Universidad de Valladolid

E. T. S. de Ingeniería Informática

(Grado en Ingeniería Informática)

Software de apoyo para para una competición de robots de un torneo FLL

Autor: Alberto Fernández Delgado

Tutor: Cesar Vaca González

Agradecimientos

Tras un largo camino ha llegado el momento de agradecer a todas aquellas personas que han estado a mi lado y me han ayudado a llegar hasta aquí.

En primer lugar agradecer a mi tutor del proyecto Don César Vaca por haberme ayudado a tomar las distintas decisiones técnicas durante el desarrollo del proyecto, por haber estado ahí para resolver las dudas que me han ido surgiendo y por haberme dado la oportunidad de participar en la organización del evento First Lego League (FLL) celebrado en Valladolid.

También me gustaria nombrar a mi familia y amigos que me han apoyado desde el momento que empece a estudiar en la universidad.

Por último, no se me puede olvidar agradeceder enormemente a Alicia Marcos por haberme dicho las palabras que necesitaba oir para animarme a continuar con mis estudios cuando he pasado por malas etapas de mi vida, sin ella, hoy no estaría escribiendo esta memoria de proyecto de fin de carrera.

A todos ellos, una vez más, gracias.

Indice de contenidos

Indice de ilustraciones

Bloque I. Introducción

Capítulo 1. Introducción

1.1 Descripción del proyecto

El objetivo de este proyecto es implementar o materializar un producto software que sea capaz de ayudar a los organizadores de un evento FLL a arbitrar y mostrar los resultados en tiempo real a los espectadores que asistan al evento FLL, así como, automatizar ciertas tareas que hasta ahora se hacía de forma manual, como pueden ser la generación de informes de las diferentes partidas que se disputan en un evento FLL.

A este software lo hemos bautizado con el nombre de "FLL+". Está programado en lenguaje PHP, mantiene los datos de forma persistente con ayuda de un sistema gestor bases de datos, en concreto, MySQL y sirve las peticiones a través de un servidor web Apache, en definitiva, el producto software que hemos creado es una página web dinámica que se basa en la realización de peticiones asíncronas mediante AJAX para mostrar la información en tiempo real a los distintos stakeholders de la aplicación.

Al ser una aplicación web alojada en un servidor apache, FLL+ puede mostrar los diferentes eventos que ocurran a lo largo de la celebración de un torneo no solo a personas que estén físicamente en el lugar, sino también a cualquier persona que disponga de un dispositivo con conexión a internet.

1.2 Alcance del proyecto

La aplicación web FLL+ está pensada para ser utilizada en eventos FLL (First Lego League) por el personal perteneciente a la organización, así como, por los espectadores ya sea de forma local o remota. El orden de los usuarios que pueden estar haciendo uso de FLL+ se le puede suponer de cientos, ya que, suelen ser los asistentes medios entre organización, equipos participantes y espectadores que acuden a un evento de estas características.

A los usuarios finales de la aplicación se les supone unos conocimientos básicos a nivel usuario para poder hacer un uso razonable y correcto de la aplicación, en cualquier caso, para facilitar el uso de la aplicación, se ha construido un manual de usuario en el que se detallan de forma muy precisa como llevar a cabo cada una de las diferentes funcionalidades que ofrece FLL+.

1.3 Objetivos

El objetivo fundamental de la aplicación que se desea construir es el siguiente:

-Realizar un producto software que apoye a los organizadores de un evento FLL a arbitrar, mostrar los resultados al público, gestionar las partidas y automatizar todas las tareas que conlleva la celebración de un evento FLL.

A continuación vamos a desglosar este objetivo en objetivos más pequeños ya que este objetivo principal engloba varios elementos diferentes:

-Permitir a los árbitros arbitrar una partida de FLL de forma cómoda e intuitiva, a través de una interfaz sencilla pero potente.

-Permitir al jefe de árbitros gestionar las partidas, es decir, poder crearlas, eliminarlas y modificarlas.

-Permitir al jefe de árbitros exportar hojas de puntuación de forma automatizada sin necesidad de realizar la labor tediosa de rellenarlas a mano.

-Permitir tanto al público que esté presente de forma física en el evento como al público que no lo esté, seguir la evolución del evento FLL a través de un dispositivo con conexión a internet.

En definitiva el objetivo es realizar una aplicación que facilite radicalmente las tareas de organización de un evento FLL.

1.4 Motivación

La principal motivación para la realización de este proyecto es que hasta este momento no existía un software dedicado concretamente a administrar, gestionar y automatizar las tareas propias de un evento FLL. Algunas tareas como el arbitraje, hasta día de hoy, se hacían con un árbitro tomando datos a mano para luego pasarle un papel al jefe de árbitros que tenía que introducir esos datos de forma manual en un dispositivo para publicar los resultados tanto al público como a los equipos que participaban.

Esta labor en muchos casos es tediosa y repetitiva, sin embargo FLL+, permite arbitrar, gestionar y administrar las partidas que forman un evento FLL de forma interactiva y en tiempo real. Por poner un ejemplo, un árbitro puede arbitrar una partida con un dispositivo móvil tipo Tablet, que estará enviando los datos a un servidor que publicará los datos de forma automática, haciendo así, que el evento arroje una imagen más profesional a la vez que facilita la labor a los organizadores del evento FLL.

1.5 Definiciones, acrónimos y abreviaturas

FLL (First lego league) es el nombre que Lego ha dado a una competición en la que unos robots programables deben llevar a cabo una serie de pruebas en un tablero de juego.

Prueba Hace referencia a cada uno de los diferentes objetivos que un robot lego debe conseguir para obtener una puntuación concreta.

Jefe de árbitros Es la persona o personas encargadas de gestionar, administrar y controlar a los árbitros, con el fin de que las puntuaciones obtenidas por cada uno de los equipos en cada una de las rondas sea lo más ajustada a la realidad.

Arbitro Persona encargada de arbitrar las diferentes partidas que conforman un evento FLL

1.6 Documentación

En este apartado resumiremos brevemente como está organizada o esquematizada esta memoria así como la información que podemos encontrar en cada uno de los bloques en los que está dividida.

Bloque I Introducción En este bloque, que es en el que nos encontramos, damos una visión general del proyecto, su alcance, motivación y estructura de la memoria del mismo. El objetivo es presentar de forma breve en que consiste el software que se ha realizado.

Bloque II Análisis En el bloque de análisis, pretendemos presentar que es lo que queremos hacer, definir los actores que interactúan con el sistema y presentar los primeros diagramas UML de análisis, en definitiva, el objetivo de este bloque es definir qué es lo vamos a hacer.

Bloque III Diseño El bloque de diseño pretende explicar cómo vamos a hacer el sistema, presentar los diagramas UML de diseño, dar una visión global de la arquitectura del sistema, así como, explicar cómo se estructurarán y funcionaran las capas de presentación, lógica y persistencia de datos de la aplicación.

Bloque IV Implementación En este bloque explicaremos como hemos llevado a cabo la implementación del sistema estudiado en los bloques II y III, así como las tecnologías, frameworks, plugins… utilizados para programar la aplicación.

Bloque V Pruebas y resultados En el bloque V, presentaremos las diferentes pruebas que hemos lanzado sobre la aplicación y los resultados que hemos obtenido, así como, si estos eran los esperados o si se podrían mejorar.

Bloque VI Manual de usuario En el bloque de manual de usuario, explicaremos que se necesita para poder utilizar la aplicación, como instalarla y por supuesto como utilizarla.

Bloque VII Conclusiones Una vez se haya terminado de implementar la aplicación y de realizar el manual de usuario, pasaremos al apartado de conclusiones, en el que explicaremos que cosas hemos, y también, que cosas han ido bien y mal sin olvidar que cosas se pueden mejorar.

Bloque VIII Apéndices En este último bloque, presentaremos el contenido del CD adjunto con el proyecto y las referencias o recursos en los que nos hemos apoyado para analizar, diseñar e implementar FLL+.

Capítulo 2. FLL

2.1 ¿Qué es FLL?

FIRST LEGO League (FLL) es un torneo que desafía a jóvenes de 10 a 16 años con una temática del mundo real. A través de la resolución del desafío los jóvenes se entusiasman con la ciencia y la tecnología, y aprenden valiosas habilidades para su futuro profesional y para la vida.

Los equipos, compuestos por un máximo de diez miembros y al menos un entrenador adulto, pueden asociarse en un club u organización preexistente, o simplemente ser un grupo de amigos que quieran hacer algo impresionante.

En FLL, los niños hacen el trabajo, y el trabajo es programar un robot autónomo (utilizando el set LEGO ® MINDSTORMS ®) para sumar puntos en una superficie de juego temática, intentando conseguir el mayor número de puntos haciendo que su robot realice una serie de pruebas bien definidas por la FLL.

2.2 Valores FLL

Los valores en los que se basa un torneo FLL son los siguientes:

-Somos un equipo

-Trabajamos para encontrar soluciones con la ayuda de nuestros entrenadores

-Honramos el espíritu de una competición amistosa

-Lo que descubrimos es más importante que lo que ganamos

-Compartimos nuestras experiencias con los demás

-Mostramos cortesía profesional en todo lo que hacemos

-¡Nos divertimos!

Mientras se divierten aprendiendo, desarrollan habilidades de trabajo en equipo, emprendimiento, innovación, creatividad y comunicación que utilizarán para toda la vida.

2.3 ¿Qué pruebas existen en un torneo FLL?

Como ya hemos mencionado, el objetivo en un torneo FLL es obtener la mayor puntuación en un tablero de juego haciendo que el robot, programado por cada uno de los equipos, interactúe con una serie de pruebas para alcanzar un objetivo diferente en cada una de ellas.

A continuación pasamos a detallar esas pruebas y el objetivo de cada una.

2.3.1 Apertura de puertas

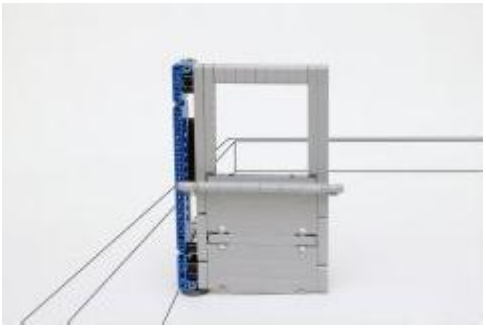


Figura 2.1 Apertura de puertas en estado inical

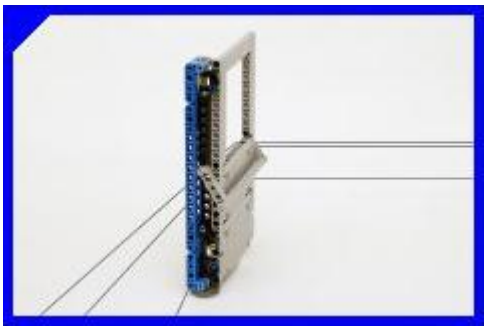


Figura 2.2 Apertura de puertas superada

Esta prueba consiste en que el robot sea capaz de abrir una puerta. Para hacerlo debe bajar la manija por completo y pasar a través de ella.

El estado inicial de la prueba se puede ver en la figura 2.1.

El final de la prueba superada, es decir, habiendo conseguido abrir la puerta, puede verse en la figura 2.2.

2.3.2 Acceso a la nube

Figura 2.3 Acceso a la nube estado incial

Figura 2.4 Acceso a la nube superada

En esta prueba, el objetivo es que el robot introduzca una tarjeta a través de los orificios de los que dispone la prueba. Al hacerlo se levantará una tarjeta de color azul y rojo.

El estado inicial de la prueba puede verse en la figura 2.3.

El estado de la prueba superado puede verse en la figura 2.4.

2.3.3 Adaptación a condiciones cambiantes

Figura 2.5 Adaptación a condiciones cambiantes estado inicial

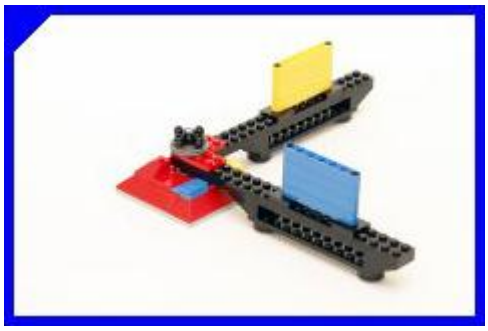


Figura 2.6 Adaptación a condiciones cambiantes superada

El objetivo de esta prueba es bien sencillo. Se basa en conseguir que el objeto gire 90 grados.

El estado inicial de la prueba es el mostrado en la figura 2.5.

Su estado final, en caso de que se haya superado, es el mostrado en la figura 2.6.

2.3.4 Aprendizaje

Figura 2.7 Aprendizaje estado inicial

Figura 2.8 Aprendizaje modelo construido

Figura 2.9 Aprendizaje superada

En esta prueba, los participantes deben construir un pequeño modelo de lego y presentárselo al árbitro, con eso, ya reciben una bonificación de puntos. Después, el robot Lego, debe ser capaz de llevar ese modelo hasta un punto determinado.

En la figura 2.7, se muestra dos personajes Lego. Esta imagen correspondería al estado inicial de la prueba.

En la figura 2.8, se puede ver un ejemplo de modelo Lego construido por el equipo.

Por último, en la figura 2.9, se puede observar el estado de la prueba en el que se conseguiría la mayor puntuación, es decir, con el modelo presentado al árbitro y además situado en el área correspondiente.

2.3.5 Aprendizaje basado en proyectos

Figura 2.10 Aprendizaje basado en proyectos (8 asas)

En esta prueba, el objetivo es conseguir colocar el mayor número de asas posibles sobre un soporte. Cuantas más asas se coloquen más puntos se obtendrán. Estas asas se consiguen superando otras pruebas.

En la figura 2.10, se muestra la prueba con varias asas colocadas por el robot.

2.3.6 Aprendizaje comunitario

Figura 2.11 Aprendizaje comunitario estado inicial

Figura 2.12 Aprendizaje comunitario superada

En la prueba “Aprendizaje comunitario” el robot Lego debe ser capaz de sacar un asa de la prueba, de tal forma, que el asa no toque la prueba.

En la figura 2.11, podemos ver la prueba en su estado inicial.

En la figura 2.12 se aprecia la prueba superada, es decir, con el asa retirada del modelo y sin tocarlo.

