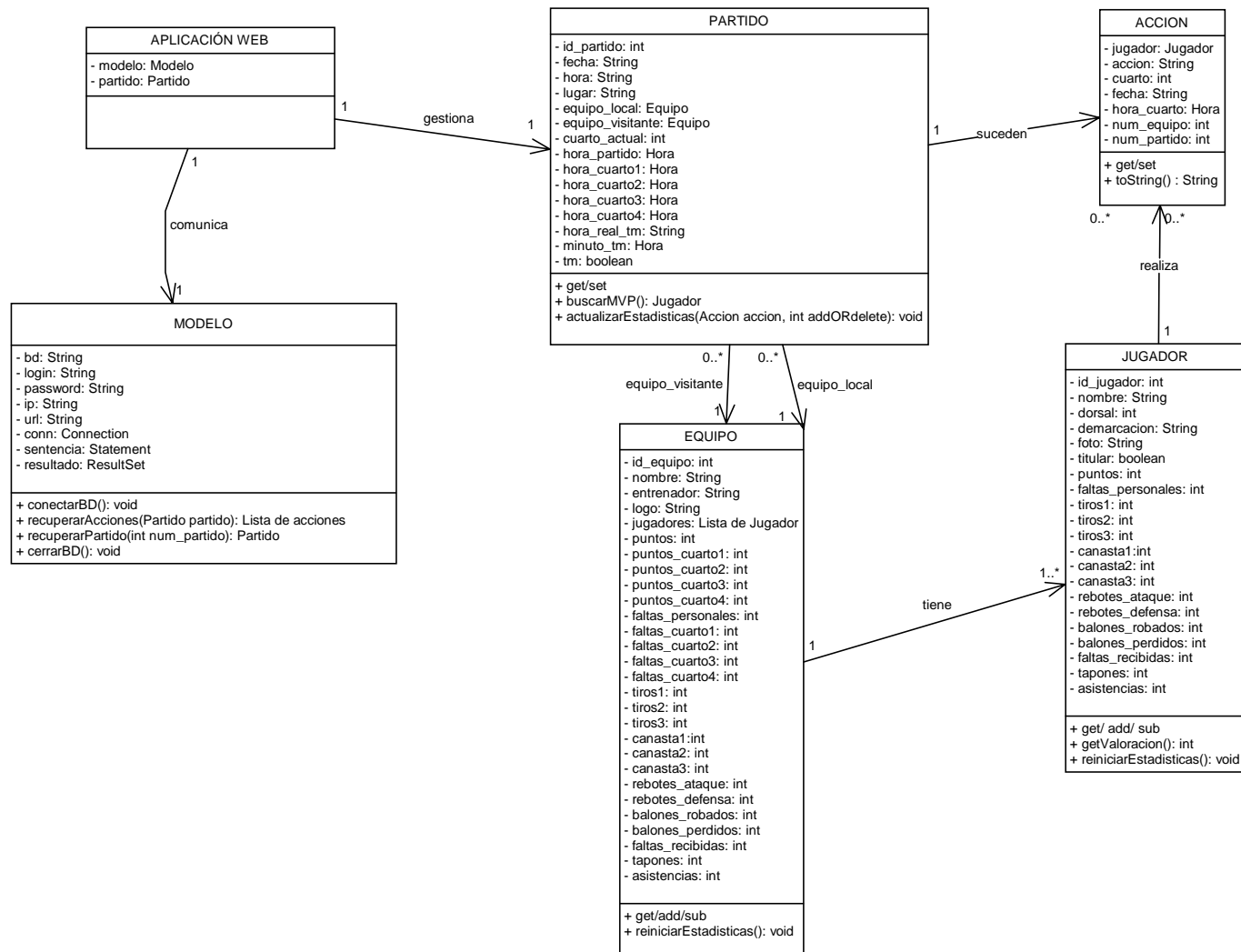


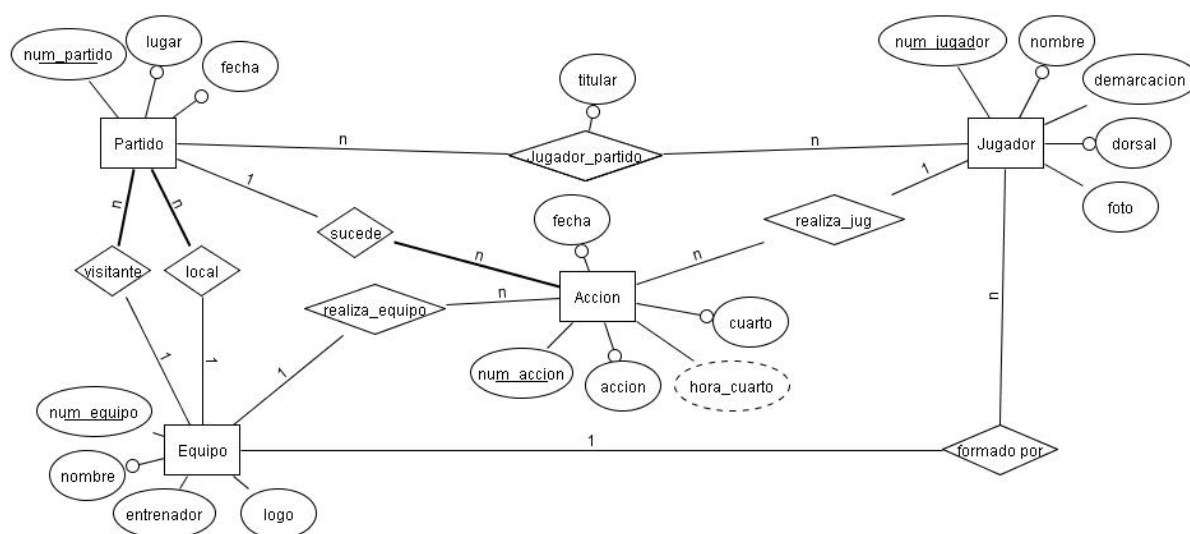
Figura 20: Diagrama de clases de la aplicación web

2.2.3 Diseño de la Base de Datos

Las características de los datos que se deben almacenar en este proyecto son las siguientes:

- Cada partido de baloncesto se realiza en un determinado estadio, en una determinada fecha y hora. En un partido se enfrentan un equipo que actúa como local y otro que actúa como visitante.
- De cada equipo se pretende almacenar al menos su nombre, entrenador, y escudo.
- Cada equipo cuenta con un conjunto de jugadores, de los que se necesita almacenar al menos su dorsal, su nombre, demarcación y fotografía.
- Para cada partido se convocan parte o el total de los jugadores de cada equipo, y se pretende conocer en cada momento si cada jugador convocado está disputando el partido en ese momento o está en el banquillo.
- Cada jugador puede realizar un conjunto de acciones (canasta de 2 puntos, rebote en ataque, asistencia, etc.) a lo largo de un partido, pero una acción también puede no ser realizada por ningún jugador; por ejemplo, un tiempo muerto lo solicita un equipo, y el inicio del partido, o de sus cuartos, son acciones no asociadas ni a un jugador ni a un equipo, sólo asociadas al partido.
- De cada acción se desea almacenar al menos la fecha y hora en que se realiza, la descripción de la acción, y el cuarto del partido en que se ha realizado.

El esquema entidad-relación que modela estos datos es el siguiente:



A continuación se detalla su traducción a esquema lógico:

PARTIDO (num_partido:integer, lugar:varchar(50), fecha:date,
local:integer, visitante:integer)

CP: { num_partido }

CAj: { local } hace referencia a Equipo

CAj: { visitante } hace referencia a Equipo

VNN: { local, visitante, lugar, fecha }

EQUIPO (num_equipo: integer, nombre:varchar(50),
entrenador:varchar(50), logo:varchar(50))

CP: { num_equipo }

VNN: { nombre }

JUGADOR (num_jugador:integer, nombre:varchar(50), demarcacion:
varchar(20), dorsal: integer, foto:varchar(50),
equipo:integer)

CP: { num_jugador }

CAj: { equipo } hace referencia a Equipo

VNN: { nombre, dorsal }

JUGADOR_PARTIDO (num_jugador: integer, num_partido: integer,
titular: booleano)

CP: {num_jugador, num_partido}

CAj: {num_jugador} hace referencia a Jugador

CAj: {num_partido} hace referencia a Partido

VNN: {titular}

ACCION (num_accion: integer, accion: varchar(3), cuarto: integer,
fecha: date, num_jugador: integer, num_partido: integer,
num_equipo: integer)

CP: {num_accion}

CAj: {num_jugador} hace referencia a Jugador

CAj: {num_equipo} hace referencia a Equipo

CAj: {num_partido} hace referencia a Partido

VNN: { num_partido, cuarto, accion, fecha}

También se detectan las siguientes restricciones de diseño:

- Se debe asegurar que los jugadores que participen en un partido pertenezcan al equipo que actúa como local en ese partido o con el que actúa como visitante.
- Si la acción la ha realizado un jugador, la Clave Ajena de la tabla Jugador que hace referencia a la tabla Equipo, deber ser la misma que la Clave Ajena de la tabla Acción que hace referencia a la tabla Equipo.
- Si la acción la ha realizado un equipo, se debería comprobar que éste disputa el partido como visitante o como local. La Clave Ajena al jugador será nula en la tabla Acción.
- Si la acción sólo está asociada al partido, las Claves Ajenas a Equipo y Jugador serán nulas en la tabla Acción. Esto ya está solucionado con este diseño lógico.

2.2.4 Diseño de las interfaces de usuario

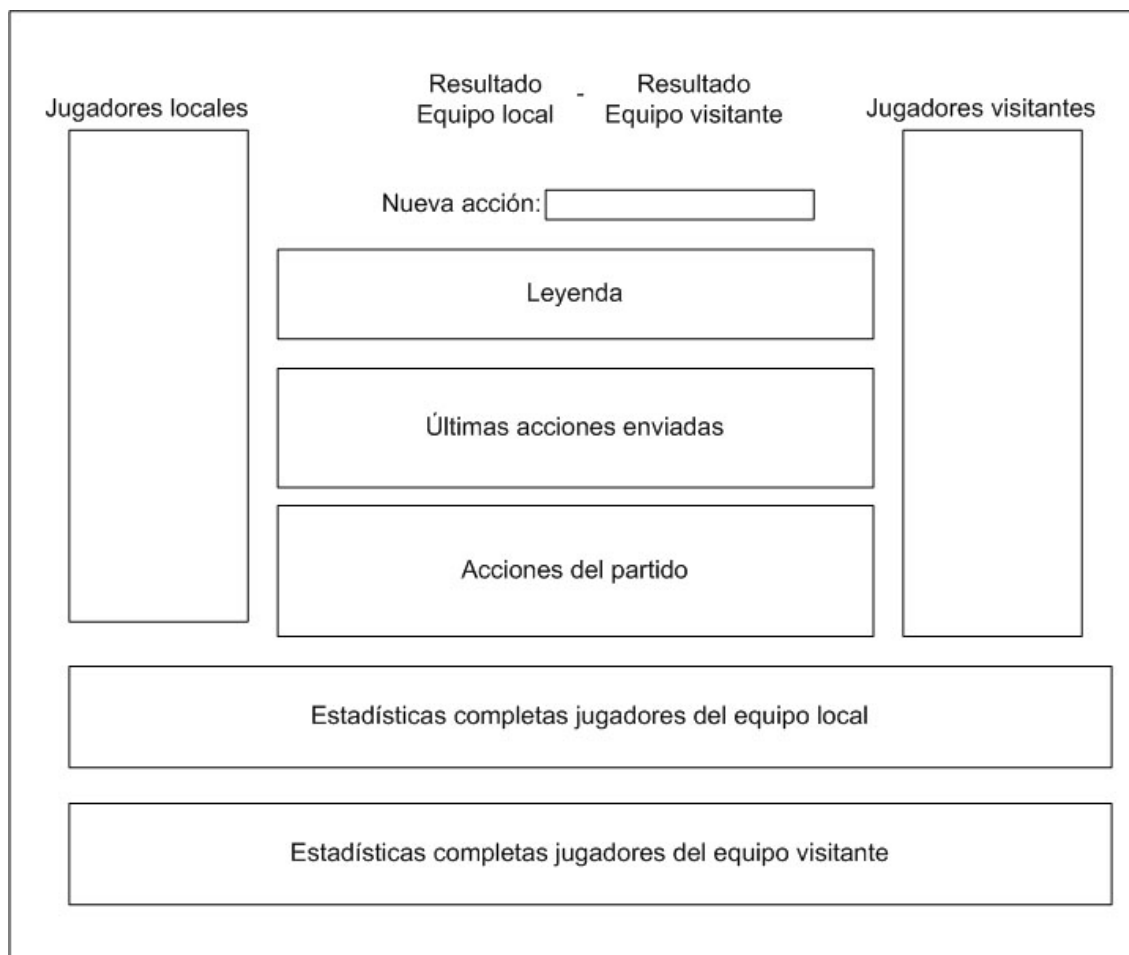


Figura 21: Boceto inicial de interfaz para la aplicación de escritorio

En una primera etapa, se decide diseñar el boceto de la interfaz de la aplicación de escritorio, e ir mejorándolo hasta conseguir un diseño final de la interfaz para esta aplicación. Por su parte, la aplicación web, se diseñará en base al diseño de la aplicación de escritorio intentando dotarla de la misma o mayor interactividad y completitud de información.

En el boceto inicial para la aplicación de escritorio (ver Figura 21:), se intenta mostrar en una misma vista toda la información, ya que se piensa que de un simple vistazo el entrenador o el estadígrafo deben obtener toda la información. Así pues, aparece el resultado del partido actual en la parte superior de la página, y debajo el campo de texto que permitirá introducir las acciones que van sucediendo a lo largo del partido.

Este campo de texto debe manejarlo un gestor de eventos de teclado para que al pulsar la tecla ENTER envíe las acciones escritas en él, sin necesidad de existir un botón que pueda causar confusión y se intente pinchar con el ratón. Toda la interfaz debe estar enfocada a la máxima utilización del teclado para favorecer la rapidez a la hora de introducir los datos. Debajo del campo de texto aparecen en forma de leyenda la lista de comandos que representan todos los posibles tipos de acciones que se pueden utilizar. Más abajo ya aparece una sección en la que se mostrarán las últimas acciones que se han enviado, y por tanto, contendrá una única acción si sólo se ha enviado una o muchas si se han enviado varias acciones simultáneamente. Las acciones aparecen ya formateadas en un lenguaje fácilmente legible para cualquier persona. Por último, en esta sección central, aparece una zona en la que se enumerarán todas las acciones que se han desarrollado cronológicamente a lo largo del partido en un formato legible para cualquier persona.

En la parte izquierda, está la sección de los jugadores del equipo local, donde se separan los jugadores que están disputando el encuentro en un determinado momento de los que están en el banquillo. Para cada jugador se mostrará su dorsal y su nombre junto a los puntos que haya anotado y las faltas personales que haya realizado. De esta forma, se tiene una valoración instantánea del rendimiento de ese jugador. Para facilitar la lectura de estos datos las faltas personales se mostrarán en colores, pasando del verde cuando llevan pocas faltas (1, 2) al amarillo (3), naranja (4) y rojo (5) indicando que el jugador ha sido expulsado. Así en un vistazo rápido se sabe cuál es la situación de los jugadores, conociendo quiénes están disponibles para jugar, cuáles están en peligro al llevar varias faltas, y cuáles han sido expulsados del partido.

En la parte derecha se muestra toda esta misma información pero para los jugadores del equipo visitante.

Por último, en la zona inferior de la interfaz, se mostrarán las estadísticas completas para todos los jugadores del equipo local y del equipo visitante.

Este diseño es bastante completo y sirve de base para el diseño final, pero no contempla algunas áreas de información necesarias para describir el partido de forma completa. Además está muy sobrecargado mostrando demasiada información en un solo vistazo. Por tanto, se decide que lo mejor es utilizar pestañas para organizar la información, dejando 3 pestañas con los siguientes contenidos:

- La *primera pestaña* (ver *Figura 22:*) es muy parecida a ese boceto inicial. Los únicos cambios son que desaparecen las estadísticas completas de los jugadores tanto del equipo local como del visitante de la zona inferior de la interfaz. En su lugar, se incluye nueva información más “light” para no sobrecargar mucho la interfaz, como es la información del partido (lugar, fecha y hora donde se celebra) y la información de los tanteos de ambos equipos por cuartos, es decir, una comparación de la puntuación total que ha obtenido cada equipo en cada uno de los cuatro cuartos. También se añade una nueva funcionalidad destinada sobre todo para los entrenadores, que consiste en el aviso de Bonus. Cuando el equipo local o visitante cometa 5 faltas se activará. Se utiliza el mismo código de colores que para las faltas personales de los jugadores, pero en este caso, a partir de las 5 faltas que hayan cometido los jugadores de un equipo en un determinado cuarto se queda activado el bonus en color rojo.
- La *segunda pestaña* (ver *Figura 23:*) se corresponde con la información que se elimina de esa primera pestaña, es decir, con las estadísticas completas de los jugadores del equipo local y del equipo visitante. Es mucha información concentrada en poco espacio y por tanto esta pestaña ya contiene suficiente información.
Se sigue el mismo código de colores para indicar las faltas personales de los jugadores, y además en el campo valoración, si el jugador posee en un determinado momento una valoración negativa, se indica con números rojos, para resaltar visualmente que el jugador no está teniendo una buena actuación en el partido.

- La *tercera pestaña* (ver *Figura 24:*) contiene información que no aparecía en el boceto inicial. Se trata de las estadísticas completas de los equipos, con lo que se está proporcionando otra visión diferente de las estadísticas calculadas. Se añaden nuevas estadísticas con respecto a las de los jugadores como son los puntos y las faltas personales por cada cuarto que ha realizado el equipo. Puede ser útil esta información por ejemplo para saber en qué cuartos se ha incurrido en bonus. Como todas las estadísticas van a ser una comparativa entre los 2 equipos, para cada una de ellas se muestra en **negrita** el valor del equipo que mejor haya realizado esa estadística o se dejan ambos sin **negrita** en caso de empate. También se utiliza el código de colores para indicar las faltas de cada equipo ha cometido en cada cuarto. Por tanto, se está intentando facilitar una rápida comprensión de los datos a partir de medidas gráficas aparentemente sencillas pero útiles.

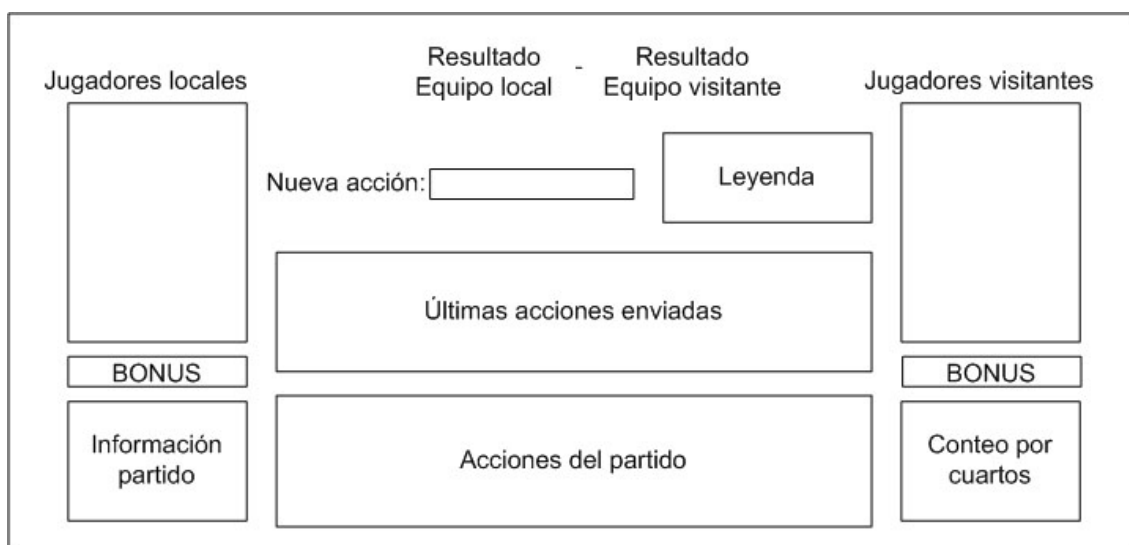


Figura 22: Diseño final de las aplicación de escritorio (Pestaña 1)

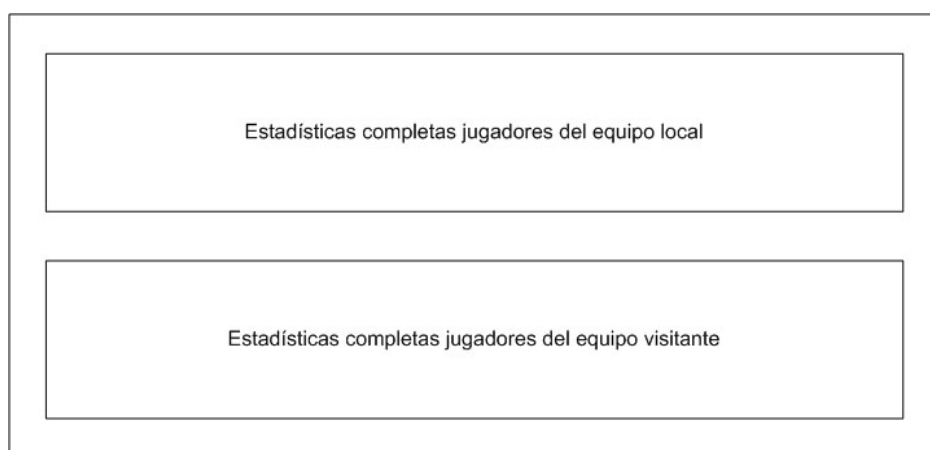


Figura 23: Diseño final de la aplicación de escritorio (Pestaña 2)

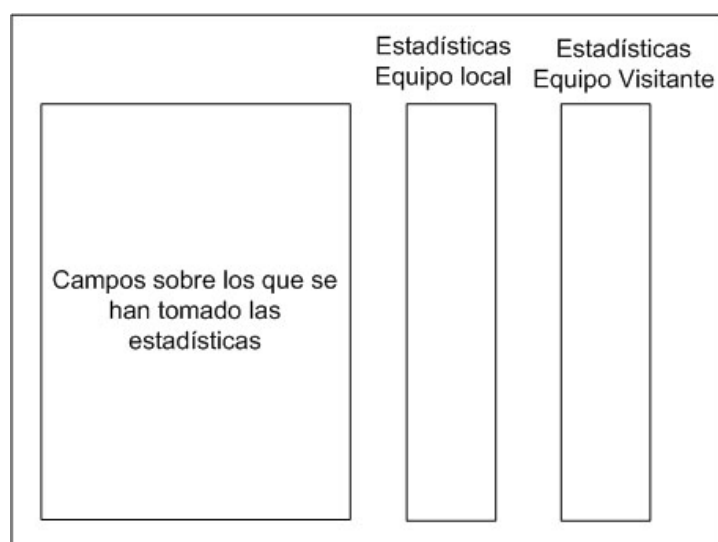


Figura 24: Diseño final de la aplicación de escritorio (Pestaña 3)

El diseño de la aplicación web es prácticamente idéntico al de la aplicación de escritorio salvo para la primera pestaña. En la aplicación web sólo es necesario mostrar las acciones que han sucedido a lo largo del partido en un formato legible que permita seguir la retransmisión del mismo. Así pues, se elimina de esta primera interfaz (ver *Figura 25:*) el campo de texto para introducir acciones, la leyenda de comandos posibles y las acciones enviadas. Todo ese espacio lo van a ocupar las acciones del partido, que se van a mostrar con un formato legible, acompañadas del minuto y segundo del cuarto en que se han producido, y dependiendo de la acción de iconos gráficos que faciliten su entendimiento. El resto de la interfaz es análogo a la aplicación de escritorio pero en este caso en formato web.

La segunda (ver *Figura 26:*) y tercera pestañas (ver *Figura 27:*) apenas varían de diseño. Únicamente se les agrega una cabecera en la parte superior en la que aparece el resultado del partido, ya que si el usuario está viendo la información de los jugadores no es fácil calcular el resultado en ese instante, y al fin y al cabo es lo que más importancia tiene. De esta forma, un usuario puede estar consultando las estadísticas detalladas de los jugadores o de los equipos, y a la vez conoce el resultado real en ese instante del partido sin necesidad de cambiar a la primera pestaña. En esta segunda pestaña que contiene las estadísticas detalladas de los jugadores, con respecto a la aplicación de escritorio se añade la foto (si existe) y la demarcación de cada jugador, ya que así es más fácil identificarlo por parte del visitante de la web.



Figura 25: Diseño final de la aplicación web (Pestaña 1)

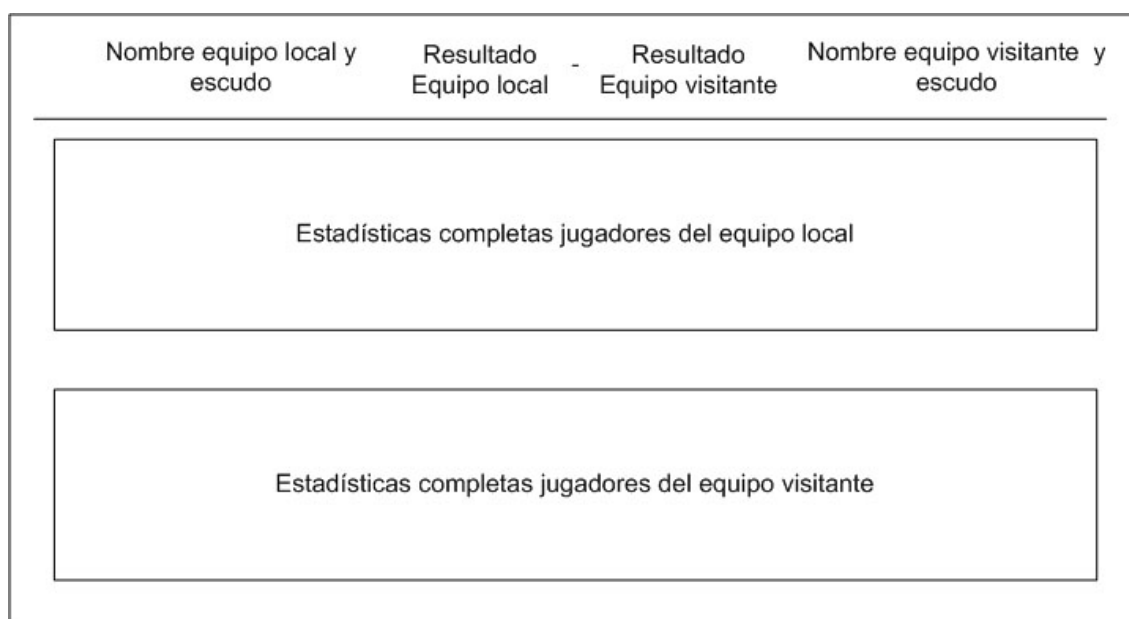


Figura 26: Diseño final de la aplicación web (Pestaña 2)

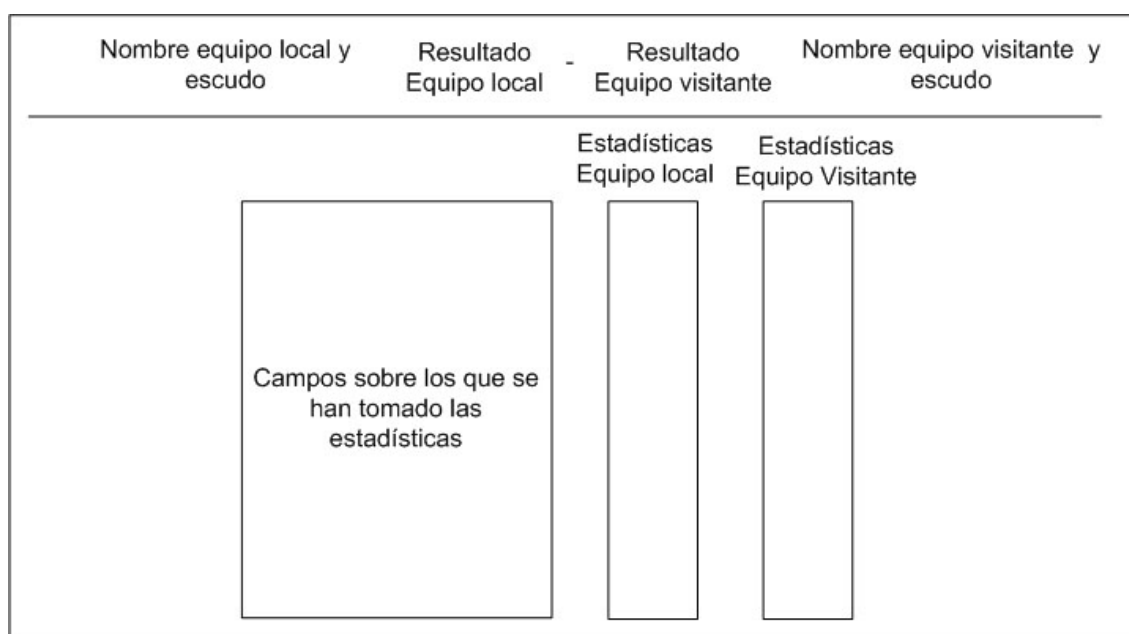


Figura 27: Diseño final de la aplicación web (Pestaña 3)

En resumen, se realiza un diseño de interfaces a conciencia, muy orientado hacia el usuario para que rápidamente pueda extraer toda información gracias al empleo de sencillos pero útiles recursos gráficos.

La aplicación de escritorio además se orienta a la rapidez y sencillez en la introducción de datos, permitiendo el manejo de la misma si se desea exclusivamente a través del teclado, que es más rápido que el ratón.

CAPÍTULO 3

Experimentación: pruebas y resultados

3. EXPERIMENTACIÓN: PRUEBAS Y RESULTADOS

El 8 de Noviembre de 2007, se prueba la aplicación en un partido real televisado correspondiente a la Euroliga de baloncesto, donde se enfrentaban los equipos Prokom Trefl y Tau Cerámica. La prueba consiste en introducir en la base de datos la información de los jugadores de ambos equipos, del partido y a lo largo del partido introducir las acciones que suceden según se ven por la televisión. Al ser una primera prueba donde todavía no se está muy familiarizado con los comandos con que se introducen las acciones, y al visualizar el partido a través de la televisión, donde no se aprecian todas acciones como en vivo, sólo se registran las acciones que realiza el Tau Cerámica y al menos las acciones correspondientes a canastas del Prokom Trefl, de forma que se pueda comparar el resultado final.

Una vez finalizado el partido, se comprueban los resultados obtenidos con la aplicación con los resultados oficiales que se encuentran en la web oficial de la Euroliga

<http://www.euroleague.net/main/results/showgame?gamenumber=3&gamecode=26&phasetypecode=rs>

Para el Tau Cerámica, las estadísticas que se obtienen con la aplicación (para aquellas que coinciden con las que aparecen en esta página ya que en esta página se obtienen unas estadísticas más completas) apenas difieren de las que aparecen en esa página, y el conteo de puntos por cuartos para los 2 equipos es el mismo así como el resultado final. Por tanto se están obteniendo resultados satisfactorios, ya que a través de la televisión es fácil anotar las canastas, pero el resto de acciones con aproximarse es suficiente debido a que estas acciones no se aprecian tan bien como en vivo, y no se ve como interactúan los 3 árbitros.