2.3.7 Captar la atención

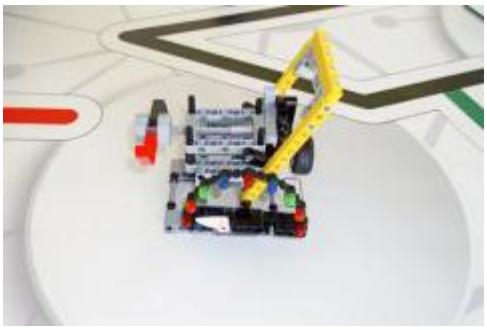


Figura 2.13 Captar la atención estado inicial

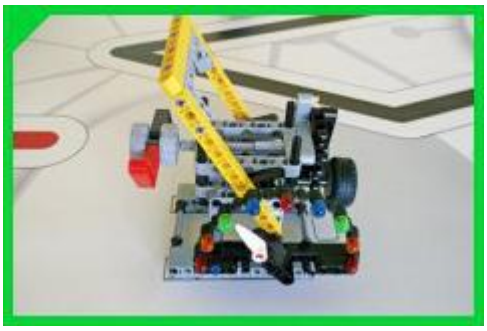


Figura 2.14 Captar la atención con palanca colocada

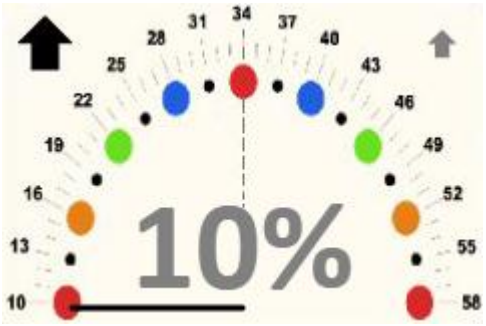


Figura 2.15 Captar la atención (ejemplo multiplicador)

Esta prueba es una de las que más puntos puede dar al equipo. El primer objetivo que tiene que lograr el robot Lego es mover la palanca de color amarillo. Después, debe girar una ruleta todas las veces que pueda para intentar que el indicador suba lo máximo posible. Los puntos obtenidos en las otras pruebas se multiplicarán por el número que marca el indicador.

En la figura 2.13 se observa la prueba en su estado inicial.

En la figura 2.14, podemos ver la palanca amarilla colocada en su posición.

Por último en la figura 2.15, podemos ver un ejemplo del multiplicador, el cual, marcará una puntuación mayor cuantas más vueltas haya dado el robot a la ruleta

2.3.8 Competición de robótica

Figura 2.16 Competición de robótica estado inicial

Figura 2.17 Competición de robótica con modelo colocado

Figura 2.18 Competición de robótica superada

En esta prueba, hay dos objetivos. El primero de ellos es introducir una figura lego dentro del modelo de la prueba. El segundo es empujar este modelo hasta que se libere el asa y deje de tocar la prueba.

En la figura 2.16 se muestra la prueba en su estado inicial.

En la figura 2.17 se puede ver como se ha colocado el modelo lego dentro de la prueba.

Por último, en la figura 2.18, se muestra la prueba en su estado "superada", es decir, el asa ha sido liberada y ya no está tocando la prueba.

2.3.9 Comunicación

Figura 2.19 Comunicación en estado inicial

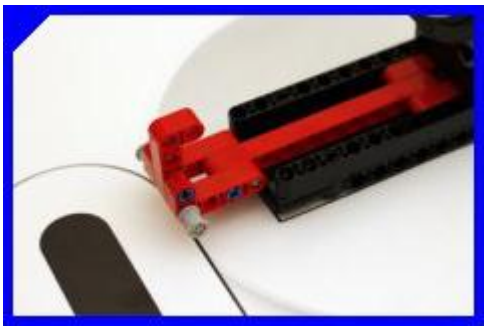


Figura 2.20 Comunicación superada

En la prueba "Comunicación" el objetivo está bien claro. Consiste en conseguir que el robot Lego tire del soporte de la prueba.

La figura 2.19 muestra la prueba en su estado inicial.

En la figura 2.20 podemos ver como el robot ha desplazado el soporte.

2.3.10 Deportes

Figura 2.21 Deportes en estado inicial

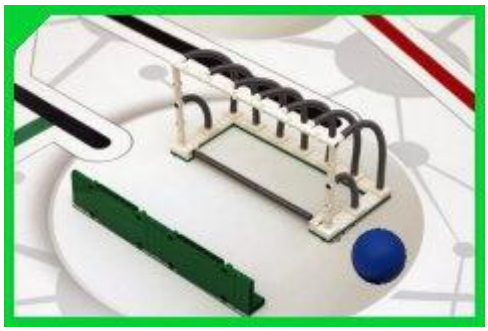


Figura 2.22 Deportes con disparo fuera de la porteria

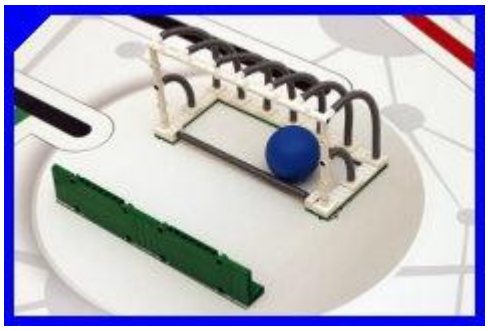


Figura 2.23 Deportes superada

La prueba “Deportes” es una de las más vistosas para los espectadores. Consiste en que el robot Lego meta un gol con una pequeña pelota de plástico. La prueba puede quedar en 3 estados que son, el robot no hizo ningún lanzamiento, el robot hizo un lanzamiento pero no metió gol y el robot hizo un lanzamiento y metió gol.

La figura 2.21 muestra el estado de la prueba en su estado inicial.

En la figura 2.22 puede verse como el robot hizo un lanzamiento pero no metió gol.

Por último en la figura 2.23, puede verse la prueba completada (máxima puntuación), el robot hizo un lanzamiento y metió gol.

2.3.11 Ingeniería inversa

Figura 2.24 Ingeniería inversa en estado inicial

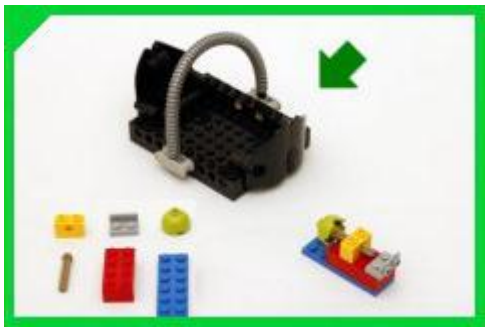


Figura 2.25 Ingeniería inversa con modelo construido

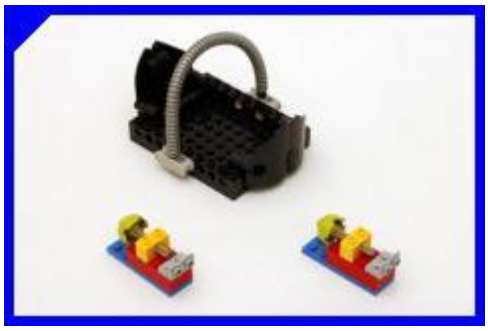


Figura 2.26 Ingeniería inversa superada

La prueba “Ingeniería inversa” consiste en construir un modelo de Lego e introducirlo en una cesta que se situará en el tablero de juego del equipo rival. El otro equipo deberá conseguir que su robot coja la cesta, la lleve a la base y construya un modelo idéntico al que creó el equipo contrincante.

En la figura 2.24, se muestra la prueba en su estado inicial.

La figura 2.25, refleja el estado en el que el equipo consiguió llevar la cesta a la base pero no construyó un modelo idéntico al creado por el equipo rival.

Por último, en la figura 2.26 se puede ver la prueba en su estado de máxima puntuación, es decir, la cesta se encuentra en la base y además el modelo construido es idéntico al creado por el rival.

2.3.12 Motor de búsqueda

Figura 2.27 Motor de búsqueda en estado inicial

Figura 2.28 Motor de búsqueda con palanca desplazada

Figura 2.29 Motor de búsqueda superada

En esta prueba, el robot lego debe empujar un soporte que producirá un movimiento giratorio de una ruleta que contiene 3 colores. Una vez haya girado, el robot deberá recoger el asa del color que indique la ruleta.

La figura 2.27 muestra la prueba en su estado inicial.

La figura 2.28, nos muestra el estado de la prueba cuando el robot empujo el soporte he hizo girar la ruleta.

Por último, en la figura 2.29 vemos que el robot ha retirado el asa que indica la ruleta, es decir, el asa de color azul.

2.3.13 Pensamiento fuera de la caja

Figura 2.30 Pensamiento fuera de la caja en estado inicial

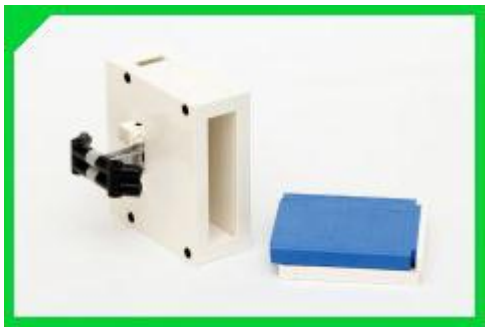


Figura 2.31 Pensamiento fuera de la caja con idea bocaabajo

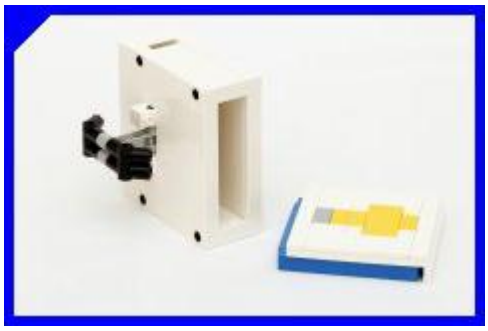


Figura 2.32 Pensamiento fuera de la caja superada

La prueba “Pensamiento fuera de la caja” consiste en liberar una tarjeta construida con piezas de lego. Esta tarjeta debe quedar mirando hacia arriba, es decir, se debe poder ver la imagen de la bombilla.

En la figura 2.30 se muestra la prueba en su estado inicial.

En la figura 2.31 podemos ver como el robot logró sacar la tarjeta de la prueba pero esta quedó mirando hacia abajo no hacia arriba.

Por último, en la figura 2.32, se aprecia la prueba en su estado de máxima puntuación, es decir, con la tarjeta fuera de la caja y la imagen de la bombilla visible.

2.1.14 Uso de los sentidos correctos

Figura 2.33 Uso de los sentidos correctos en estado inicial

Figura 2.34 Uso de los sentidos correctos superada

En esta prueba el objetivo es bien sencillo. El robot debe empujar un deslizador para hacer que el asa se libere. Una vez se haya liberado el asa, el robot debe cogerla para hacer que esta no toque la prueba.

En la figura 2.33 podemos ver la prueba en su estado inicial.

En la figura 2.34, podemos ver el estado de la prueba en su estado de puntuación máxima, es decir, con el deslizador desplazado y el asa completamente liberada (Sin tocar el modelo).

Capítulo 3. Fundamentos teóricos

3.4 PHP

PHP es el lenguaje en el que está escrito la lógica de FLL+, por ello, en este apartado daremos una breve introducción en la que explicaremos que es PHP y cuáles son sus principales características.

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en código HTML.

En lugar de usar muchos comandos para mostrar HTML (como ocurre en C o en Perl), las páginas escritas en PHP contienen HTML con código incrustado que hace "algo", generalmente mostrar al usuario datos generados de forma dinámica. El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo del lado del cliente como puede ser JavaScript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no podrá conocer el código subyacente que se ejecutó en el servidor, al contrario de lo que ocurre con JavaScript, el cual, puede ser visualizado fácilmente viendo el código fuente de cualquier página web. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué hay detrás de lo que el navegador web les está mostrando.

Una característica básica del lenguaje PHP es su extrema simplicidad a la hora de escribir código, ya que, entre otras cosas, es un lenguaje débilmente tipado, pero a su vez ofrece muchas características avanzadas para los programadores más avanzados.

Aunque el desarrollo de PHP está centrado en la programación de scripts del lado del servidor, se puede utilizar para muchas otras cosas, de hecho, desde hace ya varios años PHP permite al programador utilizar el paradigma de programación orientada a objetos, lo que le dota de un gran potencial a la hora de elaborar páginas web dinámicas, casi como si fuese una aplicación de escritorio corriente con la salvedad de que necesita hacer uso del protocolo HTTP para transferir los datos hacía el navegador web del cliente.

3.2 JavaScript

En la parte de presentación al usuario o interfaz gráfica, JavaScript ha jugado un papel muy importante sobre el que se ha apoyado el framework jQuery, que introduciremos más adelante, por ello, merece un apartado en el capítulo de fundamentos teóricos. A continuación, pasamos a explicar que es JavaScript y cuáles son las características principales de este lenguaje.

JavaScript es un lenguaje que no requiere de compilación ya que el lenguaje funciona del lado del cliente, es decir, los navegadores son los encargados de interpretar estos códigos.

Algunas personas confunden JavaScript con Java pero ambos lenguajes son diferentes y tienes sus propias características. JavaScript tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado.

Como síntesis se pude decir que JavaScript es un lenguaje interpretado, basado en prototipos, mientras que Java es un lenguaje más orientado a objetos.

JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript podemos crear diferentes efectos e interactuar con nuestros usuarios.

Este lenguaje posee varias características, entre ellas podemos mencionar que es un lenguaje basado en acciones que posee menos restricciones. Además, es un lenguaje que utiliza Windows y sistemas X-Windows, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas o cargas de páginas entre otros.

3.3 MySQL

FLL+ mantiene la persistencia de datos en una base de datos MySQL. En ella guarda datos como los equipos, partidas, puntuaciones, usuarios… por ello vamos a explicar brevemente que es MySQL y cuáles son sus principales características.

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

3.4 jQuery

En la implementación de gran parte de la presentación al usuario de la aplicación FLL+ hemos utilizado jQuery, que básicamente es un framework de JavaScript que facilita la escritura de este lenguaje. En las etapas iniciales del desarrollo del proyecto empezamos utilizando JavaScript sin jQuery, sin embargo, a lo largo del desarrollo indagamos un poco en este framework y enseguida nos dimos cuenta de que jQuery podía hacer mucho por nosotros, facilitándonos la tarea considerablemente, por tanto, haremos un breve introducción en este apartado sobre jQuery.

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en NYC. jQuery es la biblioteca de JavaScript más utilizada.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

3.5 UML

En las etapas de análisis y diseño de FLL+, se han realizado varios diagramas en UML que nos han ayudado a entender el problema que teníamos que resolver y cómo resolverlo. A continuación, haremos una breve descripción explicando que es UML.

El término “lenguaje” ha generado bastante confusión respecto a lo que es UML. En realidad el término lenguaje quizás no es el más apropiado, ya que UML no es un lenguaje propiamente dicho, sino una serie de normas y estándares gráficos respecto a cómo se deben representar los esquemas relativos al software. Mucha gente piensa por confusión que UML es un lenguaje de programación y esta idea es errónea: UML no es un lenguaje de programación. Como decimos, UML son una serie de normas y estándares que dicen cómo se debe representar algo.