Estos resultados se exponen al Director de Proyecto en la reunión del 23 de Noviembre, y se decide que se va a realizar una prueba de la aplicación en vivo para certificar estos buenos resultados. Para ello, el 2 de Diciembre de 2007, a las 12:00 se acude al partido en el pabellón Ciutat de Castelló, en el que se enfrentaban VERDICE CASTELLON – C.D. SAGRADO CORAZON, equipos de la liga infantil femenina. Se introducen los dorsales de las jugadoras y nombre aleatorios ya que se desconocen y se registran las acciones de los 2 equipos. Los resultados no se pueden comprobar más que con la planilla que utiliza el entrenador de uno de los equipos. Por tanto sólo se comprueba el resultado del partido que es el correcto 21-45. Sin embargo, se ha comprobado que la forma de introducir los datos es cómoda una vez conocidos los comandos, de forma que apenas se pierde contacto visual con el campo.

Las pruebas de la aplicación web se realizan en modo local y en red local. Para realizar una prueba para ordenadores de una red externa a través de Internet, hubiera sido necesario abrir los puertos del router y esto no es posible en la UJI. Se ha probado en distintos ordenadores con distintos sistemas operativos instalados (Windows XP, Linux), con distintas resoluciones de pantalla (800x600, 1024x768, 1280x768) y con diferentes navegadores (Internet Explorer, Mozilla Firefox, Opera) y en todos los casos se ha obtenido un buen resultado, visualizándose la página web correctamente.

CAPÍTULO 4

Conclusiones

4. CONCLUSIONES

La realización de este proyecto ha sido la excusa perfecta para que el alumno se enfrentase a un proyecto real, en el cual se tiene un cliente que realiza su encargo y se aporta una solución a este encargo. La función principal de un ingeniero es la de realizar diseños o desarrollar soluciones tecnológicas a necesidades dadas. Para ello, se deben identificar y comprender las características más importantes del problema y a partir de ellas deducir cuáles son las mejores soluciones para afrontar las limitaciones encontradas.

En la realización de este proyecto, se ha tenido total libertad en la toma de decisiones, tanto en la forma de trabajar, de elegir las tecnologías a utilizar, etcétera, para diseñar y dar solución a las necesidades que plantea el cliente, lo que en ocasiones ha sido una ventaja pero en otras ocasiones ha supuesto un gran número de dudas que surgían y que se debían resolver.

Muchas veces el cliente no sabe realmente qué es lo que quiere y es aquí donde entra en juego el realizar un exhaustivo análisis de las soluciones existentes a problemas similares. Por tanto, es muy importante la realización del apartado de "Estado del Arte". Uno de los errores más comunes en el proceso de desarrollo de una nueva aplicación consiste en reinventar la rueda, es decir, a partir del análisis del problema que se pretende resolver, plantear una solución completamente autónoma, creada desde cero, y sin tener en cuenta posibles soluciones aportadas anteriormente al mismo problema. Siempre que se busquen y analicen soluciones reales de problemas reales, es más fácil poder ofrecer y observar qué es lo que realmente necesita un cliente en referencia a lo que pide y no tiene muy claro cómo lo quiere. Así además, se observan las deficiencias o carencias de esas soluciones y se extraen sólo las ideas que parezcan productivas. De esta forma, reutilizando ideas, se va a desarrollar un software de calidad, eficaz y eficiente, y además en un menor tiempo ya que no se parte desde cero.

En resumen, hoy en día, gran parte del trabajo de un ingeniero de software consiste más en articular las piezas de que dispone, que de reinventar nuevas piezas, sobre todo en el ámbito de las aplicaciones de gestión. El único problema fundamental, es el tiempo que se debe dedicar a descubrir las funcionalidades y la filosofía de funcionamiento de estos componentes reutilizables.

Para realizar este proyecto, se han tenido que adquirir muchos conocimientos nuevos (Java Swing, Ajax, JSP, frameworks, etc.) y utilizar, integrar y profundizar en otros que ya se tenían previamente gracias a las asignaturas cursadas en el plan de estudios (Procesadores del lenguaje, Ingeniería de Software, Sistemas cliente - servidor, etc.). Se han tenido que estudiar frameworks o componentes reutilizables de los que no se tenía ni siquiera conocimiento de su existencia, pero que han permitido aligerar el desarrollo de algunas características del proyecto. También se han tenido que aprender las principales ideas, normas y conceptos a utilizar acerca del baloncesto. Estos conocimientos ayudaron desde un principio a tener claro como iba a estar organizada la aplicación, así como toda la estructura de datos que ésta iba a manejar. Así pues, estos conocimientos también influyeron junto a la experiencia, las soluciones similares estudiadas en el "Estado del arte" y los conocimientos del alumno, en el establecimiento de la planificación general del proyecto, es decir, en la estimación del tiempo que iba a suponer aproximadamente realizar cada una de las tareas para llevar a buen término este proyecto.

Es importante destacar también la continua evaluación a la que se ha sometido al proyecto a lo largo de todo su desarrollo, efectuando diversas pruebas con las que poder comparar el resultado que se obtenía con las características inicialmente planificadas, corrigiendo así en muchos casos parte del planteamiento inicial o el resultado obtenido. En esta fase de evaluación han participado tanto el cliente probando los prototipos que se le presentaban, como distintos usuarios con diferentes grados de conocimientos informáticos y distinta afinidad al baloncesto.

En la mayoría de casos, se obtuvieron resultados satisfactorios y se recogieron un conjunto de recomendaciones que se tuvieron en cuenta para mejorar los resultados el proyecto.

Los resultados finales del proyecto indican un alto grado de consecución de los objetivos que se plantearon al inicio del proyecto. Como resultado tangible del mismo se ha obtenido una aplicación totalmente funcional que cumple completamente los requisitos que había solicitado el cliente, estando preparada para su implantación inmediata, de forma que se haga un uso real de la misma.

Con esta aplicación se obtienen prácticamente el mismo conjunto de estadísticas que obtienen las aplicaciones de los organismos oficiales de baloncesto como la ACB, NBA, Euroliga, etc. pero además, se aporta una herramienta que permite retransmitir los partidos en directo en modo online. Por otra parte, comparando la forma de introducir las acciones que suceden en un partido de esta aplicación respecto a la de los organismos oficiales, se proporciona una nueva forma de introducirlas de forma más rápida, cómoda y eficaz al utilizar el teclado en vez del ratón y al permitir introducir varias acciones simultáneamente.

Otra de las características importantes, es que la página web que se genera para retransmitir el partido es accesible y muy completa y permite rápidamente obtener un resumen de toda la información del partido a cualquier persona, desde cualquier lugar que se tenga acceso a Internet.

Como conclusión final, y a nivel particular, puedo asegurar, que la experiencia y el conocimiento adquirido con la realización de este proyecto, ha sido fundamental a la hora contactar con la problemática de la vida real, dejando de lado la vida académica.

En esta vida todo es mejorable, y aún estando muy orgulloso del trabajo realizado, siempre se pueden hacer las cosas mejor. Por eso, una vez finalizado el proyecto, se tiene la sensación de que, aún cumpliendo todos los requisitos, hay cosas que han quedado pendientes, como ampliaciones o mejoras, pero un proyecto debe acotarse en el tiempo, ya que de lo contrario se estarían agregando mejoras infinitamente. Algunas de estas posibles mejoras o ampliaciones se exponen en el siguiente apartado.

Dicen que la experiencia es un grado y ésta ha sido de lo más gratificante.

CAPITULO 5

Posibles mejoras y ampliaciones

5. POSIBLES MEJORAS Y AMPLIACIONES

Este proyecto se enmarca dentro de una idea mucho más amplia que pretendería gestionar toda la información relativa a una o varias competiciones de baloncesto, en las que se disputan encuentros entre los equipos inscritos en esas competiciones. Sería un sistema capaz de gestionar la información de todos los equipos, de sus jugadores, de los estadios donde se disputan los encuentros, y de organizar estos equipos en diferentes competiciones como ligas, copas, etc. Dentro de este sistema, para cada encuentro se establecería el estadio en que se disputa, la fecha y la hora, y para cada equipo cuáles son los jugadores convocados y cuáles forman el quinteto inicial. Desde este punto es desde donde parte este proyecto, que trata de gestionar la información de cada partido en concreto. Por tanto, la ampliación más grande y a su vez necesaria para dotar al sistema de mayor flexibilidad, y capacidad sería llevar a cabo todo este proyecto global, el cual una vez finalizado se podría pensar en comercializarlo a organismos relacionados con el baloncesto, como por ejemplo algún tipo de federación, o institución que gestione campeonatos de baloncesto.

Sin embargo, pensando más a corto plazo, la ampliación más rápida y que habría que hacer en primer lugar, es crear las interfaces que permitan introducir los equipos y jugadores para cada partido sin tener que cambiarlos directamente en la base de datos. La base de datos se ha diseñado de forma que almacena información de distintos equipos, con sus distintos jugadores y almacena la información de todos los partidos que puedan disputarse entre ellos, por lo que ya está preparada para esta primera ampliación.

Otras ampliaciones también a corto plazo que se podrían realizar serían las siguientes:

- Se podría introducir la zona desde donde han tirado los jugadores a canasta para obtener así el mapa de temperatura de las zonas desde donde más se ha tirado. Esta información podría ser útil, pero en el ámbito en que se va a implantar este proyecto no es útil, porque al estar enfocado a equipos de cantera, los jugadores tiran la gran mayoría de las veces desde la misma zona, que es debajo de la canasta.
- Mostrar información que ya está almacenada como el número de tiempos muertos que ha solicitado cada equipo y la información del tiempo que ha disputado cada jugador. Ambas se podrían visualizar para cada cuarto y para el partido completo.
- Incrementar más la precisión en la medición de los tiempos introduciendo nuevos comandos que permitan parar el tiempo y reiniciarlo de forma muy rápida e intuitiva.
- Mostrar un reloj que indique el tiempo concreto transcurrido, incluso que marque los 24 segundos de posesión, y que cada 10 minutos finalice automáticamente cada cuarto.
- Permitir a la aplicación de escritorio ejecutarse sin conexión a Internet, creando un fichero de texto formateado adecuadamente para que cuando exista conexión a Internet se importe ese fichero y se actualice la base de datos del servidor con las estadísticas que contiene del partido correspondiente.
- Adaptar el funcionamiento, la configuración y resolución del sistema para que pueda ser utilizado en dispositivos portátiles pequeños como las PDA, ya que actualmente sólo se ha comprobado su correcta ejecución en ordenadores de sobremesa y ordenadores portátiles.

CAPÍTULO 6

Referencias

6. REFERENCIAS

Se enumeran aquí las distintas fuentes consultadas que han servido de ayuda en la realización de este proyecto. Todas las páginas web que se exponen en este apartado fueron visitadas por última vez el 17 de Enero de 2008, estando activas. Se agrupan en bloques según su relación.

Información general

- [1] Información general sobre el Baloncesto
<http://es.wikipedia.org/wiki/Baloncesto>
- [2] ¿Qué es la Estadística?
<http://www.ebawords.com/tiki-index.php?page=Estad%C3%ADsticas>
- [3] MVC, patrón de diseño Modelo Vista Controlador
http://es.wikipedia.org/wiki/Modelo_Vista_Controlador
- [4] Arquitectura cliente - servidor
<http://www.uv.mx/iiesca/revista4/distribuidos.htm>
- [5] Lenguajes del lado del servidor y del lado del cliente
http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html
- [6] Comparación de los distintos SGBD
http://en.wikipedia.org/wiki/Comparison_of_databases

Estado del arte

- [7] StStats for Basketball. Software utilizado por la FEB y la FIBA
<http://www.mbt.lt>
- [8] BasketGest 3. Software similar para Windows
<http://personales.ya.com/estadisticas/>
- [9] Planilla 3000. Software similar para Windows
<http://www.bdbaloncesto.com/portal-baloncesto-stablemb45/content/w/99/2/>
- [10] Kstatistics Basket. Software similar para Palm
<http://www.kinetical.com/basquet/index.html>
- [11] TurboStats 7.0 for Basketball. Software similar para Windows y Palm
<http://www.turbostats.com/download.htm>

- [12] StatsNow. Software similar para Windows y Pocket PC.

<http://statsnow.net/statsnow/download/index.htm>

Información sobre las tecnologías utilizadas

- [13] AJAX

<http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php>

- [14] ¿Qué es un Framework?

<http://es.wikipedia.org/wiki/Framework>

- [15] Framework Prototype

<http://es.wikipedia.org/wiki/Prototype>

- [16] Framework Spry

<http://labs.adobe.com/technologies/spry/>

- [17] Framework Spry (Tabbed Panels)

http://labs.adobe.com/technologies/spry/articles/tabbed_panel/index.html

- [18] Java

http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java

- [19] Java (Swing)

http://es.wikipedia.org/wiki/Swing_%28biblioteca_gr%C3%A1fica%29

- [20] Java (¿Qué es JDBC?)

<http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap6-5.html>

- [21] Java (¿Cómo se utiliza JDBC como conector de la base de datos?)

<http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/parte21/cap21-4.html>

- [22] JSP (Java Server Pages)

<http://programacion.com/java/tutorial/jspxml/1>

Software para la instalación

[23] JDK Java (Máquina virtual de Java junto a otras utilidades)

<http://java.sun.com/javase/downloads/index.jsp>

[24] ¿Qué es XAMPP?

<http://es.wikipedia.org/wiki/XAMPP>

[25] XAMPP (MySQL + Apache + Tomcat)

<http://www.apachefriends.org/en/xampp-windows.html>

Manual de usuario

Aplicación de escritorio.....	90
Gestión acciones.....	91
Estadísticas jugadores	97
Estadísticas equipos	99
Aplicación web.....	101

ANEXO 1: MANUAL DE USUARIO

En este apartado, se pretende presentar una visión de las aplicaciones desarrolladas para que los usuarios finales conozcan sus características y su funcionamiento.

APLICACIÓN DE ESCRITORIO

Al ejecutar la aplicación de escritorio, la primera pantalla que aparece permite establecer los parámetros de conexión a la base de datos del servidor (ver *Figura 28:*). Se deben introducir correctamente el login, la contraseña y la dirección IP donde está alojado el servidor de bases de datos para que la aplicación tenga acceso a la base de datos sobre la que funciona.



Figura 28: Pantalla de conexión al servidor de bases de datos de la Aplicación de escritorio

Una vez establecida la conexión con el servidor de bases de datos, aparece la pantalla principal de la aplicación de escritorio. Esta aplicación está organizada en 3 pestañas:

- *Gestión acciones:* Permite introducir las acciones que están sucediendo en el partido y visualizar la información más relevante del mismo.
- *Estadísticas jugadores:* Muestra las estadísticas completas para todos los jugadores tanto del equipo local como del visitante en el momento en que se consultan.

- *Estadísticas equipos*: Muestra las estadísticas completas del equipo local comparándolas con las del equipo visitante para el momento en que se realiza la consulta.

Gestión acciones

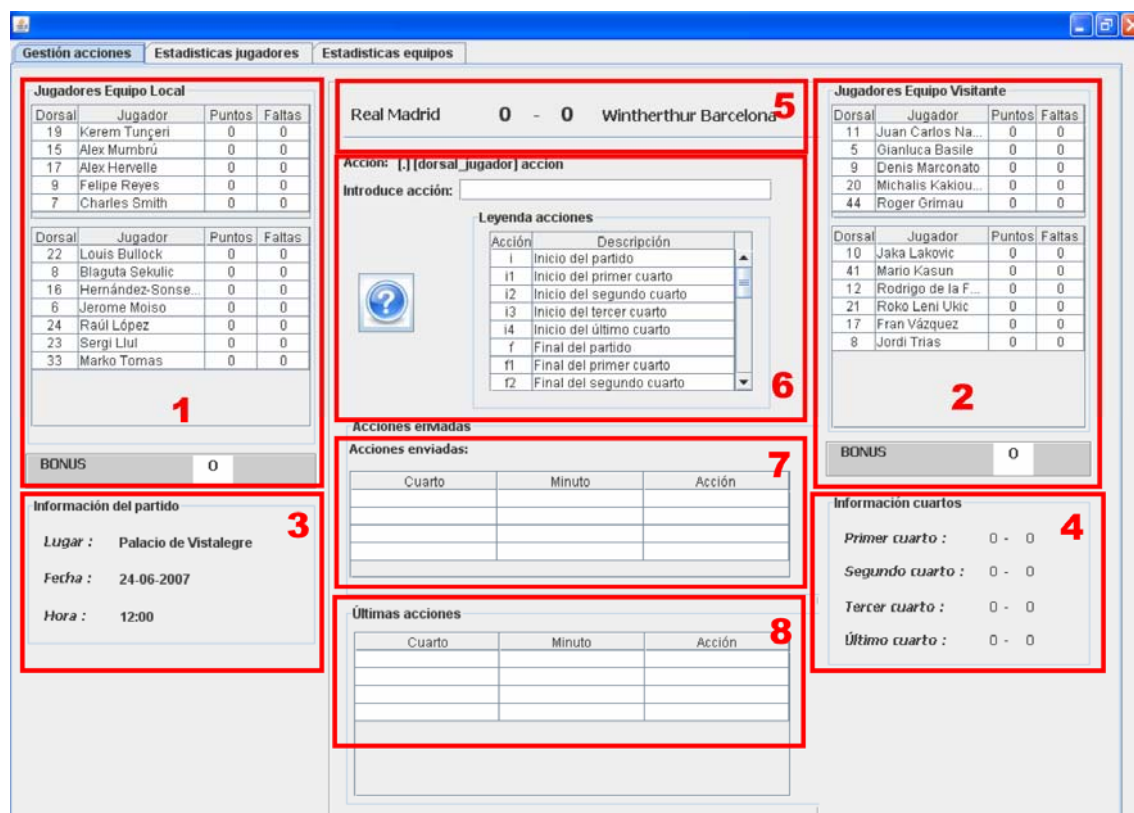


Figura 29: Identificación de las principales zonas de la 1ª pestaña de la aplicación de escritorio

En la *Figura 29*: se muestran las principales zonas en que se divide la primera pestaña de la aplicación de escritorio

- *Zona 1*: Muestra los jugadores del equipo local, separando los 5 jugadores que están disputando el encuentro en un determinado momento de los que están en el banquillo. Para cada jugador se muestra su dorsal y su nombre junto a los puntos que ha anotado y las faltas personales que ha cometido. De esta forma, se tiene una valoración instantánea del rendimiento del jugador.

Para facilitar la lectura de estos datos, las faltas personales se muestran con un fondo de distintos colores dependiendo del número de faltas que lleva el jugador en ese momento. El color verde indica pocas faltas (1, 2), el amarillo (3), el naranja (4) y el rojo (5) indicando que el jugador ha sido expulsado. Así en un vistazo rápido se sabe cuál es la situación de los jugadores, conociendo cuáles están disponibles para jugar, cuáles están en peligro al llevar varias faltas, y cuáles han sido expulsados del partido. Este mismo código de colores se utiliza en el BONUS, que también se incluye dentro de esta zona, y que hace referencia al número de faltas que lleva ese equipo en el cuarto actual. A partir de 4 faltas se muestra con fondo rojo indicando que el bonus está activo.

- *Zona 2:* Posee las mismas características que la Zona 1 pero para los jugadores del equipo visitante.
- *Zona 3:* Muestra la información general del partido, es decir, el lugar, la fecha y la hora donde se celebra.
- *Zona 4:* Muestra una comparación de la puntuación total que ha obtenido cada equipo en cada uno de los cuatro cuartos. Si se da el caso, aparece en negrita la puntuación que sea mayor. Así se puede visualizar rápidamente que equipo ha tenido un mejor resultado en ese cuarto.
- *Zona 5:* Muestra los nombres de los equipos que disputan el partido y el resultado actual del mismo.
- *Zona 6:* Contiene el campo de texto en el que se deben introducir las acciones que van sucediendo a lo largo del partido. Para introducir una nueva acción, se escribe en este campo de texto y se presiona la tecla ENTER para enviarla. Las posibles acciones que se pueden introducir aparecen debajo en la leyenda, y las reglas sintácticas que deben seguir se muestran en el apartado 2.2.1 de la presente memoria. El botón con un símbolo de interrogación aporta una pequeña ayuda que puede ayudar a recordar estas reglas sintácticas. Destacar también que se pueden introducir varias acciones encadenadas.

- **Zona 7:** Muestra en primer lugar el conjunto de comandos que se introdujo en el último envío para poder verificarlo en caso de error. Después muestra las acciones extraídas de este conjunto de comandos en un formato legible, donde cada acción tiene una descripción, y el minuto y segundo en que se ha realizado correspondiente a uno de los cuartos del partido.
- **Zona 8:** Muestra todas las acciones que se han desarrollado cronológicamente a lo largo del partido caracterizadas de la misma forma que en la Zona 7.

La *Figura 30*: muestra como la aplicación es capaz de tratar acciones enviadas mediante un conjunto de comandos en los que se usan indistintamente mayúsculas, minúsculas o una combinación de ambas. También muestra como es capaz de extraer todas las acciones que se han enviado de forma simultánea.

The screenshot displays a web-based application for managing basketball games. The interface is divided into several sections:

- Top Navigation:** Includes tabs for 'Gestión acciones', 'Estadísticas jugadores', and 'Estadísticas equipos'.
- Scoreboard:** Shows 'Real Madrid 3 - 4 Wintherthur Barcelona'.
- Player Statistics:**
 - Jugadores Equipo Local:** A table listing players like Kerem Tunçeri (3 points), Alex Mumbrú (0 points), and Charles Smith (0 points).
 - Jugadores Equipo Visitante:** A table listing players like Juan Carlos Navarro (2 points), Denis Marconato (0 points), and Roger Grimau (0 points).
- Action Management:**
 - Acción:** A field with a placeholder '[.] [dorsal_jugador] accion'.
 - Introduce acción:** A text input field.
 - Leyenda acciones:** A dropdown menu with options like 'i Inicio del partido', '11 Inicio del primer cuarto', '12 Inicio del segundo cuarto', etc.
- Action Log:**
 - Acciones enviadas:** Shows a list of actions with columns for 'Cuarto', 'Minuto', and 'Acción'. Example: '1° 00:53 Robo de balón de Roger Grimau'.
 - Últimas acciones:** A similar table showing the most recent actions.
- Game Information:**
 - BONUS:** A field showing '0'.
 - Información del partido:** Includes 'Lugar: Palacio de Vistalegre', 'Fecha: 24-06-2007', and 'Hora: 12:00'.
 - Información cuartos:** Shows the score by quarter: 'Primer cuarto: 3 - 4', 'Segundo cuarto: 0 - 0', 'Tercer cuarto: 0 - 0', and 'Último cuarto: 0 - 0'.

Figura 30: Ejemplo de envío de varias acciones simultáneas intercalando mayúsculas y minúsculas

En esta primera pestaña de la aplicación de escritorio se muestran también los mensajes de error correspondientes a entradas erróneas que haya realizado el usuario. Estas entradas erróneas pueden contener errores léxicos, sintácticos o semánticos.

Un ejemplo de detección de un error léxico se muestra en la *Figura 31*., donde el usuario introdujo "9rk", y el comando "rk" no existe.

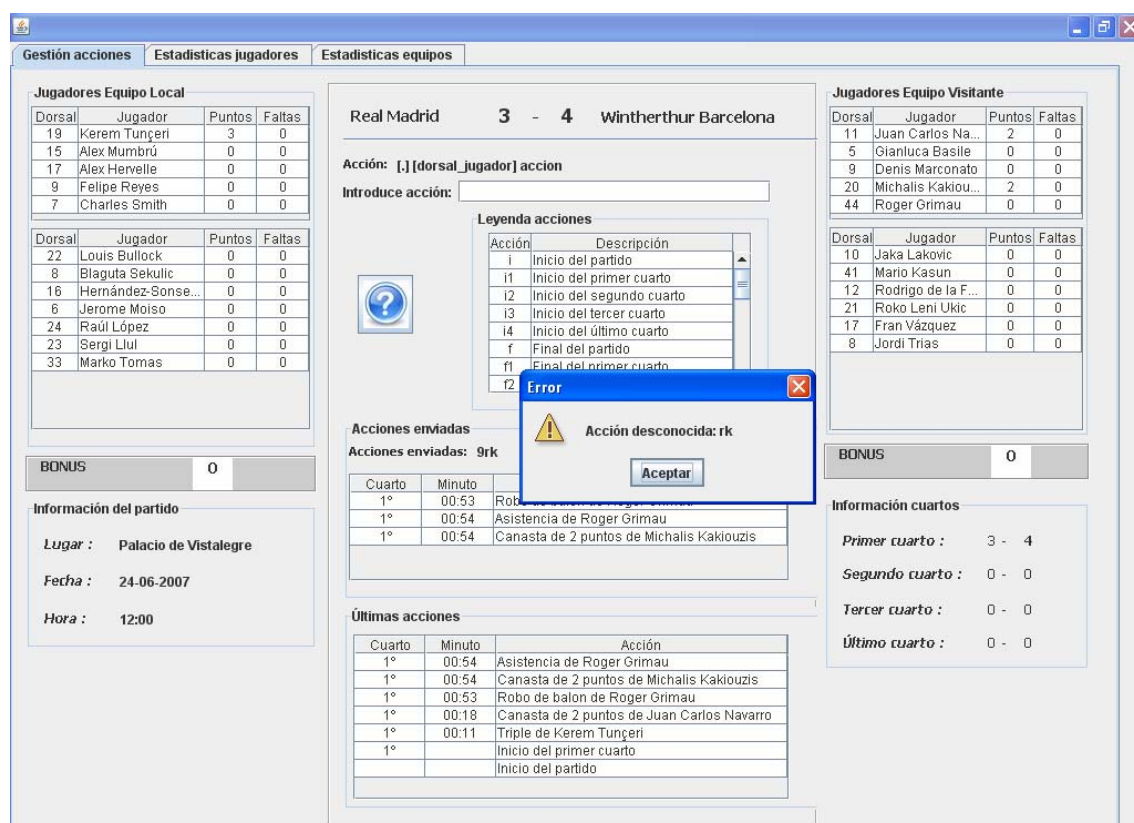


Figura 31: Ejemplo de detección de un error léxico

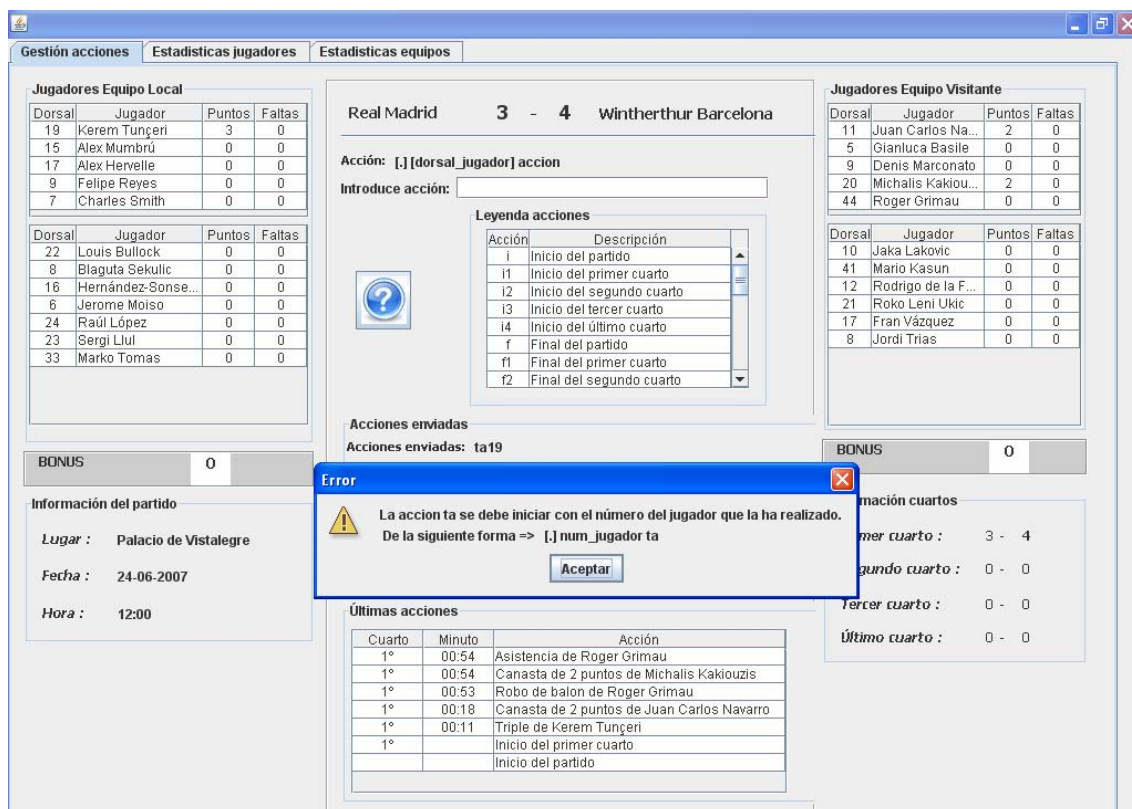


Figura 32: Ejemplo de detección de error sintáctico

Un ejemplo de detección de un error sintáctico se muestra en la *Figura 32*, donde el usuario introdujo "ta19", y la gramática del lenguaje que utiliza este proyecto indica que se debe introducir primero el dorsal del jugador y después la acción realizada.

En la *Figura 33*: se muestra un ejemplo de detección de un error semántico, donde el usuario introdujo ".11cs.44ce", indicando que el jugador Juan Carlos Navarro iba a ser sustituido por Roger Grimau. Pero como Roger Grimau ya se encuentra disputando el partido se produce un error.

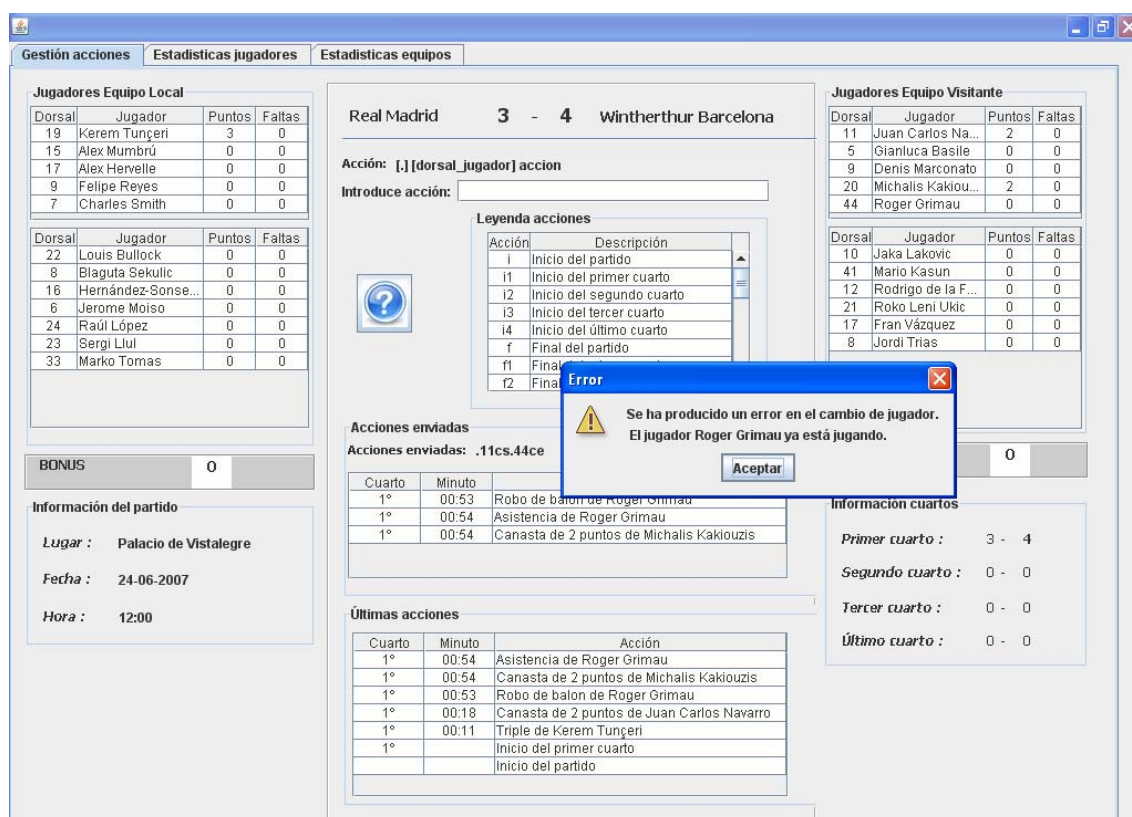


Figura 33: Ejemplo de detección de un error semántico

Se muestra por último en la *Figura 34*: un ejemplo más completo de esta primera pestaña, donde ya aparecen identificadas por diferentes colores las faltas de los jugadores y el bonus de los equipos, y donde el conteo de los cuartos muestra en negrita el resultado del equipo que ha obtenido mayor puntuación en ese cuarto.