UML es una herramienta propia de personas que tienen conocimientos relativamente avanzados de programación y es frecuentemente usada por analistas funcionales (aquellos que definen qué debe hacer un programa sin entrar a escribir el código) y analistas-programadores (aquellos que dado un problema, lo estudian y escriben el código informático para resolverlo en un lenguaje como Java, PHP, Python o cualquier otro).

3.6 DreamWeaver

En la etapa de implementación hemos utilizado la herramienta DreamWeaver para escribir el código PHP, JavaScript y HTML. Esta herramienta es muy utilizada por desarrolladores web, y en nuestro caso, nos ha facilitado enormemente la tarea a la hora de implementar FLL+, alertándonos de sintaxis incorrectas, facilitándonos la escritura de código HTML… Por tanto, como en los casos anteriores haremos una breve introducción en la que explicaremos que es DreamWeaver y en qué casos es recomendable utilizarlo.

Dreamweaver es probablemente el mejor editor de páginas web para diseñadores que busquen resultados profesionales.

Dreamweaver es la herramienta de diseño de páginas web más avanzada, tal como se ha afirmado en muchos medios. Aunque sea un experto programador de HTML el usuario que lo maneje, siempre se encontrarán en este programa razones para utilizarlo, sobre todo en lo que a productividad se refiere.

Cumple perfectamente el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, además muy fáciles de usar:

-Hojas de estilo y capas

-JavaScript para crear efectos e interactividades

-Inserción de archivos multimedia...

Además es un programa que se puede actualizar con componentes, que fabrica tanto Macromedia como otras compañías, para realizar otras acciones más avanzadas.

En resumen, el programa es realmente satisfactorio, incluso el código generado es de buena calidad. La única pega que podríamos decir sobre esta herramienta de diseño web consiste en que al ser tan avanzado, puede resultar un poco difícil su manejo para personas menos experimentadas en el diseño de páginas web.

3.7 Tecnología Ajax

El marcador, el planificador, el menú de configuración de arbitraje y el propio arbitraje del producto software FLL+ hace uso de la tecnología Ajax para poder mostrar a los diferentes actores que interactúan con la aplicación la información en tiempo real sin necesidad de que estos tengan que recargar continuamente la página para ver los resultados de forma actualizada, es por ello, que esta tecnología merece una descripción relativamente amplia ya que es la base que hace posible que FLL+ muestre la información en tiempo real.

Veamos un esquema para comprender mejor la idea que hay detrás del uso de Ajax. Este esquema lo iremos comentando y comprendiendo a medida que avancemos en la explicación. Ver figura 3.1.

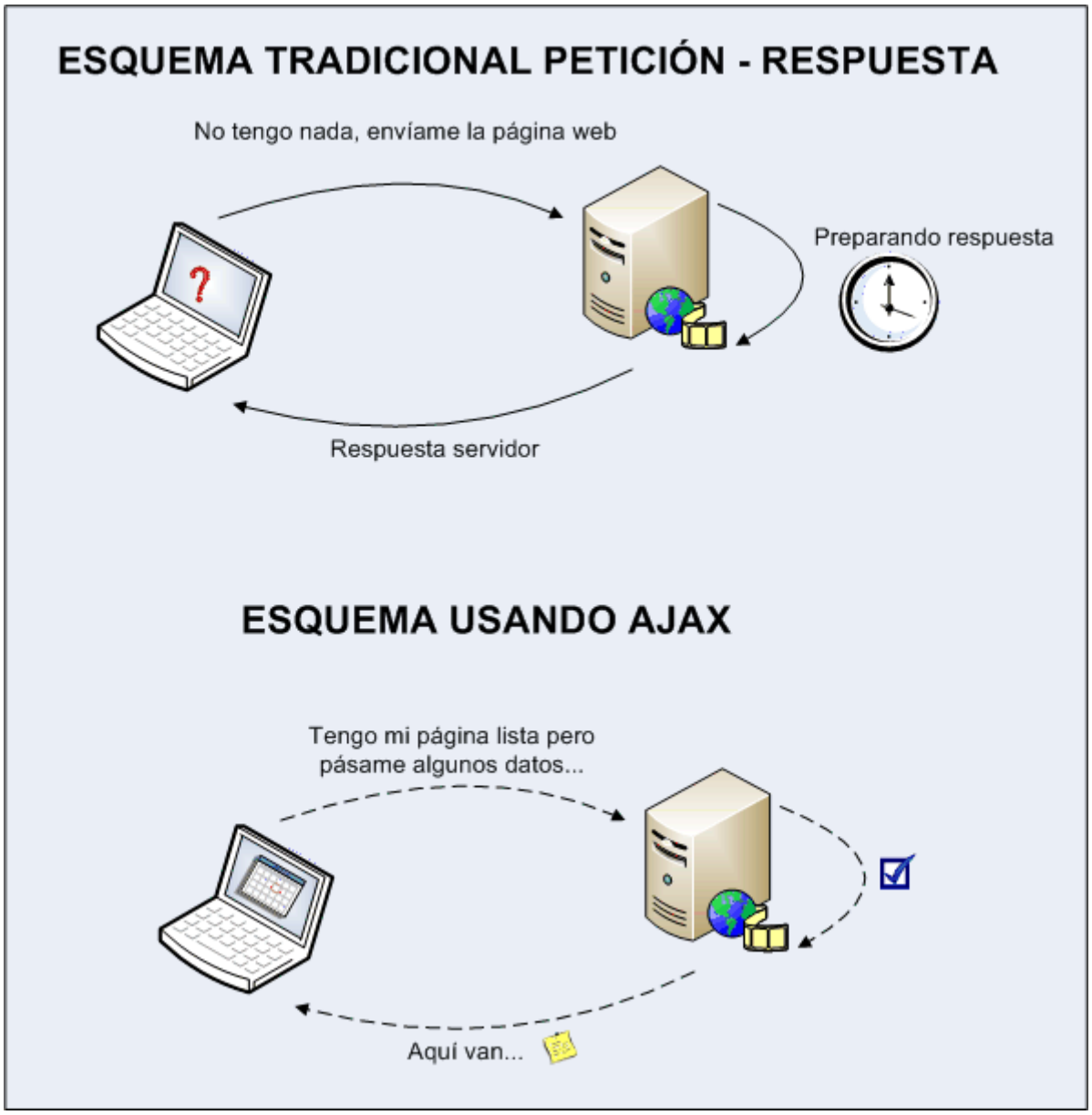


Figura 3.1 Esquema AJAX

Realizar peticiones al servidor y esperar respuesta puede consumir tiempo (el tiempo necesario para recargar una página completa). Para agilizar los desarrollos web surgió Ajax (inicialmente Asynchronous JavaScript And XML, aunque hoy día ya no es una tecnología ligada a XML con lo cual no pueden asociarse las siglas a estos términos), una tecnología que busca evitar las demoras propias de las peticiones y respuestas del servidor mediante la transmisión de datos en segundo plano usando un protocolo específicamente diseñado para la transmisión rápida de pequeños paquetes de datos.

Con Ajax, se hace posible realizar peticiones al servidor y obtener respuesta de este en segundo plano (sin necesidad de recargar la página web completa) y usar esos datos para, a través de JavaScript, modificar los contenidos de la página creando efectos dinámicos y rápidos.

En la figura 3.1 vemos las ideas en torno a Ajax de forma gráfica. En la parte superior hemos representado lo que sería un esquema de comunicación tradicional: el cliente solicita una página web completa al servidor. El servidor recibe la petición, se toma su tiempo para preparar la respuesta y la envía. El resultado, una pequeña demora debido al tiempo que

tarda en llegar la petición al servidor, el tiempo que éste tarda en preparar la respuesta, y el tiempo que tarda en llegar la respuesta más recargarse en el navegador.

En la parte inferior de la figura 3.1 vemos lo que sería un esquema de comunicación usando Ajax: el cliente tiene una página web cargada (puede ser una página web completa, o sólo el esqueleto de una página web). El cliente sigue trabajando y en segundo plano (de ahí que hayamos dibujado con líneas punteadas las comunicaciones) le dice al servidor que le envíe un paquete de datos que le hacen falta. El servidor procesa la petición. Ahora la respuesta es mucho más rápida: no tiene que elaborar una página web completa, sino sólo preparar un paquete de datos. Por tanto el tiempo de respuesta es más rápido. El servidor envía el paquete de datos al cliente y el cliente los usa para cambiar los contenidos que se estaban mostrando en la página web.

Las ventajas que proporciona Ajax son varias:

- No es necesario recargar y redibujar la página web completa, con lo que todo es más rápido.

- El usuario no percibe que haya demoras: está trabajando y al ser las comunicaciones en segundo plano no hay interrupciones.

- Los pasos que antes podía ser necesario dar cargando varias páginas web pueden quedar condensados en una sola página que va cambiando gracias a Ajax y a la información recibida del servidor.

Ajax, por supuesto, también tiene inconvenientes:

-El usuario puede perder la capacidad para hacer cosas que hacía con webs tradicionales puesto que no hay cambio de página web. Por ejemplo, usar los botones de avance y retroceso del navegador o añadir una página a favoritos puede dejar de ser posible. Esto en algunos casos no es deseable.

-El desarrollo de aplicaciones web se puede volver más complejo. Supongamos que antes tuviéramos un proceso en el que avanzábamos a través de varias páginas web como 1, 2, 3. De este modo la organización resulta sencilla. Si condensamos todo en una sola página web: 1, escribir y depurar el código puede volverse más complicado. En sitios complejos, puede ser muy difícil depurar errores.

-Existen problemas y restricciones de seguridad relacionados con el uso de Ajax. Hay que tener en cuenta que por motivos de seguridad no todos los procesos se pueden realizar del lado del cliente (que por su propia naturaleza es “manipulable”). También existen restricciones de seguridad para impedir la carga de contenidos mediante Ajax desde sitios de terceras partes.

-La indexación para los motores de búsqueda se ve dificultada, con lo cual nuestros sitios web pueden perder visibilidad en los buscadores. No es lo mismo un contenido “constante” o aproximadamente estático, fácilmente rastreable para un buscador, que un contenido “cambiante” en función de la ejecución de JavaScript, difícilmente rastreable para un buscador.

Por último cabe mencionar que Ajax no es un lenguaje de programación sino un conjunto de técnicas que se usan para lograr un objetivo y se basa en lenguajes ya existentes como JavaScript.

Podríamos dar esta definición de Ajax: “Ajax es un conjunto de métodos y técnicas que permiten intercambiar datos con un servidor y actualizar partes de páginas web sin necesidad de recargar la página completamente, es decir, de forma asíncrona”.

Aunque Ajax se pensó inicialmente para transferir datos en un solo formato (XML), actualmente Ajax permite la transmisión de datos en múltiples formatos: XML, JSON, EBML, texto plano, HTML, etc.

3.8 Web services

El marcador, el planificador de partidas, el arbitraje… de FLL+ no son más que llamadas asíncronas a un servicio web, el cual, devuelve una serie de datos en JSON, que el navegador web es capaz de interpretar y procesar con la ayuda de JavaScript para ser mostrados al usuario en el lugar y con la forma adecuada.

A continuación pasamos a explicar brevemente que es un servicio web.

Un servicio web (en inglés, Web Service o Web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

Para entender mejor este último párrafo vamos a poner un ejemplo. Imaginemos que deseamos construir una aplicación que nos muestre en tiempo real la información meteorológica de un lugar concreto. Pues bien, existen servicios web a los que podemos enviar una petición pasándole unas coordenadas y que nos devolverán la información meteorológica de dicho lugar definido por esas coordenadas. Normalmente las respuestas de los web services suelen ser en XML, JSON… dado que permiten al cliente del servicio web poder procesar de forma adecuada los datos de respuesta del servidor.

3.8 JSON

En el caso de la herramienta FLL+, las peticiones a los servicios web de los que dispone y que explicaremos de forma detallada más adelante, producen respuestas en formato JSON, por ello, merece en esta memoria una descripción al menos pormenorizada de este formato.

JSON (JavaScript Object Notation) es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una alternativa a XML, el fácil uso en JavaScript ha generado un gran número de seguidores de esta alternativa. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

A continuación vamos a ver un ejemplo de una respuesta JSON por parte de un servicio web para que queden más claros estos últimos párrafos.

```
{"Clasificacion":
 {"Equipos":
  [
   {"Nombre":"Locas Tuercas","Puntuacion":10},
   {"Nombre":"RoboMarques","Puntuacion":20},
   {"Nombre":"I.E.S Nuñez de arce","Puntuacion":30},
  {"Nombre":"Roboticos","Puntuacion":100}
  ]
 }
}
```

En este ejemplo podemos ver que hemos creado un objeto llamado clasificación.

Este objeto contiene un array de equipos, en concreto, contiene 3 equipos:

-Locas tuercas

-Robo Marqués

-I.E.S Nuñez de arce

-Roboticos

Como vemos, para cada equipo se muestra su nombre y la puntuación obtenida.

En el caso de FLL+ este ejemplo podría ser válido para gestionar la tabla donde se están almacenando los datos de la clasificación del evento. Una vez lleguen al navegador del cliente, estos datos, serán procesados y ubicados en una tabla en función de la puntuación obtenida, colocando al equipo que haya obtenido una puntuación mayor en las filas superiores de la tabla y viceversa.

Con esto podemos dar por concluido el apartado de fundamentos teóricos, ya que, hemos explicado los lenguajes, tecnologías y herramientas utilizadas en el desarrollo de FLL+.

Bloque II. Plan de desarrollo software

Capítulo 4. Introducción al plan de desarrollo de software

En este capítulo describiremos la planificación del proyecto, especificando las características más importantes del plan de desarrollo de software.

En primer lugar presentaremos los recursos disponibles para la realización del proyecto. Continuaremos con la descripción de las tareas a realizar, mostrando un calendario en el que mostraremos dichas tareas junto con la estimación temporal de las mismas. Esta estimación estará basada en la experiencia que tenemos de la realización de otros trabajos similares.

Por último, tras haber definido las tareas y su estimación temporal, presentaremos un estudio de los riesgos que podemos encontrar durante la realización del proyecto, evaluando el impacto en él y buscando posibles soluciones a cada uno de ellos.

La metodología empleada para la realización del proyecto será un modelo en cascada con fases de análisis, diseño, implementación y pruebas.

Capítulo 5. Identificación de recursos

5.1 Recursos humanos

Durante la realización del proyecto contaremos con dos personas. Por un lado el alumno, sobre el que recaerá los roles de jefe de proyecto, analista, diseñador, programador y verificador. Por otra parte estará presente el tutor cuyas funciones principales serán guiar y asesorar al alumno durante el transcurso del proyecto.