Gestión acciones | **Estadísticas jugadores** | **Estadísticas equipos**

Jugadores Equipo Local

Dorsal	Jugador	Puntos	Faltas
22	Louis Bullock	14	2
19	Kerem Tunçeri	17	3
15	Alex Mumbrú	5	1
17	Alex Hervelle	4	2
24	Raúl López	11	1

Jugadores Equipo Visitante

Dorsal	Jugador	Puntos	Faltas
11	Juan Carlos Na...	19	3
5	Gianluca Basile	15	2
9	Denis Marconato	6	1
20	Michalis Kakiou...	3	0
44	Roger Grimau	3	0

Real Madrid 86 - 74 Wintherthur Barcelona

Acción: [.] [dorsal_jugador] accion

Introduce acción:

Leyenda acciones

Acción	Descripción
i	Inicio del partido
i1	Inicio del primer cuarto
i2	Inicio del segundo cuarto
i3	Inicio del tercer cuarto
i4	Inicio del último cuarto
f	Final del partido
f1	Final del primer cuarto
f2	Final del segundo cuarto

Acciones enviadas

Acciones enviadas: .44rd

Cuarto	Minuto	Acción
4º	07:39	Rebote en defensa de Roger Grimau

Últimas acciones

Cuarto	Minuto	Acción
4º	07:39	Rebote en defensa de Roger Grimau
4º	07:29	Tiro de 2 puntos fallado por Raúl López
4º	07:17	Robo de balón de Raúl López
4º	07:13	Perdida de balón de Roger Grimau
4º	06:54	Asistencia de Raúl López
4º	06:54	Canasta de 2 puntos de Alex Hervelle
4º	06:25	Tiro libre anotado por Roger Grimau
4º	06:22	Canasta de 2 puntos de Roger Grimau

BONUS **3**

Información del partido

Lugar : Palau Blaugrana

Fecha : 24-06-2007

Hora : 12:00

BONUS **6**

Información cuartos

Primer cuarto : 17 - 19

Segundo cuarto : 19 - 22

Tercer cuarto : 31 - 19

Último cuarto : 19 - 14

Figura 34: Ejemplo del estado de la 1ª pestaña de la aplicación de escritorio en el último cuarto de un partido

Estadísticas jugadores

La segunda pestaña de la aplicación de escritorio (ver *Figura 35:*) muestra las estadísticas completas de los jugadores del equipo local y del equipo visitante. Se sigue el mismo código de colores para indicar las faltas personales de los jugadores, y además en el campo valoración, si el jugador posee en un determinado momento una valoración negativa se indica con números rojos, para resaltar rápidamente que ese jugador no está teniendo una buena actuación en el partido. Para cada jugador se muestra:

- Dorsal que lo identifica.
- Nombre.
- Valoración obtenida a partir del resto de estadísticas, que permite conocer rápidamente el rendimiento del jugador hasta ese instante en el partido.
- Número de puntos que ha anotado.
- C3: Número de triples anotados.
- T3: Número de triples errados.

- C2: Número de canastas de 2 puntos convertidas.
- T2: Número de canastas de 2 puntos erradas.
- C1: Número de tiros libres anotados.
- T1: Número de tiros libres errados.
- RD: Número de rebotes defensivos capturados.
- RA: Número de rebotes ofensivos capturados.
- TA: Número de tapones realizados.
- A: Número de asistencias proporcionadas.
- BP: Número de balones perdidos.
- BR: Número de balones recuperados.
- FP: Número de faltas personales cometidas.
- FR: Número de faltas recibidas.

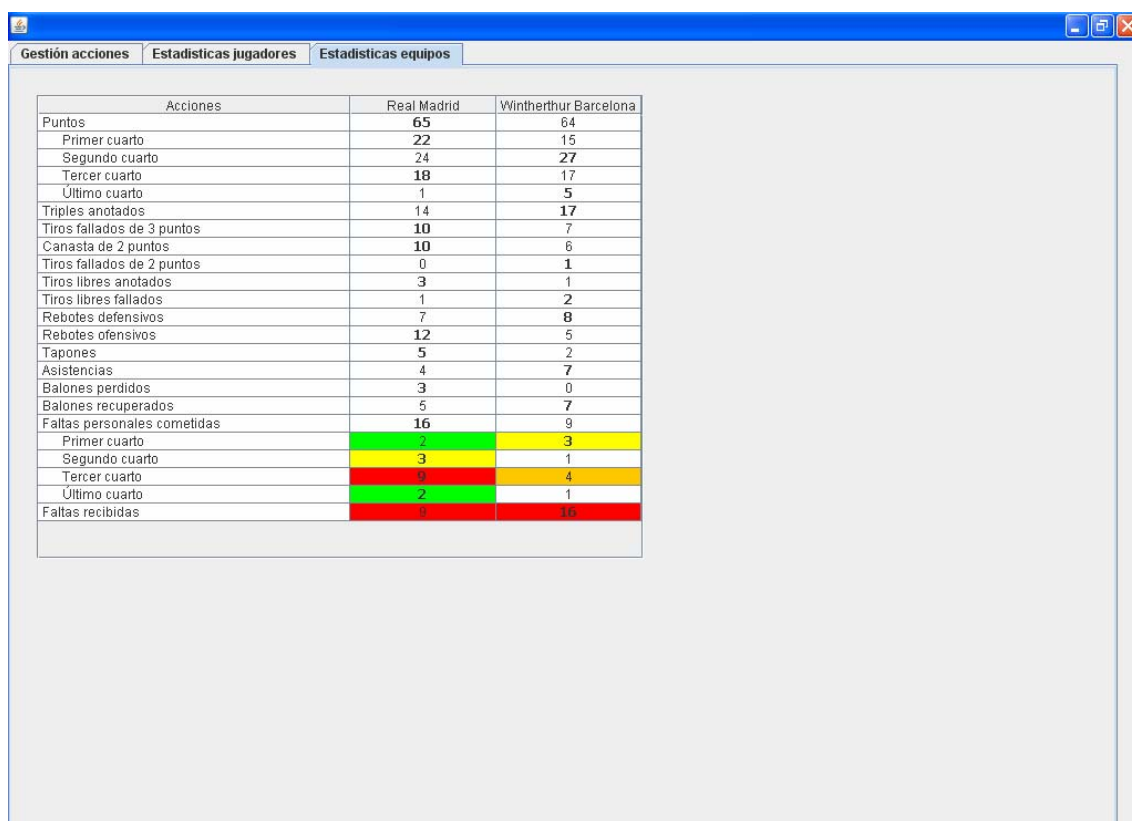
Gestión acciones Estadísticas jugadores Estadísticas equipos																	
Estadísticas jugadores equipo local																	
Real Madrid																	
Dorsal	Nombre	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
22	Louis Bullock	12	12	2	2	1	3	4	0	3	0	1	3	2	5	0	2
19	Kerem Tunçeri	3	6	1	1	1	1	1	1	4	2	0	0	4	2	2	1
15	Alex Mumbrú	-5	3	0	2	1	2	1	2	0	0	2	1	1	3	1	0
17	Alex Hervelle	8	9	2	0	1	0	1	0	2	1	1	0	2	0	3	0
9	Felipe Reyes	14	14	0	1	7	4	0	3	5	4	4	2	3	2	0	0
7	Charles Smith	-21	0	0	3	0	3	0	2	1	1	0	0	1	0	5	0
8	Blaguta Sekulic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	Hdez-Sonseca	-1	9	0	1	1	3	7	2	0	2	1	1	2	3	4	0
6	Jerome Moiso	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	Raúl López	-1	0	0	0	0	4	0	3	3	1	0	5	2	5	2	0
23	Sergi Llul	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	Marko Tomas	-10	8	2	4	1	5	0	2	3	4	2	0	4	1	0	0
Estadísticas jugadores equipo visitante																	
Wintherthur Barcelona																	
Dorsal	Nombre	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
10	Jaka Lakovic	4	12	2	1	3	3	0	2	1	3	1	0	1	2	3	0
11	JC Navarro	9	15	2	2	4	2	1	3	2	0	0	4	3	4	1	1
5	O.Basile	18	13	2	0	3	0	1	0	1	4	2	2	3	1	2	0
9	D. Marconato	-3	11	1	4	1	1	6	2	3	0	0	0	2	0	0	1
41	Mario Kasun	-4	0	0	1	0	4	0	1	2	2	0	1	0	3	0	0
12	R. de la Fuente	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	M. Kakiouzis	2	5	1	0	0	0	2	0	1	2	2	0	4	0	4	0
21	Roko L. Ukic	2	5	1	1	0	1	2	0	0	1	1	2	1	2	5	2
17	Fran Vázquez	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44	Roger Grimau	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	Jordi Trias	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 35: Ejemplo del estado de la 2ª pestaña de la aplicación de escritorio en el último cuarto de un partido

Estadísticas equipos

La tercera pestaña (ver *Figura 36:*) muestra las estadísticas completas de los equipos, con lo que se está proporcionando otra visión diferente de las estadísticas calculadas. Se añaden nuevas estadísticas con respecto a las que se muestran para los jugadores en la segunda pestaña, como son los puntos y las faltas personales por cada cuarto que ha realizado el equipo. Esta información puede ser útil por ejemplo para saber en qué cuartos se ha incurrido en bonus.

Como todas las estadísticas van a ser una comparativa entre los 2 equipos, para cada una de ellas se muestra en negrita el valor del equipo que mejor haya realizado esa estadística o se dejan ambos valores sin negrita en caso de empate. También se utiliza el código de colores para indicar las faltas que ha cometido cada equipo en cada cuarto. Por tanto, se está intentando facilitar una rápida comprensión de los datos a partir de medidas gráficas aparentemente sencillas pero útiles.



Acciones	Real Madrid	Wintherthur Barcelona
Puntos	65	64
Primer cuarto	22	15
Segundo cuarto	24	27
Tercer cuarto	18	17
Ultimo cuarto	1	5
Triples anotados	14	17
Tiros fallados de 3 puntos	10	7
Canasta de 2 puntos	10	6
Tiros fallados de 2 puntos	0	1
Tiros libres anotados	3	1
Tiros libres fallados	1	2
Rebotes defensivos	7	8
Rebotes ofensivos	12	5
Tapones	5	2
Asistencias	4	7
Balones perdidos	3	0
Balones recuperados	5	7
Faltas personales cometidas	16	9
Primer cuarto	2	3
Segundo cuarto	3	1
Tercer cuarto	9	4
Ultimo cuarto	2	1
Faltas recibidas	9	16

Figura 36: Ejemplo del estado de la 3ª pestaña de la aplicación de escritorio en el último cuarto de un partido

Para finalizar el manual de usuario de la aplicación de escritorio, la *Figura 37*: presenta el estado de la primera pestaña cuando se introduce el comando de finalización de un partido. Cuando ésto sucede, aparece una pequeña ventana a modo de resumen del partido, donde se indica el resultado final del mismo para cada equipo, indicando en negrita el resultado vencedor. También se muestra la fotografía, el nombre y las estadísticas más importantes (número de puntos, de rebotes y de asistencias) obtenidos por el MVP o mejor jugador del partido. Este jugador es aquel que haya obtenido la mayor valoración entre todos sus compañeros de equipo y también respecto de los del equipo contrario en el momento de la finalización del partido.

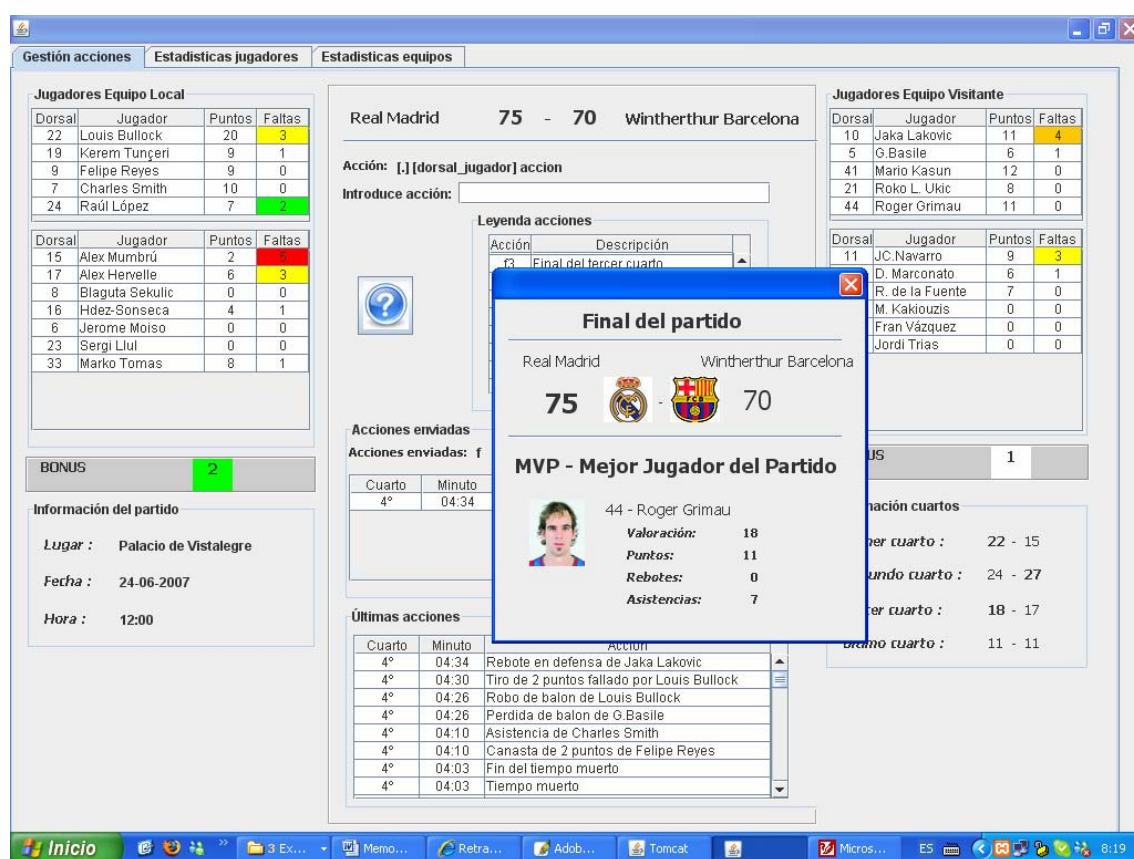


Figura 37: Ejemplo del estado de la 1ª pestaña de la aplicación de escritorio al finalizar un partido

APLICACIÓN WEB

El diseño de la aplicación web es prácticamente idéntico al de la aplicación de escritorio salvo para la primera pestaña. Esta primera pestaña (ver *Figura 38:*) elimina las Zonas 6 y 7 de la aplicación de escritorio, y su espacio lo ocupa la Zona 8 extendida. Otro cambio significativo es que se mantiene la zona 5 como cabecera para las tres pestañas. Por tanto, en la primera pestaña de la aplicación web se elimina el campo de texto para introducir acciones, la leyenda de comandos posibles y las acciones enviadas. Todo ese espacio lo ocupan las acciones del partido, que se van a mostrar con un formato legible, acompañadas del minuto y segundo del cuarto en que se han producido, y dependiendo de la acción, de iconos gráficos que faciliten su entendimiento. El resto de zonas aparecen de la misma forma que en la aplicación de escritorio, pero en este caso, en formato web.