5.2 Recursos hardware

Los recursos hardware de los que dispondremos durante el desarrollo del proyecto serán:

-Máquina personal del alumno, desde la que se llevarán a cabo las tareas de diseño e implementación.

-Tablet personal del alumno, desde donde se comprobará que la página web resultante es responsive y se adapta correctamente a este tipo de dispositivos.

-Máquina virtual alquilada por el alumno a una empresa externa a la universidad, desde la cual, se realizarán las pruebas de funcionalidad y rendimiento de la aplicación web.

5.3 Recursos software

El desarrollo de la aplicación se realizará bajo un sistema operativo Windows 7 Ultimate y las pruebas de funcionalidad y rendimiento sobre un sistema operativo Linux con distribución Debian.

Las herramientas software que usaremos en las distintas fases del proyecto son las siguientes:

-Dreamweaver, para llevar a cabo la implementación del sistema.

-Astah, para realizar los diferentes diagramas en la fase de análisis y diseño.

-Microsoft Office Word para la redacción de la memoria.

-Microsoft Office Project para la realización de los diagramas del plan de desarrollo software.

-Xampp para realizar las pruebas. El paquete Xampp incluye un servidor web, PHP y MySQL.

-Microsoft Office Excel para construir las gráficas en la fase de pruebas.

Capítulo 6. Identificación de tareas y calendario

De la aplicación del modelo en cascada, se obtienen 4 fases determinantes en el proyecto:

Fase	Tareas
Fase de análisis	-Identificación de requisitos funcionales -Identificación de requisitos no funcionales -Identificación de actores del sistema -Realización del modelo de casos de uso -Descripción de los casos de uso -Realización del modelo de dominio
Fase de diseño	-Diseño de la base de datos -Diseño de la arquitectura del sistema -Identificación de patrones a utilizar -Realización de diagramas de secuencia -Realización de diagrama de clases de diseño
Fase de implementación	-Implementación del núcleo de la aplicación -Implementación del interfaz gráfico de usuario -Integración de los componentes
Fase de pruebas	-Realización de pruebas funcionales -Realización de pruebas de rendimiento

Una vez hemos definido las tareas o actividades de las que constará el desarrollo de nuestro proyecto, vamos a realizar un calendario en el que detallaremos la duración, los recursos necesarios para llevar a cabo cada una de las tareas identificadas previamente y las relaciones existentes entre las tareas.

Como ya mencionamos anteriormente, la estimación temporal de estas tareas está basada en la experiencia de realización de otros trabajos y el objetivo será ajustarnos lo máximo posible a la realidad, sin embargo, siempre es posible que ocurran imprevistos o retrasos. Esto último lo mencionaremos en el siguiente capítulo en el que daremos un informe detallado de la gestión de riesgos que puedan ocurrir durante el periodo de tiempo que dure el desarrollo del proyecto.

En las siguientes figuras podemos ver las actividades, su estimación y la relación entre ellas. También podemos ver un diagrama de Gantt de cada de una de las fases.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Fase Análisis**	**26 días**	**lun 16/02/15**	**lun 23/03/15**		
2	Identificación de requisitos funcionales	3 días	lun 16/02/15	mié 18/02/15		Analista
3	Identificación de recursos no funcionales	3 días	jue 19/02/15	lun 23/02/15	2	Analista
4	Identificación de actores del sistema	3 días	mar 24/02/15	jue 26/02/15	3	Analista
5	Realización del modelo de casos de uso	5 días	vie 27/02/15	jue 05/03/15	4	Analista
6	Descripción de los casos de uso	6 días	vie 06/03/15	vie 13/03/15	5	Analista
7	Realización del modelo de dominio	6 días	lun 16/03/15	lun 23/03/15	6	Analista

Figura 6.1 Actividades fase análisis

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Fase Análisis**	**26 días**	**lun 16/02/15**	**lun 23/03/15**		
8	**Fase Diseño**	**20 días**	**mar 24/03/15**	**lun 20/04/15**	**1**	
9	Diseño de la base de datos	4 días	mar 24/03/15	vie 27/03/15		Diseñador
10	Diseño de la arquitectura del sistema	2 días	lun 30/03/15	mar 31/03/15	9	Diseñador
11	Identificación de patrones a utilizar	2 días	mié 01/04/15	jue 02/04/15	10	Diseñador
12	Realización de diagramas de secuencia	6 días	vie 03/04/15	vie 10/04/15	11	Diseñador
13	Realización de diagramas de clases de diseño	6 días	lun 13/04/15	lun 20/04/15	12	Diseñador

Figura 6.2 Actividades fase diseño

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Fase Análisis**	**26 días**	**lun 16/02/15**	**lun 23/03/15**		
8	**Fase Diseño**	**20 días**	**mar 24/03/15**	**lun 20/04/15**	**1**	
14	**Fase Implementación**	**25 días**	**mar 21/04/15**	**lun 25/05/15**	**8**	
15	Implementación del núcleo de la aplicación	15 días	mar 21/04/15	lun 11/05/15		Programador
16	Implementación del interfaz gráfico	7 días	mar 12/05/15	mié 20/05/15	15	Programador
17	Integración de los componentes	3 días	jue 21/05/15	lun 25/05/15	15;16	Programador

Figura 6.3 Actividades fase implementación

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Fase Análisis**	**26 días**	**lun 16/02/15**	**lun 23/03/15**		
8	**Fase Diseño**	**20 días**	**mar 24/03/15**	**lun 20/04/15**	**1**	
14	**Fase Implementación**	**25 días**	**mar 21/04/15**	**lun 25/05/15**	**8**	
18	**Fase de pruebas**	**8 días**	**mar 26/05/15**	**jue 04/06/15**	**14**	
19	Pruebas funcionales	3 días	mar 26/05/15	jue 28/05/15		Evaluador
20	Pruebas de rendimiento	5 días	vie 29/05/15	jue 04/06/15	19	Evaluador