The screenshot displays a web browser window titled 'Retransmisión partido Real Madrid - Wintherthur Barcelona'. The URL is 'http://localhost:8080/index.jsp'. The page features a blue header with the Real Madrid and Wintherthur Barcelona logos and the score 65-64. Below the header, there are tabs for 'Retransmisión partido', 'Estadísticas jugadores', and 'Estadísticas equipos'. The main content area is divided into several sections:

- Left Panel:** A table of player statistics for Real Madrid, including columns for Dorsal, Nombre, Faltas, and Puntos. It also shows a 'BONUS' section and 'INFORMACIÓN PARTIDO' (Pabellón, Fecha, Hora).
- Center Panel:** A large table titled 'RETRANSMISIÓN' showing game actions. The table has columns for Cuarto, Tiempo, Acción, and Descripción. It lists various events such as 'Tiempo muerto', 'Falta personal de Raúl López', 'Triple de Roko L. Ukic', and 'Canasta de 2 puntos de Roko L. Ukic'.
- Right Panel:** A table of player statistics for Wintherthur Barcelona, similar to the Real Madrid section. It also includes a 'BONUS' section and a 'CUARTOS' section showing the score in each quarter.

Figura 38: Ejemplo del estado de la 1ª pestaña de la aplicación web en el último cuarto de un partido

El conjunto de iconos gráficos utilizados es el siguiente:


	Inicio del partido.
	Comienzo y final de cada cuarto.
	Falta personal cometida por un jugador, indica qué número de falta ha cometido a la vez que mediante un código de colores se muestra la importancia de esa falta (azul - verde no es importante, amarillo tiene una importancia intermedia, naranja es preocupante, y rojo indica que el jugador ha sido expulsado del partido).
	Cambio de jugador
	Inicio y final de un tiempo muerto respectivamente.
	Rebote defensivo o rebote ofensivo de un jugador.
	Tapón realizado por un jugador a otro.
	Tiro libre anotado por algún jugador.
	Canasta de 2 puntos anotada por algún jugador.
	Triple anotado por algún jugador.

Tabla 3: Conjunto de Iconos utilizados en la aplicación web


La segunda (ver *Figura 39:*) y tercera pestañas (ver *Figura 40:*) de la aplicación web apenas varían su diseño respecto a la aplicación de escritorio. Únicamente se les agrega la zona 5 como cabecera en la parte superior donde aparece el resultado del partido. Así, un usuario que esté consultando las estadísticas detalladas de los jugadores o de los equipos, podrá conocer a la vez el resultado real en ese instante del partido sin necesidad de cambiar a la primera pestaña. En la segunda pestaña que contiene las estadísticas detalladas de los jugadores, con respecto a la aplicación de escritorio se añade la foto (si existe) y la demarcación de cada jugador, ya que así es más fácil identificarlo por parte del visitante de la web.

Retransmisión partido Real Madrid - Wintherthur Barcelona - Windows Internet Explorer

http://localhost:8084/BasketWeb2/index.jsp


Retransmisión partido Real Madrid - Wintherthur Bar...

Página Herramientas



73

61




Real Madrid

Wintherthur Barcelona












Retransmisión partido


Estadísticas jugadores

Estadísticas equipos



Estadísticas de los jugadores del Real Madrid

Foto	Dorsal	Nombre	Demarcación	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
	22	Louis Bullock	Escolta	12	12	2	2	1	3	4	0	3	0	1	3	2	5	0	2
	19	Kerem Tunçeri	Base	3	6	1	1	1	1	1	1	4	2	0	0	4	2	2	1
	15	Alec Mumbru	Alero	-5	3	0	2	1	2	1	2	0	0	2	1	1	3	1	0
	17	Alec Hervelle	Ala-Pivot	8	9	2	0	1	0	1	0	2	1	1	0	2	0	3	0
	9	Felipe Reyes	Pivot	14	14	0	1	7	4	0	3	5	4	4	2	3	2	0	0
	7	Charles Smith	Escolta	-21	0	0	3	0	3	0	2	1	1	0	0	1	0	3	0
	8	Blaguta Sekulic	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	Hdez-Sonseca	Pivot	-1	9	0	1	1	3	7	2	0	2	1	1	2	3	4	0
	6	Jerome Moiso	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	24	Raul Lopez	Base	11	12	2	0	2	4	2	3	3	1	0	5	2	5	3	0
	23	Sergi Llul	Base	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	33	Marko Tomas	Escolta	-10	8	2	4	1	5	0	2	3	4	2	0	4	1	0	0



Estadísticas de los jugadores del Wintherthur Barcelona










Foto	Dorsal	Nombre	Demarcación	Valoración	Puntos	C3	T3	C2	T2	C1	T1	RD	RA	TA	A	BP	BR	FP	FR
	10	Jaka Lakovic	Base	4	12	2	1	3	3	0	2	1	3	1	0	1	2	3	0
	11	JC Navarro	Base	9	15	2	2	4	2	1	3	2	0	0	4	3	4	1	1
	5	O Basile	Escolta	18	13	2	0	3	0	1	0	1	4	2	2	3	1	3	0
	9	D. Marconato	Pivot	-3	11	1	4	1	1	6	2	3	0	0	0	2	0	0	1
	41	Mario Kasun	Pivot	-4	0	0	1	0	4	0	1	2	2	0	1	0	3	0	0
	12	R. de la Fuente	Escolta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	M. Kakiozis	Ala-Pivot	2	5	1	0	0	0	2	0	1	2	2	0	4	0	4	0
	21	Roko L. Ukic	Base	2	5	1	1	0	1	2	0	0	1	1	2	1	2	5	2
	17	Fran Vázquez	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	44	Roger Crisau	Escolta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	Jordi Trias	Pivot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 39: Ejemplo del estado de la 2ª pestaña de la aplicación web en el último cuarto de un partido

Retransmisión partido Real Madrid - Wintherthur Barcelona - Windows Internet Explorer

http://localhost:8080/index.jsp

Retransmisión partido Real Madrid - Wintherthur Bar...

Real Madrid 65 64 Wintherthur Barcelona

Retransmisión partido Estadísticas jugadores Estadísticas equipos

Estadísticas de los equipos


Acción	Real Madrid	Wintherthur Barcelona
		
Puntos	65	64
Primer cuarto	22	15
Segundo cuarto	24	27
Tercer cuarto	18	17
Último cuarto	1	5
Triples anotados	14	17
Tiros fallados de 3 puntos	10	7
Canasta de 2 puntos	10	6
Tiros fallados de 2 puntos	0	1
Tiros libres anotados	3	1
Tiros libres fallados	1	2
Rebotes defensivos	7	8
Rebotes ofensivos	12	5
Tapones	5	2
Asistencias	4	7
Balones perdidos	3	0
Balones recuperados	5	7
Faltas personales cometidas	16	9
Primer cuarto	2	3
Segundo cuarto	3	1
Tercer cuarto	9	4
Último cuarto	2	1
Faltas recibidas	9	16

Figura 40: Ejemplo del estado de la 3ª pestaña de la aplicación web en el último cuarto de un partido

Procedimiento de instalación

Java JDK	107
XAMPP Básico	109
ADD-Ons Tomcat para XAMPP	118

ANEXO 2: PROCEDIMIENTO DE INSTALACIÓN

Dependiendo del rol del usuario del proyecto se deberá realizar un proceso de instalación diferente.

- Para el usuario con rol de **estadígrafo**, únicamente será necesario que se instale el JDK de Java y se copie la carpeta denominada "Aplicación de escritorio" a su equipo. Dentro de esta carpeta encontrará el archivo "*Basket2.jar*", que es el ejecutable de la aplicación de escritorio. Es importante copiar toda la carpeta, porque en el subdirectorio *lib* está el conector JDBC imprescindible para la conexión con el SGBD MySQL.
- Para el usuario con rol de **visitante** o cliente web, no será necesario que realice ninguna instalación. A través de cualquier navegador web que tenga instalado puede acceder a la aplicación web.

Sin embargo, para ambos casos es necesario utilizar un servidor, para el primer caso como servidor de bases de datos y para el segundo caso como servidor de bases de datos y servidor web. Por tanto, aparte de esa instalación específica en el ordenador local de cada usuario, se debe realizar una completa instalación de un equipo que actúe como servidor. En dicho servidor, será necesario instalar en primer lugar el JDK, ya que lo solicita el servidor de Java Tomcat. Después se instalará el servidor XAMPP que integra el gestor de bases de datos MySQL y el servidor Apache. Por último, será necesario instalar una extensión de XAMPP consistente en el servidor Tomcat, que será el encargado de dar soporte a la aplicación web de este proyecto realizada en lenguaje JSP. Una vez instalado todo el software necesario, para configurarlo será necesario acceder al gestor de bases de datos MySQL e importar la base de datos que soporta tanto la aplicación web como la aplicación de escritorio de este proyecto. También será necesario copiar el contenido de la carpeta "Aplicación web" al directorio del servidor Tomcat *ruta_tomcat/webapps/ROOT* para que ponga en funcionamiento la aplicación web.

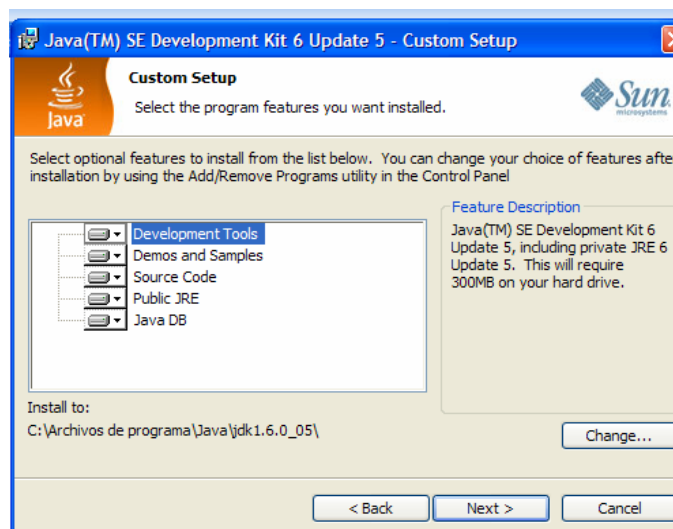
Java JDK [23]

JDK ("Java Development Kit") agrupa las diversas funcionalidades necesarias para desarrollar programas Java. Hoy en día, existen JDK's para diversos ambientes y plataformas, los cuales ofrecen lo siguiente:

- Un compilador Java, capaz de generar Byte-Code.
- Un JVM ("Java Virtual Machine"), capaz de ejecutar Byte-Code.
- Un conjunto de Clases base utilizadas para generar programas Java.
- Otras utilidades para administrar código escrito en Java.

En este proyecto se proporciona el JDK 6 Update 5 de Sun Microsystems bajo el nombre "*jdk-6u5-ea-bin-b04-windows-i586-p-27_sep_2007.exe*".

El proceso de instalación del JDK, comienza con la pantalla de bienvenida (ver *Figura 41:*). Se presiona Next y aparece el acuerdo de licencia (ver *Figura 42:*), que una vez leído se debe declinar o aceptar para continuar con la instalación. Se acepta y aparecen las características del paquete que se desea instalar (ver *Figura 43:*). Realmente para este proyecto sólo es necesaria la instalación de las 2 últimas (*Public JRE* y *Java DB*) pero para facilitar la operación es mejor instalar todas presionando Next. Por último aparecen 2 nuevas características (ver *Figura 44:*) que es necesario instalar, *Java™ SE Runtime Environment* y *Default Java for Browsers*. Se presiona Next y una vez instaladas ha finalizado el proceso de instalación (ver *Figura 45:*). Ahora el sistema ya es capaz de ejecutar aplicaciones Java de escritorio o vía web.

**Figura 41: Instalación JDK Java (Paso 1)****Figura 42: Instalación JDK Java (Paso 2)****Figura 43: Instalación JDK Java (Paso 3)**

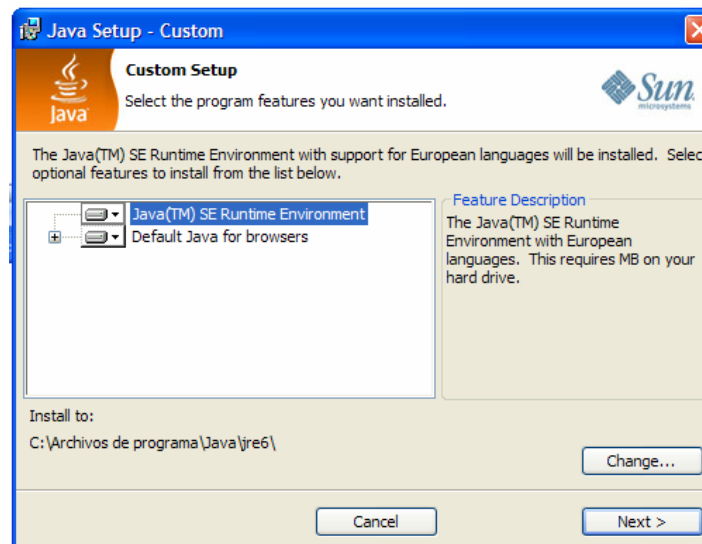


Figura 44: Instalación JDK Java (Paso 4)



Figura 45: Instalación JDK Java (Paso 5)

XAMPP Básico [25]

XAMPP [24] es un servidor independiente de plataforma, software libre, que consiste principalmente en el gestor de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para distintas plataformas como Microsoft Windows, Linux, Solaris, y MacOS X.

En este proyecto se proporciona la versión 1.6.5 básica de XAMPP bajo el nombre "*xampp-win32-1.6.5-installer.exe*".

El proceso de instalación de XAMPP comienza con la selección del idioma en que se realizará la instalación. Se selecciona el idioma deseado (ver *Figura 46:*) y tras presionar OK aparece una nueva pantalla en que se da la bienvenida al proceso de instalación (ver *Figura 47:*). Se presiona Next y la siguiente etapa consiste en indicar la ruta de destino para la instalación. Si se está de acuerdo con la ruta por defecto (C:\Xampp) se presiona OK, y si no, pinchando en el botón Browse se puede indicar una ruta particular (ver *Figura 48:*). Cuando esté establecida la ruta se presiona Next y aparece una nueva pantalla en la que se debe decidir si se quiere crear un acceso directo a XAMPP en el escritorio, una carpeta dedicada a XAMPP en el menú inicio, y si se instalan como servicios MySQL, Apache y Filezilla. Es necesario instalar al menos los 2 primeros (ver *Figura 44:*). Una vez seleccionados, se presiona Install y se instalará el servidor XAMPP con los parámetros que se le han indicado.

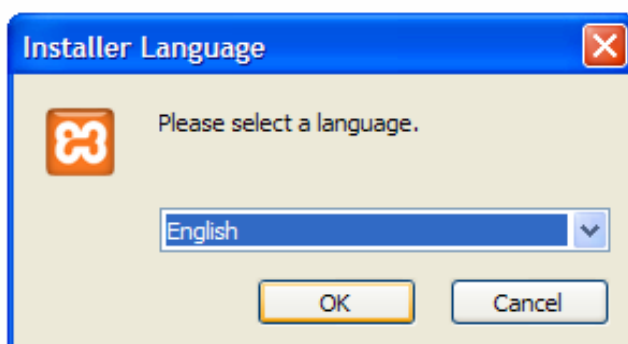


Figura 46: Proceso de instalación de XAMPP básico (Paso 1)

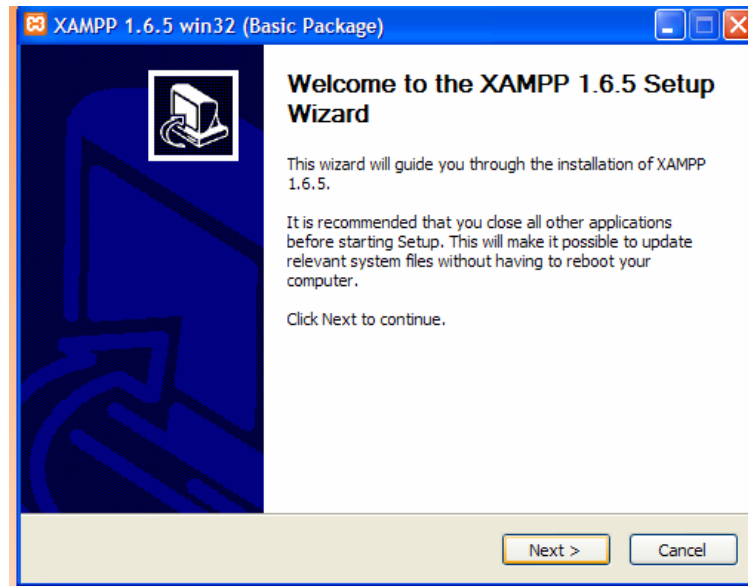


Figura 47: Proceso de instalación de XAMPP básico (Paso 2)

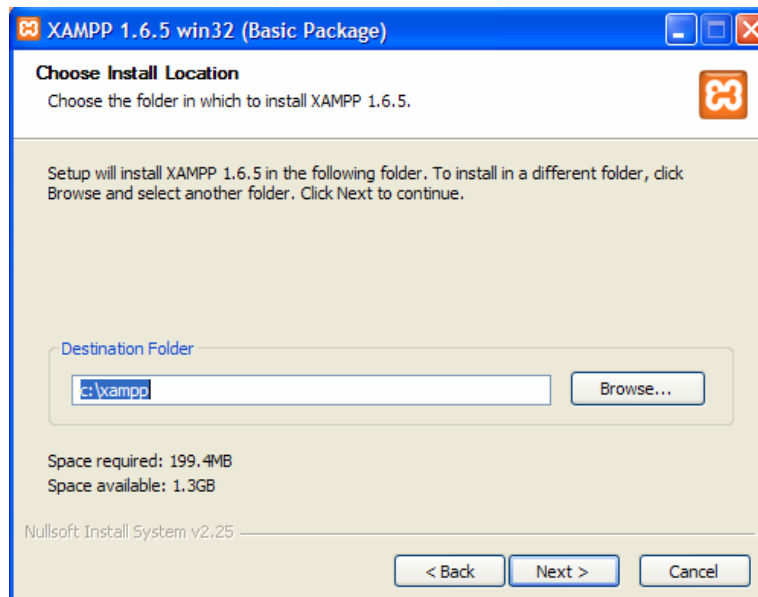


Figura 48: Proceso de instalación de XAMPP básico (Paso 3)

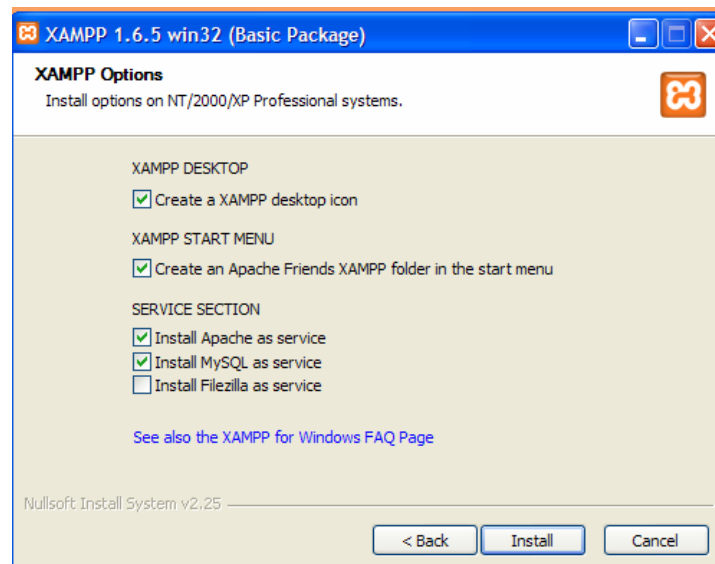


Figura 49: Proceso de instalación de XAMPP básico (Paso 4)



Figura 50: Proceso de instalación de XAMPP básico (Paso 5)

Una vez instalado XAMPP (ver *Figura 50:*), se ejecuta su Panel de Control (ver *Figura 51:*). En él se muestra de forma resumida el estado de los 2 servicios que se han instalado (Apache y MySQL). Si están en ejecución, aparece un cuadrado verde en el que se puede leer "Running". Si se pincha en el botón Admin correspondiente al servicio Apache, se abre el navegador predeterminado con la página web para configurar Apache y MySQL. En esta página se debe seleccionar primero el idioma de la configuración (ver *Figura 52:*).

Se selecciona el idioma deseado y aparece una nueva página de bienvenida a la configuración de XAMPP (ver *Figura 53:*). Toda la configuración que se necesita del servidor, de momento, se centra en el gestor de bases de datos MySQL, por lo que se accede directamente al enlace "phpMyAdmin" de la barra naranja del margen izquierdo. La configuración consiste en importar la base de datos creada para el proyecto, y en modificar la contraseña del usuario *root* para acceder al gestor. Para el primer caso, se pincha en el enlace "Importar" (ver *Figura 54:*), que lleva a una nueva página (ver *Figura 55:*) donde se debe indicar el archivo a importar, aunque sería conveniente modificar también el juego de caracteres a *latin1* para no tener problemas con las tildes y otros caracteres. Una vez importada la base de datos, se comprueba que el proceso ha tenido éxito seleccionando en la barra izquierda la base de datos que tenga como nombre *Basket*. Así, se muestra una nueva página que contiene el resumen de las tablas de la base de datos (ver *Figura 56:*).

Una vez importada la base de datos, sólo queda cambiar la contraseña del usuario *root*. Para ello, se pincha el icono de la casa (Inicio) en la barra izquierda de la página, y se vuelve así a la página de partida (ver *Figura 54:*). Ahora se selecciona el enlace "*Privilegios*", que muestra una página que contiene los distintos usuarios del gestor de bases de datos con sus respectivos permisos (ver *Figura 57:*). Para el usuario *root*, que tiene todos los privilegios (ALL PRIVILEGES) se pincha en el icono del final de la fila, y aparece una nueva página donde se permite modificar su contraseña entre otras acciones (ver *Figura 58:*). Se escribe la nueva contraseña (*basket*) las dos veces que se solicita, y se presiona continuar para que registre la modificación.

Con estos pasos se tiene configurada la base de datos del proyecto. Ahora falta instalar y configurar el servidor Tomcat.

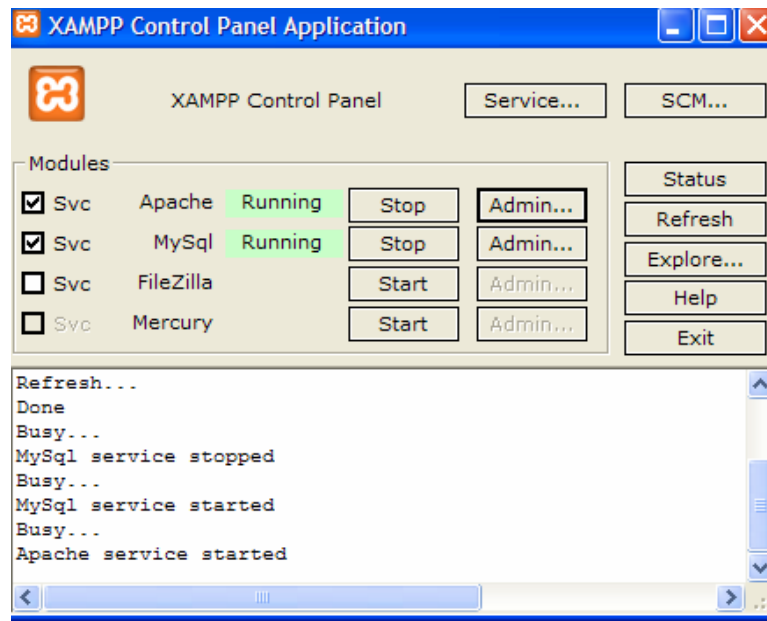


Figura 51: Panel de control de XAMPP básico



Figura 52: Pantalla de selección de idioma de XAMPP

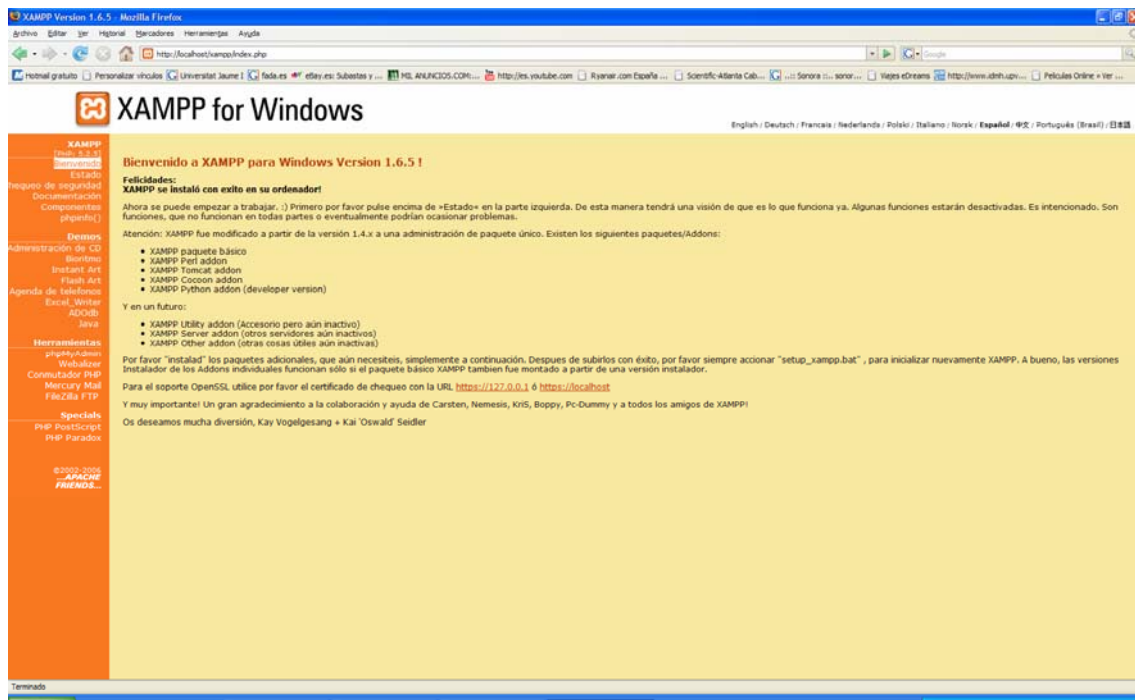


Figura 53: Pantalla de bienvenida a la configuración de XAMPP

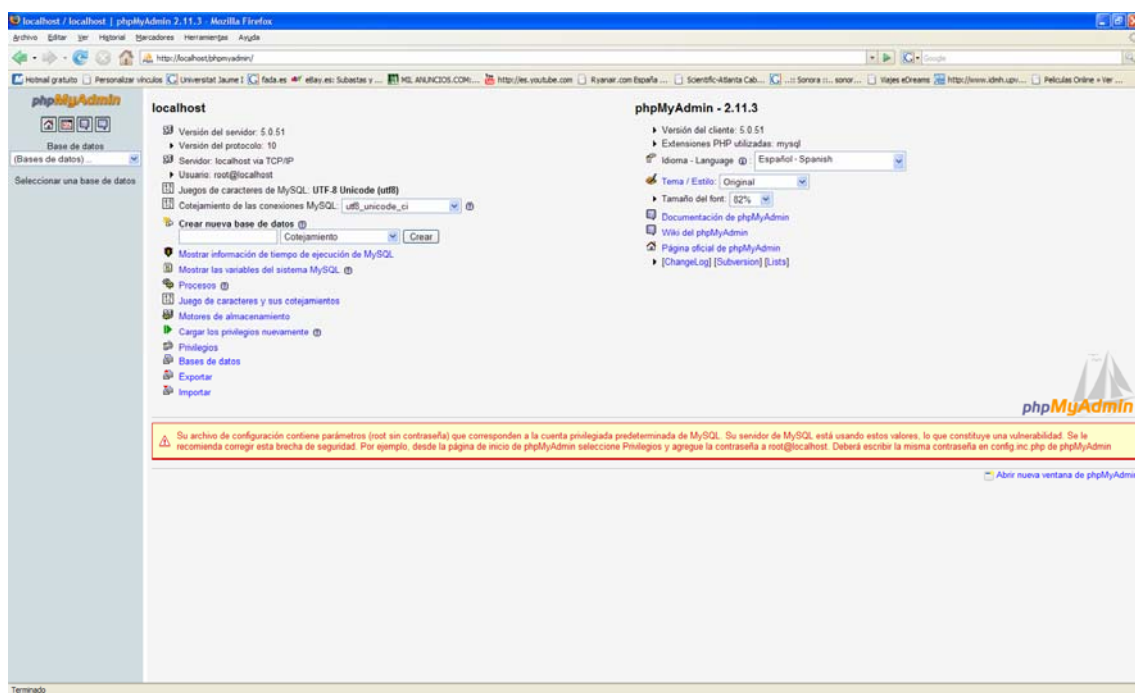


Figura 54: Pantalla de configuración de MySQL (phpMyAdmin)