Figura 6.4 Actividades fase pruebas

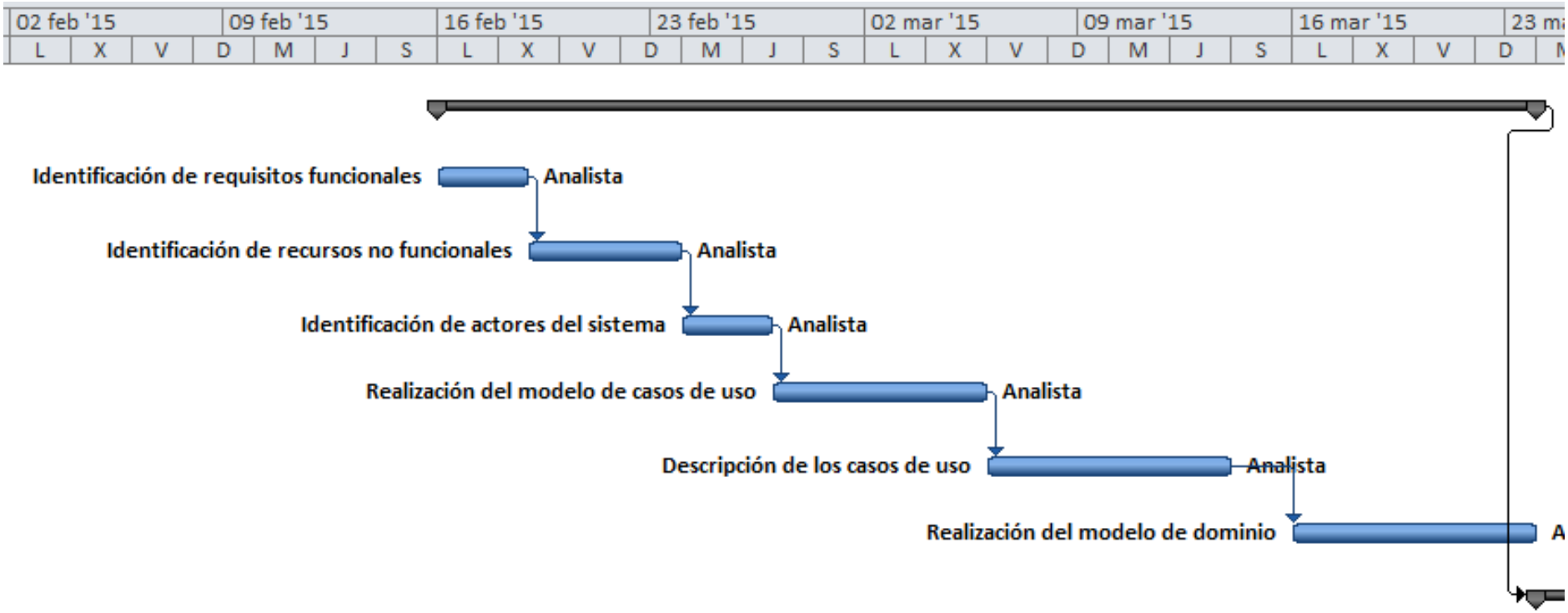


Figura 6.5 Diagrama de Gantt fase análisis

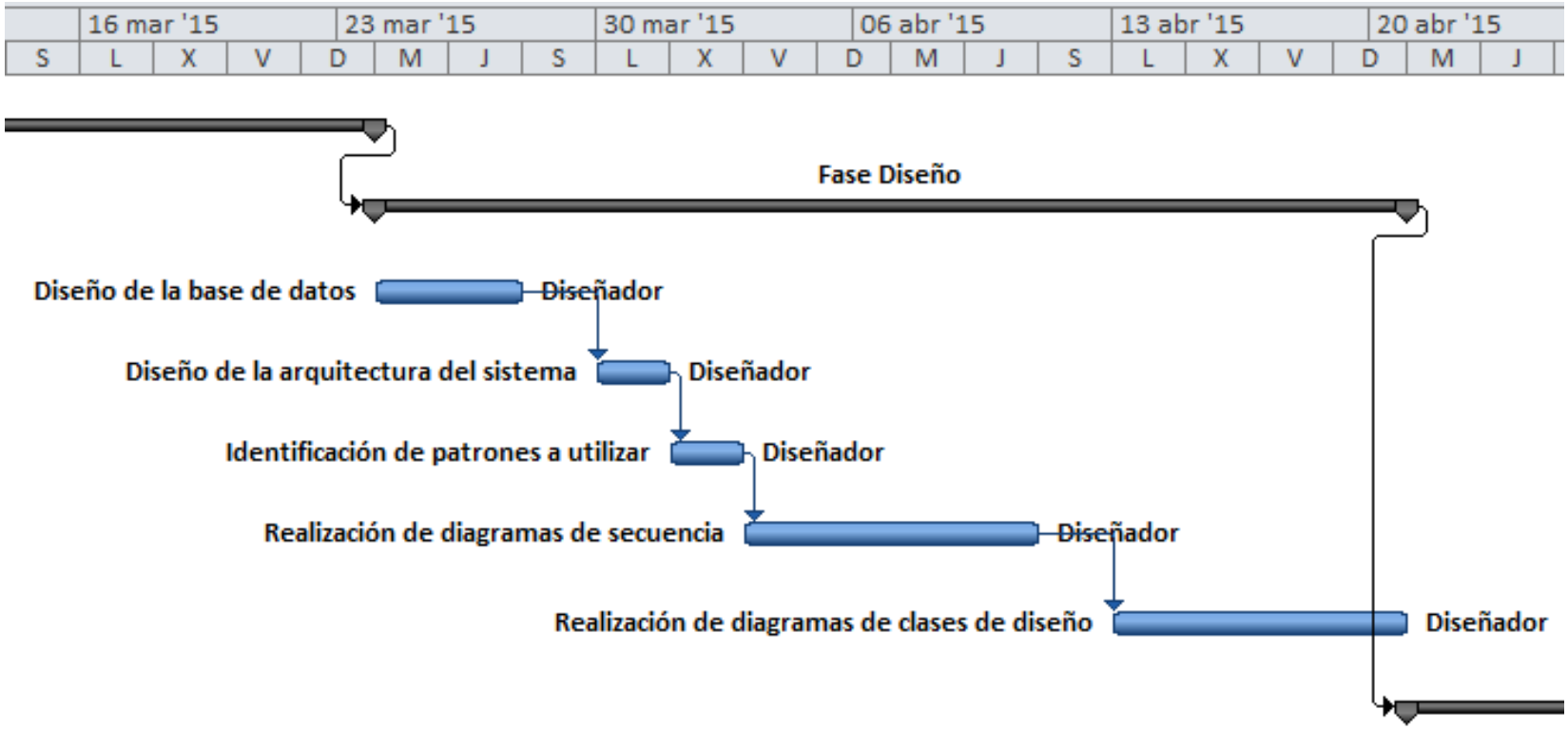


Figura 6.6 Diagrama de Gantt fase diseño

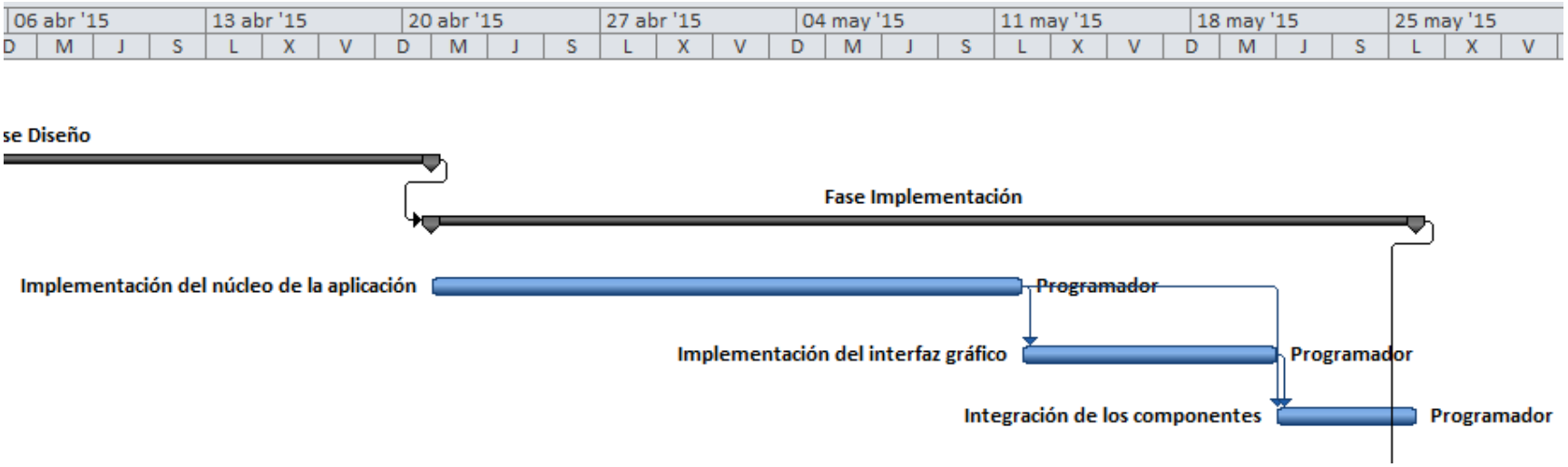


Figura 6.7 Diagrama de Gantt fase implementación

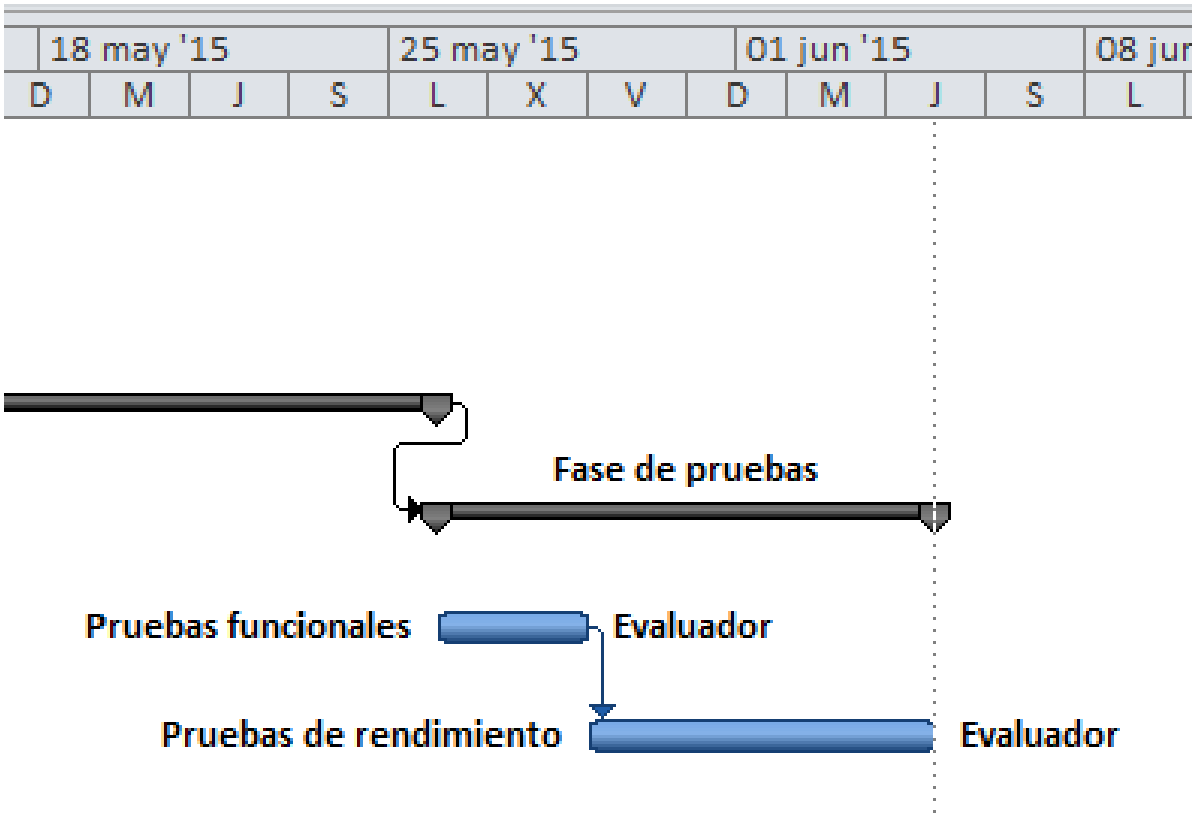


Figura 6.8 Diagrama de Gantt fase pruebas

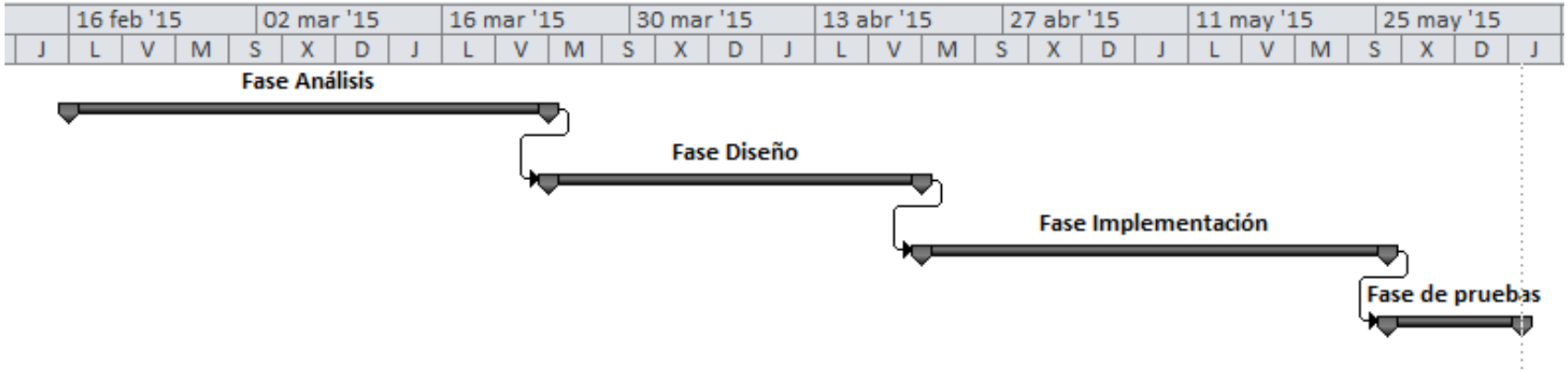


Figura 6.9 Diagrama de Gantt completo

Capítulo 7. Plan de gestión de riesgos

7.1 Introducción al plan de gestión de riesgos

El plan de gestión de riesgos está concebido para guiar a identificar, analizar y gestionar los riesgos que se puedan producir durante la etapa de desarrollo del sistema FLL+. Emplearemos una estrategia proactiva, integrada y sistémica, de tal forma que se identifiquen los riesgos potenciales teniendo en cuenta su probabilidad de ocurrencia, así como su posible impacto. Estableceremos políticas, sistemáticas y disciplinadas, de prioridad para los diferentes riesgos, que junto con los planes de acción correspondientes, controlará cualquier situación posible en caso de ocurrencia.

7.2 Identificación de los riesgos

Para realizar la identificación de riesgos, tanto el tutor del proyecto como yo nos reuniremos para realizar una pequeña tormenta de ideas de forma abierta para así tener una idea más global de los posibles riesgos a los que nos podremos enfrentar.

Después de una primera toma de contacto, y tras analizar y estudiar las primeras ideas iniciales, realizamos una segunda tormenta de ideas con el objetivo de ampliar, concretar y profundizar en los riesgos potenciales en función de las diferentes áreas, obteniendo así una nueva lista con los posibles riesgos.

En la figura 7.1 podemos ver una lista de los riesgos que hemos identificado.

Identificador	Descripción	Tipo de riesgo
R01	Interpretación errónea de los requisitos	De proyecto
R02	Calidad inadecuada	De producto
R03	Planificación optimista	De proyecto
R04	Diseño inadecuado	De producto
R05	Falta de experiencia del alumno	De proyecto
R06	Cambio del responsable del proyecto	De proyecto
R07	Problemas de comunicación	De proyecto
R08	Baja productividad	De proyecto
R09	Realización de pruebas no adecuada	De producto
R010	Falta de seguimiento	De proceso

Figura 7.1 Riesgos identificados

7.3 Análisis de riesgos

Teniendo la lista de riesgos identificados, definiremos un nivel de probabilidad de ocurrencia. El impacto que supondrá en el alcance del objetivo y el coste de tiempo, dado que es un proyecto universitario, no supone coste económico.

En la figura 7.2 podemos observar para cada riesgo, su coste, probabilidad de ocurrencia y el impacto que tendría en caso de que sucediese.

Identificador	Descripción	Probabilidad	Impacto	Coste
R01	Interpretación errónea de los requisitos	Muy alto	Muy alto	Elevado
R02	Calidad inadecuada	Alto	Muy alto	Elevado
R03	Planificación optimista	Medio	Alto	Medio
R04	Diseño inadecuado	Bajo	Alto	Medio
R05	Falta de experiencia del alumno	Bajo	Bajo	Corto
R06	Cambio del responsable del proyecto	Medio	Medio	Medio
R07	Problemas de comunicación	Bajo	Medio	Corto
R08	Baja productividad	Bajo	Bajo	Corto
R09	Realización de pruebas no adecuada	Medio	Alto	Corto
R019	Falta de seguimiento	Alto	Alto	Medio

Figura 7.2 Detalles riesgos identificados

Estableceremos una clasificación basada en la prioridad de los riesgos. La tabla, la hemos elaborado basándonos en una matriz de probabilidad-impacto, asignándose mayor prioridad a los riesgos que con más frecuencia ocurren y más impacto suponen.

En la figura 7.3 vemos la relación entre los riesgos y esta prioridad.

Identificador	Descripción	Prioridad
R01	Interpretación errónea de los requisitos	Muy alto
R02	Calidad inadecuada	Alto
R03	Planificación optimista	Medio
R04	Diseño inadecuado	Medio
R05	Falta de experiencia del alumno	Bajo
R06	Cambio del responsable del proyecto	Medio
R07	Problemas de comunicación	Bajo
R08	Baja productividad	Bajo
R09	Realización de pruebas no adecuada	Medio
R010	Falta de seguimiento	Alto

Figura 7.3 Priorización de riesgos

En la figura 7.4 podemos ver la matriz de probabilidad-impacto con su correspondiente agrupación de riesgos.

		Probabilidad			
		Muy Alto	Alto	Medio	Bajo
Impacto	Muy Alto	R01	R02	R03	R04
	Alto		R10	R09	
	Medio			R06	R07
	Bajo				R05; R06; R08

Figura 7.4 Matriz probabilidad-impacto

Según el nivel de prioridad del riesgo seguiremos un protocolo de actuación que se resume la figura 7.5.

Prioridad	Criterio	Acción
Muy alto	Riesgo crítico	Acción inmediata
Alto	Riesgo significativo y probable	Iniciar procedimientos
Medio	Riesgo significativo y menos probable	Realizar una revisión
Baja	Riesgo poco probable	Volver a valorar

Figura 7.5 Acciones a realizar en función del tipo de riesgo

7.4 Planificación de respuesta a los riesgos

Para los riesgos con prioridad alta y muy alta, hemos hecho planes de respuesta específicos para cada uno de ellos. En la figura 7.6 podemos verlo.

Identificador	Plan de mitigación	Plan de contingencia
R01	Reuniones con el tutor para confirmar la aceptación de los requisitos	Revisar requisitos afectados, código y documentación de los mismos
R02	Establecer un plan de calidad, siguiendo el estándar ISO 9000 y 9003	Hacer revisiones frecuentes con el tutor del proyecto
R010	Revisión continua y periódica del estado del proyecto y así detectar sus posibles anomalías	Realización de nuevas planificaciones. Reunión con el tutor del proyecto para recordar el cumplimiento del seguimiento

Figura 7.6 Planes de respuesta

7.1 Seguimiento y control de riesgos

El seguimiento adecuado y el control efectivo de los riegos permiten detectar de forma temprana los riesgos y ayudan a la toma de decisiones efectivas, por lo que adoptaremos las siguientes medidas:

- Revisar el estado de los riesgos entre el tutor y el alumno en reuniones periódicas.

- Revisar el estado de los riesgos en los hitos principales del proyecto

Tras cada hito se revisara y actualizara el documento, manteniéndose activo a lo largo de todo el proyecto.

Bloque III. Análisis

Capítulo 8. Introducción al análisis

8.1 Prólogo

Este bloque presenta el trabajo realizado en la fase de análisis del sistema. En él, definiremos con la mayor claridad posible que es lo que debe hacer el sistema que vamos a construir, a través de la elaboración de requisitos, tanto funcionales como no funcionales, diagramas de casos de uso, clases y secuencia, que mostrarán cual es el comportamiento que debe tener el sistema para los distintos eventos que ocurran en él.

8.2 Propósito general del sistema

El propósito general del sistema es construir un producto software que sea capaz de automatizar las tareas propias de un evento FLL y ayudar a la organización de eventos de este tipo.

No es necesario que los usuarios finales de la aplicación tengan unos conocimientos avanzados en la utilización de sistemas informáticos, pues las interfaces se crearán siguiendo un principio básico, que es la simplicidad, lo cual no implica que estas interfaces no sean potentes y puedan ayudar al usuario a realizar tareas complejas.

Además, en el bloque VI se explica de forma precisa como utilizar cada una de las diferentes funcionalidades del sistema FLL+ para que no quede duda alguna, o al menos sea la menor posible, a la hora de que los usuarios interactúen con la aplicación.

8.3 Sistema actual

Como ya se mencionó en capítulos anteriores, hasta el momento, la gestión y administración de los eventos FLL se hacía en muchos casos de forma manual, lo que conlleva la realización de tareas tediosas y repetitivas. Nuestro objetivo será eliminar esos obstáculos y proporcionar un software que apoye a la organización de un evento FLL en lo que se refiere al tratamiento automático de los datos.

8.4 Sistema propuesto

8.4.1 Visión general

Para definir de una forma fácilmente entendible la visión general del sistema propuesto, procuraré describirlo en pocas líneas dando un punto de vista desde un nivel de abstracción lo más alto posible.

El sistema deberá permitir a los árbitros arbitrar una partida a través de un interfaz sencillo, a la vez que se envían estos datos de forma automática a un servidor que irá publicando los datos de las partidas y clasificaciones en tiempo real al público interesado en el evento.

A su vez, permitirá a una persona encargada de controlar el arbitraje del evento, que denominaremos jefe de árbitros, a gestionar los datos, introduciéndolos, eliminándolos o modificándolos. También permitirá al jefe de árbitros planificar un evento FLL y poder gestionar el transcurso normal de un evento de este tipo.

Ahora llega el momento de concretar esta visión general. Para ello, continuaremos definiendo los diferentes actores que interactuarán con el sistema para después pasar a describir los requisitos funcionales y no funcionales en los siguientes apartados.

Capítulo 9. Requisitos

9.1 Requisitos funcionales

Los requisitos funcionales que debe tener el sistema se enumeran y explican a continuación.

Requisito 1

El sistema permitirá introducir al actor "árbitro" la puntuación obtenida en una prueba de una partida de un equipo concreto.

Requisito 2

El sistema permitirá visualizar al actor "árbitro" las partidas abiertas para arbitraje.

Requisito 3

El sistema permitirá identificarse a cada uno de los diferentes actores.

Requisito 4

El sistema deberá mostrar a cada actor las tareas que éste puede realizar en función de sus permisos.

Requisito 5

El sistema deberá permitir al actor "jefe de árbitros" planificar una partida para una hora determinada.

Requisito 6

El sistema permitirá al actor "jefe de árbitros" marcar una partida como "Jugada".

Requisito 7

El sistema permitirá al actor "jefe de árbitros" marcar una partida como "No jugada".

Requisito 8

El sistema permitirá al actor "jefe de árbitros" marcar una partida como "En curso".

Requisito 10

El sistema permitirá introducir equipos nuevos al actor "jefe de árbitros".

Requisito 11

El sistema permitirá eliminar equipos al actor "jefe de árbitros".

Requisito 12

El sistema permitirá eliminar partidas al actor "jefe de árbitros".

Requisito 13

El sistema permitirá generar un informe de una partida al actor "jefe de árbitros"

Requisito 14

El sistema permitirá modificar la puntuación de las partidas en cualquier momento al "jefe de árbitros"

Requisito 15

El sistema permitirá consultar la documentación oficial del torneo FLL al actor "jefe de árbitros".

Requisito 16

El sistema permitirá al actor "jefe de árbitros" iniciar la cuenta atrás de una partida

Requisito 17

El sistema permitirá al actor "jefe de árbitros" detener la cuenta atrás de una partida.

Requisito 18

El sistema permitirá visualizar al "jefe de árbitros" qué partida está siendo arbitrada por cada árbitro

Requisito 19

El sistema permitirá al actor "espectador" consultar la clasificación.

Requisito 20

El sistema permitirá al actor "espectador" consultar el estado de la cuenta atrás de una partida.

Requisito 21

El sistema permitirá consultar al actor "espectador" la puntuación de una partida que esté en curso.

9.2 Requisitos no funcionales

A continuación se describen los requisitos no funcionales que se han identificado, organizados por tipo.

9.2.1 Confiabilidad y disponibilidad

Requisito 22

El sistema deberá soportar del orden de cientos de usuarios conectados al sistema al mismo tiempo sin entrar en modo degradado.

Requisito 23

El sistema deberá mostrar los datos en tiempo real sin necesidad de que los actores tengan que solicitar la información de forma explícita.

9.2.2 Persistencia

Requisito 24

El sistema deberá almacenar de manera persistente los datos del evento en una base de datos

9.2.3 Usabilidad

Requisito 25

El sistema debe ser fácil de usar por una persona con conocimientos básicos en el manejo de dispositivos informáticos.

9.1 Seguridad

Requisito 26

El sistema solo permitirá hacer a cada actor las funcionalidades asociadas a él.

Requisito 27

El sistema solo permitirá modificar los datos a los usuarios con permisos para ello.

Requisito 28

El sistema deberá asegurar la integridad de los datos almacenados.

Capítulo 10. Casos de uso

10.1 Descripción general de los actores

Hemos identificado 3 tipos de actores que interactuarán con el sistema. A continuación pasamos a describirlos.

Arbitro El principal interés del árbitro es poder arbitrar una partida, para ello, irá introduciendo en el sistema los datos relativos a qué pruebas han sido realizadas por cada equipo y de qué forma.

Jefe de árbitros Los principales objetivos o intereses de este actor son planificar las partidas que se van a jugar, llevar un control de qué árbitros arbitran qué partidas y la obtención de informes sobre las puntuaciones obtenidas por cada uno de los equipos.

Espectador Este actor está interesado en visualizar las puntuaciones que vayan obteniendo los equipos, así como conocer cuál es la clasificación y el estado de las partidas que se están disputando.

10.2 Diagrama de casos de uso

En este apartado vamos a presentar el diagrama de casos de uso, en el cuál, mostramos como se relacionan los actores con los diferentes casos de uso presentes en el sistema. Ver figura 10.1.

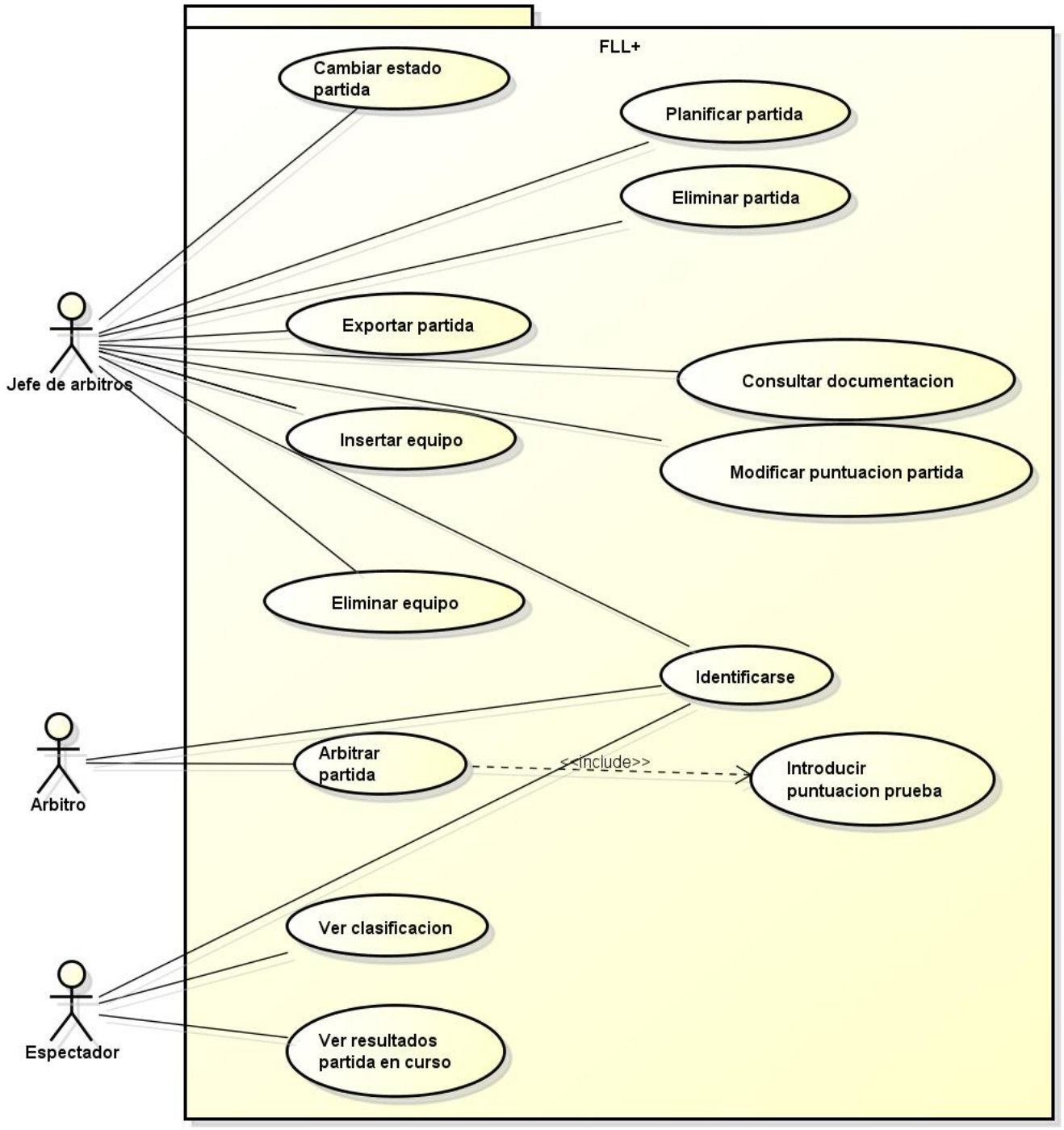


Figura 10.1 Diagrama de casos de uso

10.1 Especificación de casos de uso

A continuación pasamos a detallar cada uno de los casos de uso en cada una de las siguientes figuras.

UC-0001	Cambiar estado partida	
Descripción	El actor jefe de árbitros desea modificar el estado actual de una partida	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver el planificador de partidas
	2	El sistema muestra una lista de acciones posibles y el conjunto de las partidas planificadas
	3	El actor usuario selecciona un nuevo estado para una partida concreta
	4	El sistema registra el nuevo estado de la partida y el caso de uso finaliza
Postcondición	El actor jefe de árbitros ha modificado el estado de una partida	
Excepciones	**Paso**	**Acción**
	2	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.

Figura 10.2 Caso de uso "Cambiar estado partida"

UC-0002	Planificar partida	
Descripción	El actor jefe de árbitros va a programar una partida para un equipo concreto y una hora concreta.	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver el planificador de partidas
	2	El sistema muestra una lista de acciones posibles y el conjunto de las partidas planificadas
	3	El actor jefe de árbitros selecciona la opción planificar partida
	4	El sistema solicita al jefe de árbitros que introduzca el equipo, ronda y hora deseadas
	5	El actor jefe de árbitros introduce el equipo, ronda y hora a la que desea planificar la partida
	6	El sistema almacena los datos y el caso de uso finaliza.
Postcondición	Se ha planificado una nueva partida	
Excepciones	**Paso**	**Acción**
	2,4	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.
	5	Si el actor jefe de árbitros introduce los datos de una partida ya planificada, se informa de ello y el caso de uso queda sin efecto

Figura 10.3 Caso de uso "Planificar partida"

UC-0003	Eliminar partida	
Descripción	El actor jefe de árbitros desea eliminar una partida	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver el planificador de partidas
	2	El sistema muestra una lista de acciones posibles y el conjunto de las partidas planificadas
	3	El actor jefe de árbitros selecciona la opción eliminar de una partida concreta
	4	El sistema registra los cambios y el caso de uso finaliza
Postcondición	Se ha eliminado una partida	
Excepciones	**Paso**	**Acción**
	2	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.

Figura 10.4 Caso de uso "Eliminar partida"

UC-0004	Exportar partida	
Descripción	El actor jefe de árbitros desea obtener un informe de una partida	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver el planificador de partidas
	2	El sistema muestra una lista de acciones posibles y el conjunto de las partidas planificadas
	3	El actor jefe de árbitros selecciona la opción exportar partida de una partida concreta
	4	El sistema genera un informe de con el resumen de las puntuaciones obtenidas en la partida seleccionada
Postcondición	Se ha planificado una nueva partida	
Excepciones	**Paso**	**Acción**
	2	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.

Figura 10.5 Caso de uso "Exportar partida"

UC-0005	Consultar documentación	
Descripción	El actor jefe de árbitros desea ver la documentación de la FLL	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver la documentación de la FLL
	2	El sistema muestra una lista con los diferentes ficheros de documentación de la FLL
	3	El actor jefe de árbitros selecciona un fichero de documentación
	4	El sistema muestra al jefe de árbitro el fichero seleccionado
Postcondición	Se ha planificado una nueva partida	
Excepciones	**Paso**	**Acción**
	2	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.

Figura 10.6 Caso de uso "Consultar documentación"

UC-0006	Insertar equipo	
Descripción	El actor jefe de árbitros desea registrar un nuevo equipo en el sistema.	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver el gestor de equipos dentro del panel de administración
	2	El sistema muestra una lista de acciones posibles y el conjunto de los equipos existentes.
	3	El actor jefe de árbitros selecciona la opción insertar equipo
	4	El sistema solicita al jefe de árbitros que introduzca el nombre del equipo y una descripción del mismo
	5	El actor jefe de árbitros introduce el nombre del equipo y la descripción.
	6	El sistema almacena los datos y el caso de uso finaliza.
Postcondición	Se ha insertado un nuevo equipo en el sistema	
Excepciones	**Paso**	**Acción**
	2,4	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.

Figura 10.7 Caso de uso "Insertar equipo"

UC-0007	Eliminar equipo	
Descripción	El actor jefe de árbitros eliminar un equipo del sistema.	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver el gestor de equipos dentro del panel de administración
	2	El sistema muestra una lista de acciones posibles y el conjunto de los equipos existentes.
	3	El actor jefe de árbitros selecciona la opción eliminar de un equipo concreto
	4	El sistema elimina el equipo y todos los datos asociados con el
Postcondición	Se ha eliminado un equipo y todos los datos relacionados con él del sistema	
Excepciones	**Paso**	**Acción**
	2	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.

Figura 10.8 Caso de uso "Eliminar equipo"

UC-0008	Modificar puntuación de una partida	
Descripción	El actor jefe de árbitros desea modificar una puntuación de una partida ya sea porque el árbitro ha cometido un error o similar.	
Actores implicados	Jefe de árbitros	
Precondición	El actor jefe de árbitros está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor jefe de árbitros solicita ver el planificador de partidas
	2	El sistema muestra una lista de acciones posibles y el conjunto de las partidas planificadas
	3	El actor jefe de árbitros selecciona la opción modificar partida de una partida concreta
	4	El sistema solicita al jefe de árbitros que introduzca las nuevas puntuaciones de cada prueba de la partida
	5	El actor jefe de árbitros introduce las nuevas puntuaciones
	6	El sistema almacena las nuevas puntuaciones y el caso de uso finaliza.
Postcondición	Se ha modificado la puntuación de una partida	
Excepciones	**Paso**	**Acción**
	2,4	Si desea salir del proceso, el actor jefe de árbitros cancela la operación y a continuación el caso de uso queda sin efecto.

Figura 10.9 Caso de uso "Modificar puntuación de una partida"

UC-0009	Identificarse	
Descripción	Un usuario desea identificarse frente al sistema para realizar alguna acción	
Actores implicados	Jefe de árbitros, árbitros y espectadores	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El actor solicita identificarse frente al sistema
	2	El sistema pide al actor que introduzca su nombre de usuario y contraseña
	3	El actor introduce sus credenciales
	4	El sistema da acceso al sistema en función de los permisos asociados a la cuenta introducida.
Postcondición	Un nuevo actor se ha identificado frente al sistema	
Excepciones	**Paso**	**Acción**
	2	Si desea salir del proceso, el actor cancela la operación y a continuación el caso de uso queda sin efecto.
	4	Si los datos introducidos no son correctos, el sistema informa al usuario

		y el caso de uso queda sin efecto.