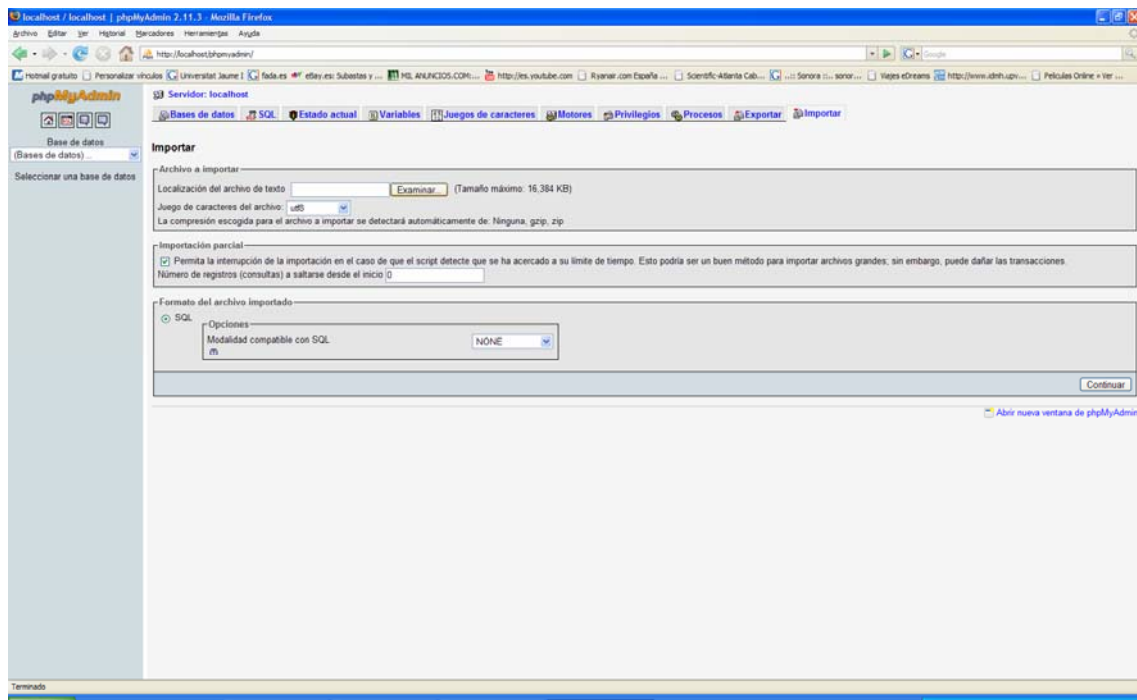


Figura 55: Pantalla para importar un archivo en MySQL

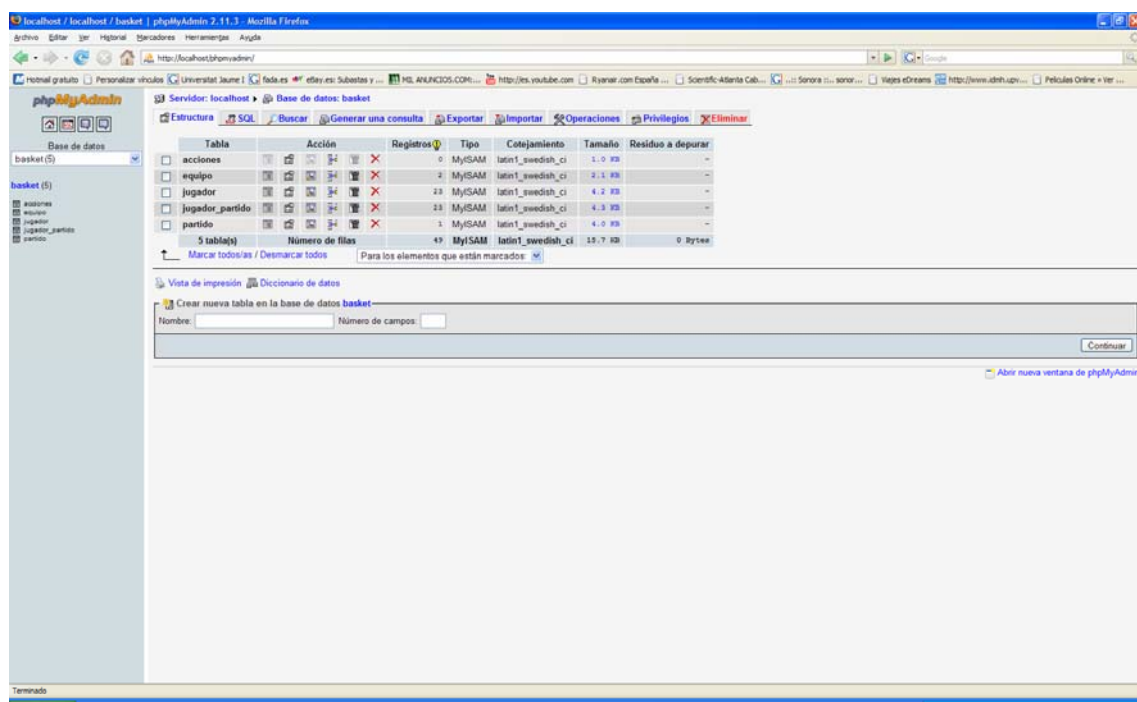


Figura 56: Pantalla de visualización de la base de datos Basket

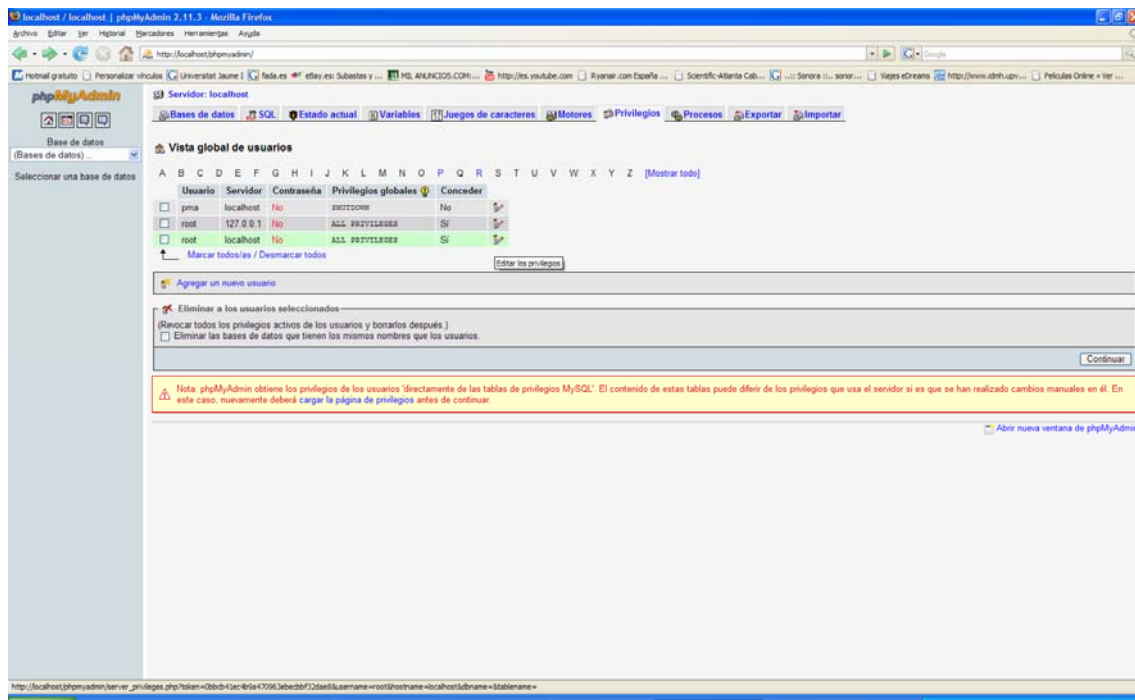


Figura 57: Pantalla de configuración de privilegios y permisos de los usuarios de MySQL

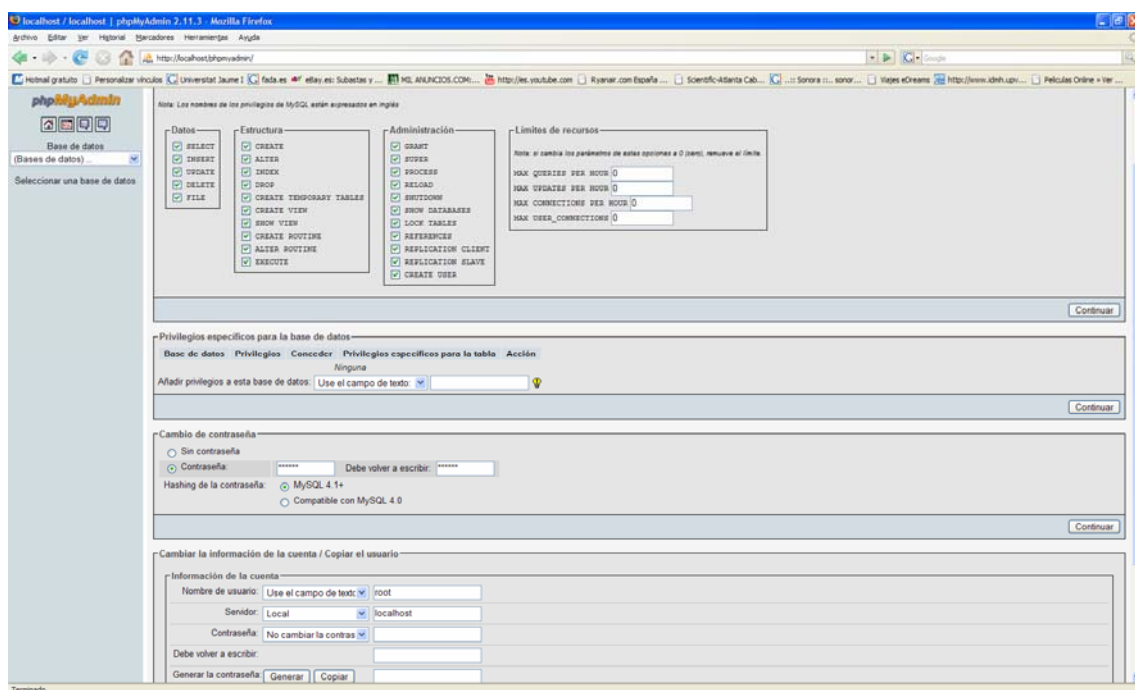


Figura 58: Modificación de la contraseña del usuario root

Con la instalación del ADD-On o extensión que se detalla en el siguiente subapartado, se instala el servidor Tomcat, que es capaz de servir páginas dinámicas JSP. De esta forma con el gestor de bases de datos y Tomcat se instala toda la arquitectura necesaria para que la aplicación web del proyecto funcione.

Además, gracias al paquete original de XAMPP se tiene acceso al servidor Apache, que permite servir páginas dinámicas en PHP, por lo que esta página del proyecto en JSP se podrá integrar dentro de un portal web de un equipo de baloncesto escrito en PHP. Esta ventaja era un requisito en el que estaba interesado el Director de proyecto.

ADD-Ons Tomcat para XAMPP [25]

Una vez instalada la versión básica de XAMPP y configurada la base de datos, se procede a instalar la extensión de XAMPP consistente en el servidor Tomcat. Tomcat es un servidor web con soporte para servlets y JSPs. Incluye el compilador Jasper, que compila páginas JSPs convirtiéndolas en servlets. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java, por lo que es necesario tener instalado el JDK de Java para instalar Tomcat.

El proceso de instalación es muy sencillo; sólo es necesario indicar la ruta en que se instaló la versión básica de XAMPP y se instala rápidamente.

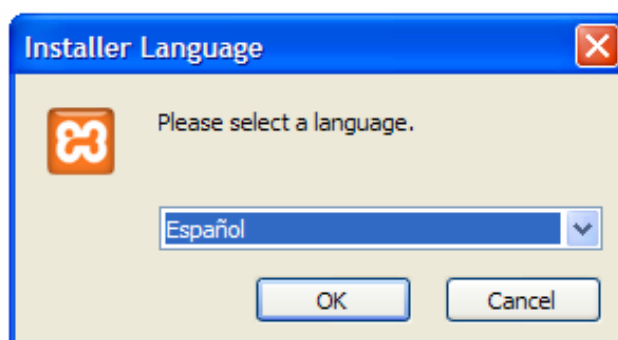


Figura 59: Proceso de instalación de la extensión Tomcat para XAMPP (Paso 1)

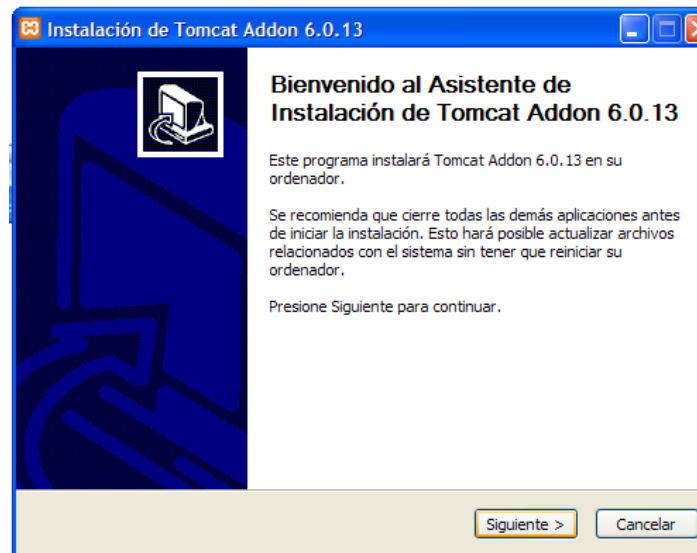


Figura 60: Proceso de instalación de la extensión Tomcat para XAMPP (Paso 2)

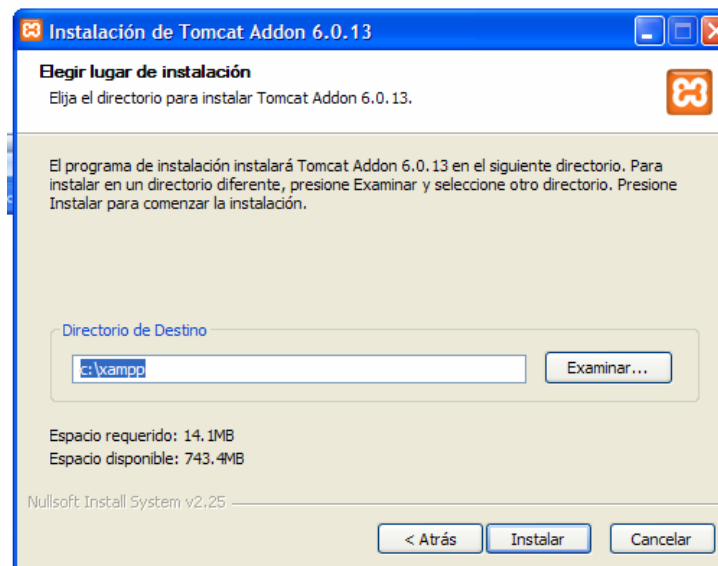


Figura 61: Proceso de instalación de la extensión Tomcat para XAMPP (Paso 3)

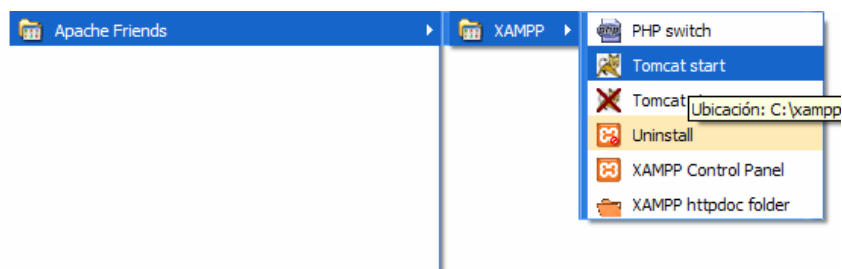


Figura 62: Inicialización del servidor Tomcat desde el menú de Inicio

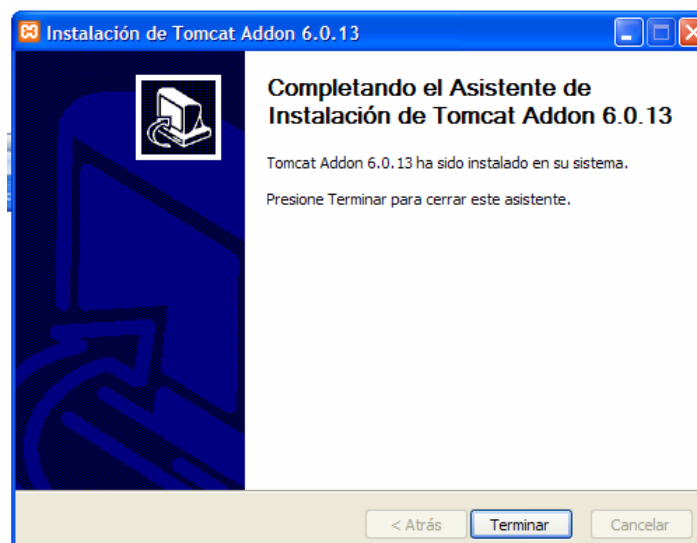


Figura 63: Proceso de instalación de la extensión Tomcat para XAMPP (Paso 4)

Una vez instalado el servidor Tomcat, para configurar la aplicación web del proyecto, únicamente es necesario eliminar todo el contenido de la carpeta C:/xampp/tomcat/webapps/ROOT (o en vez de C:\ la ruta en que se haya instalado XAMPP) y una vez eliminado el contenido, se copia en esta carpeta todo el contenido de la carpeta "Aplicación web" que se proporciona con el proyecto.

Una vez hecho esto, ya está configurada la aplicación web, sólo es necesario iniciar el servidor Tomcat, por ejemplo, desde el menú inicio, presionando "Tomcat Start" (ver *Figura 62:*), y una vez iniciado, abrir un navegador web y escribir <http://localhost:8080/index.jsp>, con lo que tendrá que aparecer la página principal de la aplicación web si todo se ha realizado correctamente.

De esta forma ya estará correctamente instalado y en funcionamiento el servidor de la aplicación web. Esta instalación funciona localmente en el ordenador en que se instale el servidor, y también en su red local cambiando en la URL del navegador *localhost* por la dirección IP del ordenador donde se haya instalado el servidor. Para que esta aplicación fuese accesible por un ordenador de una red externa, sería necesario abrir el puerto 8080 del router que gestione la red en que esté el ordenador que actúa como servidor, o alojar la aplicación en un servidor web de pago, pero estos hechos ya sobrepasan el alcance de este proyecto.