Figura 10.10 Caso de uso "Identificarse"

UC-0010	Ver clasificación	
Descripción	El actor espectador desea ver la clasificación	
Actores implicados	Espectador	
Precondición	El espectador está identificado frente al sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor espectador solicita ver el marcador
	2	El sistema muestra al usuario entre otros datos, la clasificación, que será refrescada automáticamente sin que el actor espectador lo tenga que volver a solicitar explícitamente.
Postcondición	El sistema ha consultado los datos y los ha devuelto al actor espectador	
Excepciones	Ninguna	

Figura 10.11 Caso de uso "Ver clasificación"

UC-0011	Ver resultados partida en curso	
Descripción	El actor espectador desea los resultados de la partida que se está jugando en ese preciso instante	
Actores implicados	Espectador	
Precondición	El espectador está identificado frente al sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor espectador solicita ver el marcador
	2	El sistema muestra al usuario entre otros datos, el resultado de la partida en curso, que será refrescada automáticamente sin que el actor espectador lo tenga que volver a solicitar explícitamente.
Postcondición	El sistema ha consultado los datos de la partida en curso y los ha devuelto al actor espectador	
Excepciones	Ninguna	

Figura 10.12 Caso de uso "Ver resultados partida en curso"

UC-0012	Arbitrar partida	
Descripción	El actor árbitro desea arbitrar una partida	
Actores implicados	Arbitro	
Precondición	El árbitro está identificado frente al sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor árbitro solicita arbitrar una partida
	2	El sistema muestra al árbitro las partidas marcadas como "En curso" que no están siendo arbitradas por otros árbitros
	3	El actor árbitro selecciona una de las partidas.
	4	El sistema muestra una lista con todas las pruebas existentes
	5	Mientras dure la partida, el caso de uso incluye el caso de uso "Introducir puntuación prueba)
Postcondición	El sistema ha almacenado los datos de la partida introducidos por el actor árbitro	
Excepciones	Ninguna	

Figura 10.13 Caso de uso "Arbitrar partida"

UC-0013	Introducir puntuación prueba	
Descripción	El actor árbitro desea modificar la puntuación de una prueba debido a que el equipo ha realizado interactuado con esa prueba en el tablero de juego	
Actores implicados	Arbitro	
Precondición	El árbitro está identificado frente al sistema	
Secuencia normal	**Paso**	**Acción**
	1	El actor árbitro selecciona una prueba
	2	El sistema actualiza la prueba a su estado siguiente y actualiza la puntuación de la partida.

Postcondición	El sistema ha almacenado los datos de la partida introducidos por el actor árbitro
Excepciones	Ninguna

Figura 10.14 Caso de uso "Introducir puntuación prueba"

Capítulo 11.
Modelo de dominio

11.1 Diagrama de clases de dominio

En esta sección vamos a presentar las clases que hemos descubierto en una primera aproximación para conocer que debe realizar la aplicación. Una vez hayamos mostrado el diagrama, explicaremos brevemente que clases existen y como se relacionan entre ellas, a la vez que definimos las responsabilidades de cada una de ellas.

El diagrama de clases de dominio podemos verlo en la figura 11.1.

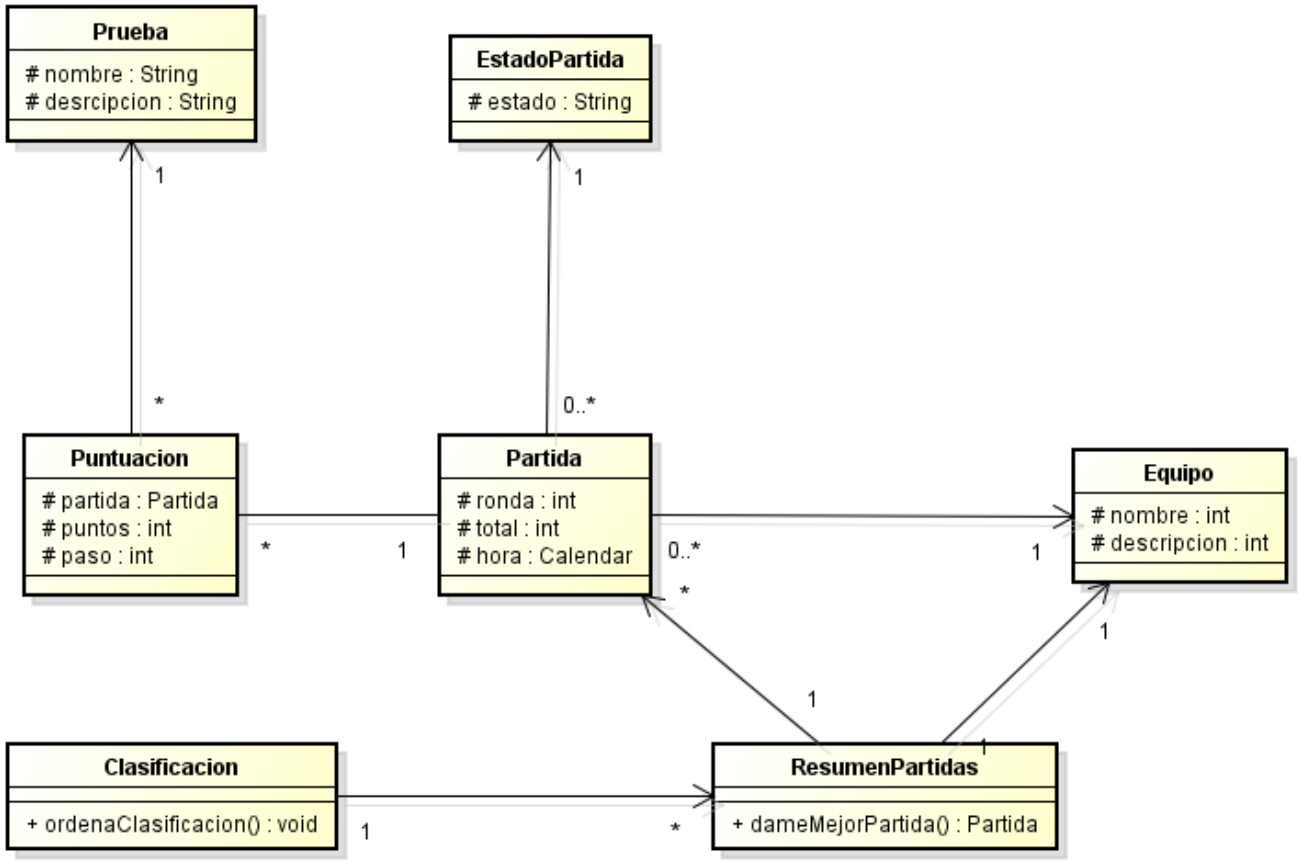


Figura 11.1 Diagrama de clases de dominio

11.1.1 Partida

Descripción La clase "Partida" modela el concepto de partida, es decir, una ronda jugada por un equipo a una hora determinada.

Relaciones Cada partida está compuesta por varias puntuaciones, se ha jugado por 1 único equipo y está en un solo estado aunque este estado pueda ir variando a lo largo del tiempo.

11.1.2 Equipo

Descripción La clase "Equipo" almacena un nombre y una descripción y se encarga de modelar el concepto de equipo de un torneo FLL.

Relaciones En este diagrama la clase equipo no conoce a ninguna otra, aunque otras clases sí que la conocen a ella.

11.1.3 Puntuación

Descripción La clase "Puntuacion" almacena los puntos obtenidos en una determinada prueba, así como el paso en el que se encuentra la prueba. Para entender mejor el concepto del atributo paso pondremos un ejemplo: Si en una partida, el equipo no ha interactuado con la prueba a la que hace referencia la clase "Puntuacion", el atributo paso valdrá 0.

Relaciones Cada objeto puntuación pertenece a una partida y hace referencia a una única prueba

11.1.4 Prueba

Descripción La clase "Prueba" almacena el nombre de la prueba y la descripción de la misma. Se encarga de modelar el concepto de prueba de una partida FLL.

Relaciones La clase "Prueba" no conoce a ninguna otra clase en tiempo de ejecución.

11.1.5 EstadoPartida

Descripción Esta clase almacena el nombre del estado, que puede ser "No Jugada", "Jugada" o "En curso".

Relaciones La clase "EstadoPartida" no conoce a ninguna otra clase, sin embargo la clase "Partida" está relacionada con ella.

11.1.6 ResumenPartidas

Descripción Esta clase almacena todas las partidas jugadas por un determinado equipo a lo largo del desarrollo del evento FLL. Almacena en sus atributos el equipo y las partidas que ha jugado ese equipo, por tanto, puede conocer cuál ha sido la partida con mayor puntuación de cada uno de los equipos.

Relaciones "ResumenPartidas" contiene una colección de elementos partidas y conoce a la clase "Equipo" que las jugó.

11.1 Clasificacion

Descripción La clase Clasificación almacena objetos del tipo “ResumenPartidas” en sus atributos. Se encarga de modelar el concepto de clasificación dentro de un evento FLL. Además puede ordenar la clasificación, ya que conoce todos los objetos “ResumenPartidas” que pueden devolver la mejor partida de cada equipo.

Relaciones Un objeto “Clasificación” se relaciona con varios elementos del tipo “ResumenPartidas”, por ello puede generar la lista ordenada de la clasificación de un evento FLL.

Bloque IV. Diseño

Capítulo 12. Diseño de la base de datos

12.1 Introducción

En este capítulo vamos a dar una visión detallada de la base de datos que hemos utilizado para mantener la persistencia de los datos. Comenzaremos explicando el porqué de las decisiones de diseño respecto de la estructura de la base de datos y del motor de almacenamiento utilizado. Continuaremos dando una visión detallada de los campos que contiene cada tabla, y por último, veremos la forma de implementación de la base de datos.

En el diseño de la base de datos se han tenido en cuenta los principios fundamentales del diseño de bases de datos, para evitar en la medida de lo posible los problemas más comunes al trabajar con bases de datos, tales como, asegurar la no redundancia de datos o facilitar el control de las modificaciones para evitar las violaciones de las restricciones de seguridad.

12.2 Decisiones de diseño de la base de datos

Las decisiones que se han tomado para diseñar la base de datos se basan en dos principios básicos:

Rapidez Dado que estamos diseñando una aplicación que muestra datos en tiempo real, la rapidez es una característica básica que debemos tener en cuenta, para que no sucedan hechos como, mostrar a los usuarios el estado de una partida o su puntuación con un retardo no razonable.

Seguridad El sistema FLL+ almacena puntuaciones de partidas que han conseguido ciertos equipos. Estos equipos, han dedicado en muchos casos gran cantidad de horas para desarrollar su robot Lego, por tanto es importante que las puntuaciones finales sean justas y por tanto mantengan su integridad. No sería admisible que una persona sin permisos pudiese modificar la información almacenada en el sistema.

Respecto al primer principio mencionado (la rapidez) tenemos que dedicar varias líneas más.

MySQL nos permite utilizar diferentes motores de almacenamiento, entre los que se encuentran dos de los más utilizados: innoDB y MyISAM.

El motor de almacenamiento (storage-engine) se encarga de almacenar, manejar y recuperar información de una tabla. La elección de uno u otro dependerá del escenario en el que nos encontremos. Si necesitamos transacciones, claves foráneas y bloqueos, la mejor opción es escoger InnoDB. Por el contrario, escogeremos MyISAM en aquellos casos en los que predominen las consultas SELECT a la base de datos.

Como ya hemos mencionado, es de vital importancia la rapidez en el sistema que estamos diseñando ya que se trata de un sistema que muestra información en tiempo real a un número considerable de usuarios conectados al mismo tiempo.

La mayor parte de las consultas son de tipo select, las cuales son utilizadas para mostrar al espectador los resultados que se vayan cargando en la base de datos. Los updates quedan en segundo plano, ya que por cada update que realice un árbitro al actualizar una puntuación de cierta partida, se consultará cientos de veces por parte de los espectadores para mostrar los resultados en tiempo real.

En resumen, estamos frente a un sistema que realizará muchas más consultas de tipo select que updates, inserts o deletes, por tanto, todas las tablas que se utilicen para mostrar los datos de las puntuaciones al espectador estarán manejadas por un motor de almacenamiento MyISAM.

Sin embargo, aquellas tablas que no se utilicen para mostrar la información en el marcador serán InnoDB.

12.3 Diagrama entidad-relación

A continuación mostramos el diagrama que muestra la relación entre las tablas de la base de datos. Como podemos ver todas las tablas que almacenan datos que deben ser consultados de forma muy frecuente no tienen relación con otras ya que utilizan MyISAM, sin embargo sí que existe una relación lógica para relacionar los datos de unas tablas MyISAM con otras a la hora de realizar las consultas desde la aplicación

Ver figura 12.1.

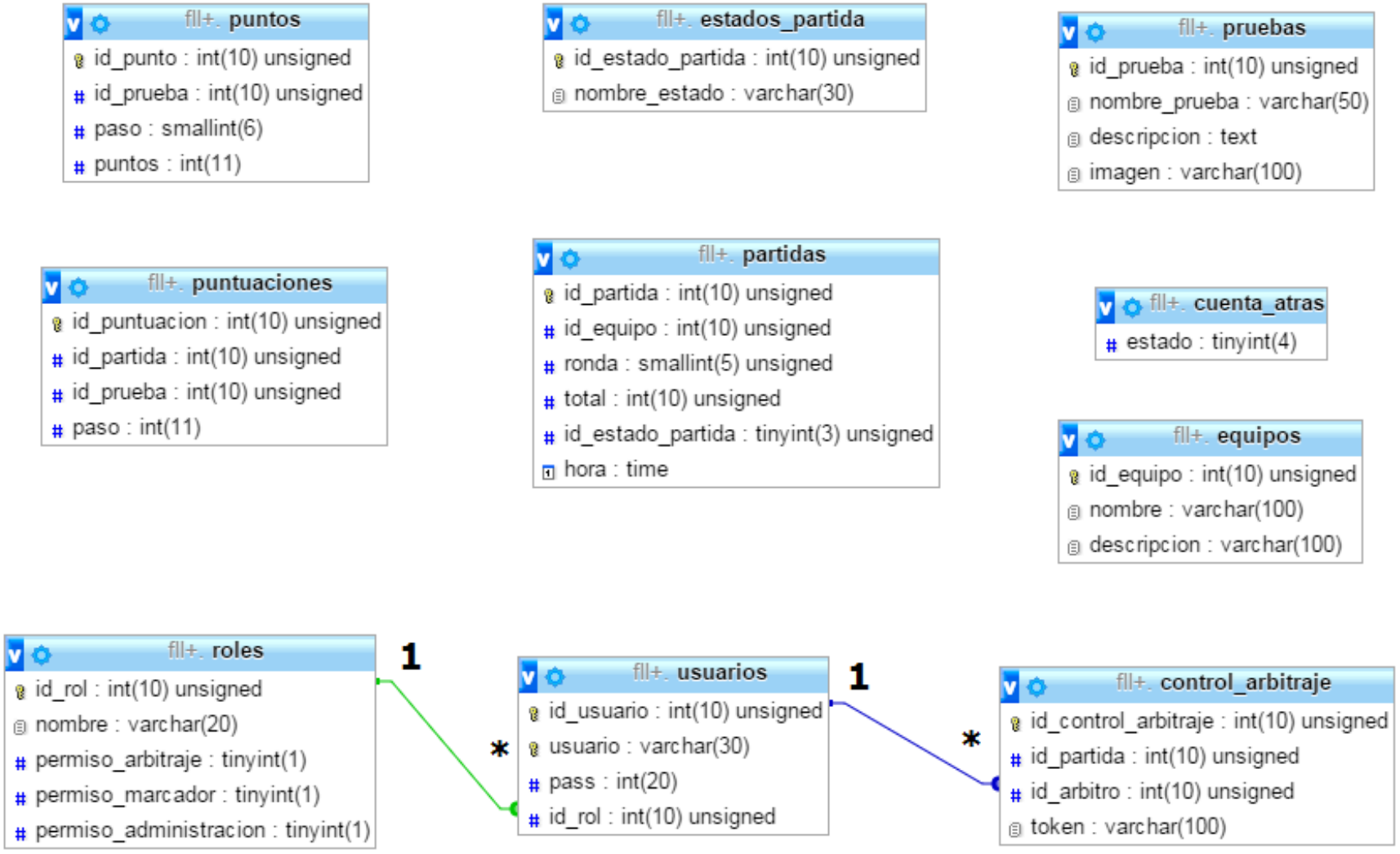


Figura 12.1 Esquema base de datos

12.3.5 Roles

Figura 12.2 Tabla "roles"

Descripción En esta tabla se almacena los diferentes roles que existen en el sistema. Cada rol posee unos permisos que le permiten al usuario acceder a ciertas acciones del sistema.

Relaciones

Usuarios Cada rol puede estar relacionado con 0 o varios usuarios

Campos

Id_rol Define un identificador único a cada rol dado de alta en el sistema.

Nombre Hace referencia al nombre del rol.

Permiso_arbitraje Este campo almacena un 1 si el tipo de rol puede arbitrar partidas o 0 en caso contrario**.**

Permiso_marcador Este campo almacena un 1 si el tipo de rol tiene permisos para visualizar el marcador o 0 en caso contrario.

Permiso_adminsitracion Este campo almacena un 1 si el tipo de rol tiene permisos para acceder al panel de administración o 0 en caso contrario.

12.3.6 Control arbitraje

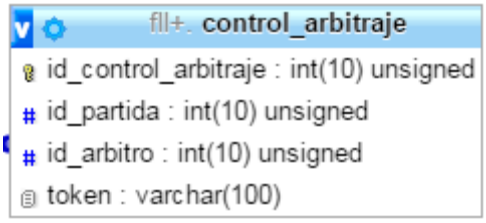


Figura 12.3 Tabla "control_arbitraje"

Descripción Esta tabla almacena un registro cada vez que un árbitro comienza a arbitrar, de tal forma que otro árbitro no pueda arbitrar la misma partida. También permite que el jefe de árbitros vea en cada momento que árbitros están arbitrando que partidas.

Relaciones

Usuarios Cada registro de esta tabla se relaciona con un único usuario, sin embargo varios usuarios pueden estar relacionados varios registros de esta tabla.

Campos

Id_control_arbitraje Define un identificador único para registro de esta tabla.

Id_partida Hace referencia a la partida que se está arbitrando.

Id_arbitro Hace referencia al árbitro que está arbitrando la partida.

12.3.7 Partidas

Figura 12.4 Tabla "partidas"

De aquí en adelante explicaremos las tablas MyISAM. Este tipo de tablas no permite la utilización de claves foráneas, sin embargo sí que definiremos las relaciones con otras tablas, ya que existe una relación lógica entre ellas.

Descripción Esta tabla almacena la información de las partidas que están planificadas por el jefe de árbitros en el sistema

Relaciones

Equipos Cada partida es jugada por un equipo.

Estados_partida Cada partida está en un estado ("No jugada", "Jugada", "En curso").

Campos

Id_partida Define un identificador único para cada partida.

Id_equipo Guarda la relación entre la partida y el equipo que la ha jugado o va a jugar.

Ronda Indica la a que ronda pertenece la partida.

Total Almacena la puntuación total obtenida en la partida.

Id_estado_partida Hace referencia al estado en el que está la partida.

Hora Es la hora a la que se ha planificado la partida.

12.3.8 Equipos

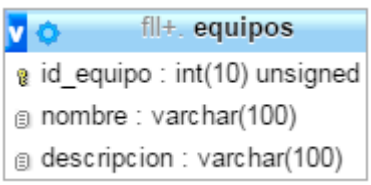


Figura 12.5 Tabla "equipos"

Descripción Almacena los equipos registrados en el sistema por parte del jefe de árbitros.

Relaciones

No tiene ninguna relación lógica con otras tablas.

Campos

Id_equipo Define un identificar único a cada equipo.

Nombre El nombre del equipo.

Descripción Una descripción del equipo.

12.3.9 Puntuaciones

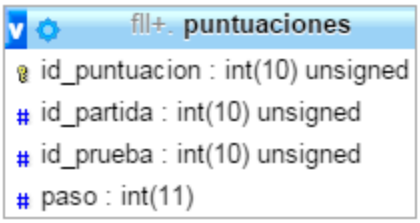


Figura 12.6 Tabla "puntuaciones"

Descripción Almacena las puntuaciones obtenidas en cada una de las pruebas de una partida.

Relaciones

Partidas Hace referencia a la partida para la cual está guardando la puntuación.

Pruebas Hace referencia a la prueba a la que hace referencia la puntuación.

Campos

Id_puntuacion Define un identificador para cada puntuación existente en el sistema.

Id_partida Almacena el id de la partida a la que hace referencia la puntuación.

Id_prueba Almacena el id de la prueba a la que hace referencia la puntuación.

Paso Almacena el paso de la puntuación. Por ejemplo, una prueba con la que el robot no haya interactuado tendrá un paso 0.

12.3.10 Puntos

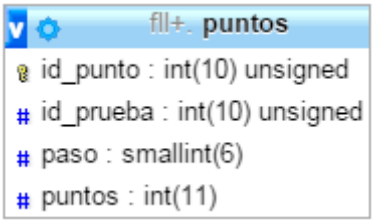


Figura 12.7 Tabla "puntos"

Descripción Almacena los puntos para cada paso de cada prueba, de esta forma, conseguimos que si el reglamento modifica el valor de cada prueba, se pueda configurar fácilmente para adaptar la aplicación al nuevo reglamento.

Relaciones

Pruebas La prueba a la que hacen referencia los puntos.

Campos

Id_punto Define un identificador único para cada registro.

Id_prueba Almacena el identificador de la prueba a la que hacen referencia los puntos.

Paso Guarda el paso para el que se almacenan los puntos.

Puntos Puntos obtenidos, para el paso definido en el campo "paso", en la prueba.

12.3.11 Estados_partida

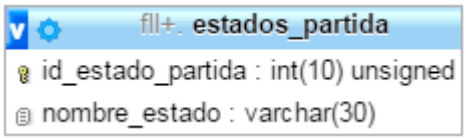


Figura 12.8 Tabla "estados_partida"

Descripción Almacena los posibles estados en el que puede estar una partida. Esta tabla podría haberse convertido en un campo de la tabla partida, pero de esta forma dejamos abierta la posibilidad de que en un futuro haya más estados posibles.

Relaciones

No existen relaciones lógicas

Campos

Id_estado_partida Define un identificador único para cada registro.

Nombre_estado Nombre del estado en el que puede estar la partida.

12.3.12 Pruebas

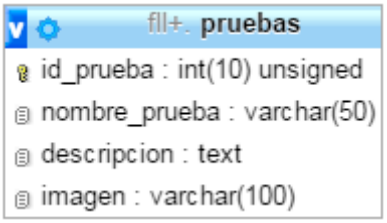


Figura 12.9 Tabla "pruebas"

Descripción Almacena los datos relativos a las pruebas que existen en el reglamento de la FLL.

Relaciones

No existen relaciones lógicas con otras tablas.

Campos

Id_prueba Define un identificador único para cada prueba.

Nombre_prueba Es el nombre de la prueba.

Descripción Muestra una pequeña descripción de la prueba.

12.1 Cuenta atrás

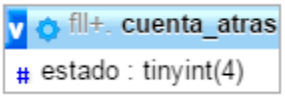


Figura 12.10 Tabla "cuenta-atrás"

Descripción Almacena el estado de la cuenta atrás.

Relaciones

No existen.

Campos

Estado Identifica a través de un número el estado de la cuenta atrás, el cual puede ser, "detenida", cuenta atrás de 10 segundos antes de que empiece la partida o cuenta atrás de 2 minutos y medio mientras transcurre la partida.

Capítulo 13. Diseño de la aplicación

13.1 Introducción

En este capítulo vamos a detallar el proceso de diseño de la arquitectura del sistema. En la primera parte del documento elaboraremos un diseño arquitectónico del sistema, incluyendo aspectos organizacionales, una subdivisión del sistema completo en subsistemas lógicos que ayudarán a comprender qué elementos físicos se necesitan para componer el sistema. Iremos aplicando patrones de diferentes referencias que ayudarán a diseñar el sistema de modo que cumpla con las condiciones de calidad de software de forma probada y documentada. Una vez hayamos identificado los subsistemas, haremos la realización de los casos de uso principales del sistema FLL+.

13.2 Descripción del sistema

13.2.1 Diseño arquitectónico

Para el desarrollo este apartado partimos desde un punto en el que se dispone de una serie de requisitos y restricciones que describen el sistema y que este debe desempeñar satisfactoriamente. En esta y en las sucesivas secciones crearemos un diseño arquitectónico del sistema FLL+ obteniendo una serie de subsistemas que compondrán el sistema completo, junto con las relaciones entre los mismos. El diseño debe ser realizado de forma que se aseguren las siguientes condiciones, a la vez que exista un equilibrio entre ellas:

Rendimiento Como ya se ha mencionado, nuestro reto es crear una herramienta software que sea capaz de transmitir datos en tiempo real a un número relativamente elevado de espectadores, a la vez que se facilita la tarea a la organización del evento FLL a la hora de realizar las acciones básicas en un evento de este tipo, por tanto, es especialmente importante que la herramienta ofrezca tiempos de respuesta razonables. No sería admisible mostrar a los distintos usuarios de la aplicación datos generados hace varios minutos.

Integridad La integridad de los datos es otra característica clave que debe presentar el sistema. Los equipos que participan en estos eventos han trabajado mucho para participar en esta competición, por tanto, los datos deben mantener su integridad en todo momento.

Disponibilidad La disponibilidad es otro punto clave, ya que el hecho de que el sistema deje de hacer lo que debe hacer puede producir retrasos en la celebración del evento FLL.

Confidencialidad La confidencialidad asegura que los datos sean únicamente accedidos o visualizados por los usuarios que tengan permiso para ello.

En definitiva, la aplicación deberá ser especialmente diseñada para obtener un buen rendimiento y debe ser segura. Disponibilidad, integridad y confidencialidad son los 3 principales elementos que componen el concepto de seguridad en informática.

13.2.2 Organización del sistema

Para la organización más elemental del sistema utilizaremos unos de los modelos más extendidos: el modelo cliente-servidor. Pensamos que es el modelo más adecuado para un sistema de estas características, en el que sus elementos se interrelacionan por medio de la red, así, este modelo tiene la ventaja de ser una arquitectura distribuida, es posible hacer una separación de los elementos diferenciados más sencilla, mantenible y segura. De esta forma ya se diferencian 2 grandes subsistemas: los clientes, y el servidor.

Intentaremos integrar el modelo cliente-servidor con un modelo de repositorio de datos, ya que nuestro sistema contará con varios elementos físicos diferentes que compartirán los mismos datos. En este modelo de repositorio de datos estarán todos los datos que deban ser almacenados en el sistema, como pueden ser los equipos, las pruebas o las puntuaciones de las partidas.

También aplicaremos un modelo de capas a todo el sistema separando la capa de persistencia, la capa de lógica de la aplicación y la capa de presentación. De esta forma, cada capa sólo puede comunicarse con sus capas adyacentes añadiéndose un nivel de protección extra al estar los datos en la capa más interna de la aplicación. Así tendremos un modelo cliente-servidor de tres capas con repositorio de datos. De esta forma, el servidor queda descompuesto en otros dos subsistemas, uno que se encargará de la lógica de los datos y el otro de la lógica de aplicación.

A continuación mostramos un diagrama que describe la arquitectura que proponemos en 3 capas claramente diferenciadas. Ver figura 13.1.

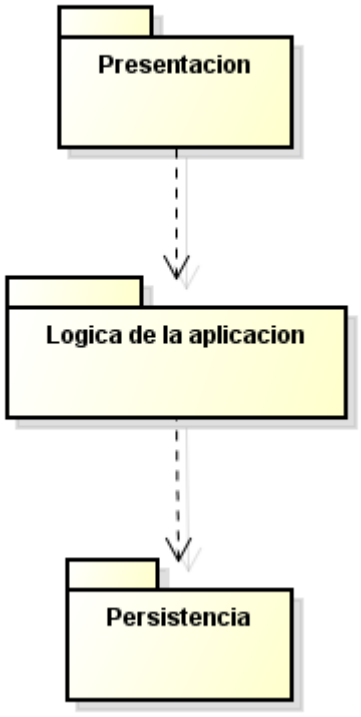


Figura 13.1 Detalle capas del sistema

Presentación Esta capa es la responsable de mostrar el interfaz gráfico de la aplicación, por tanto es la que interactúa directamente con los usuarios de FLL+. Esta capa depende de la lógica de la aplicación, por tanto, será a esta capa a la que realizará las peticiones provenientes de las solicitudes de los usuarios de FLL+.

Lógica de la aplicación En esta capa se encuentran las clases, las cuales, colaboran entre ellas para resolver un problema. Esta capa por tanto es la encargada de dar solución a los casos de uso que se han identificado previamente. La lógica de la aplicación depende de la capa de persistencia, pues necesita recuperar los datos almacenados para poder realizar su trabajo.

Persistencia Esta capa tiene una misión bien definida que es almacenar los datos de forma persistente. La capa de lógica tendrá que hacer uso de esta capa para poder dar respuesta a las diferentes peticiones que se le puedan solicitar.

13.2.3 Descomposición modular

Una hemos diferenciado los subsistemas básicos a un nivel conceptual del sistema, veremos una descomposición en componentes de los mismos. Estos componentes en ocasiones son fundamentales en el funcionamiento de otros componentes. De esta forma, el conjunto de funcionalidades delegadas a los componentes hacen posible el funcionamiento global del sistema.

Esta división del sistema al completo se representa en la figura 13.2, que es un diagrama de componentes siguiendo una estrategia de descomposición orientada a objetos. Se trata de un conjunto componentes que se comunican entre si.

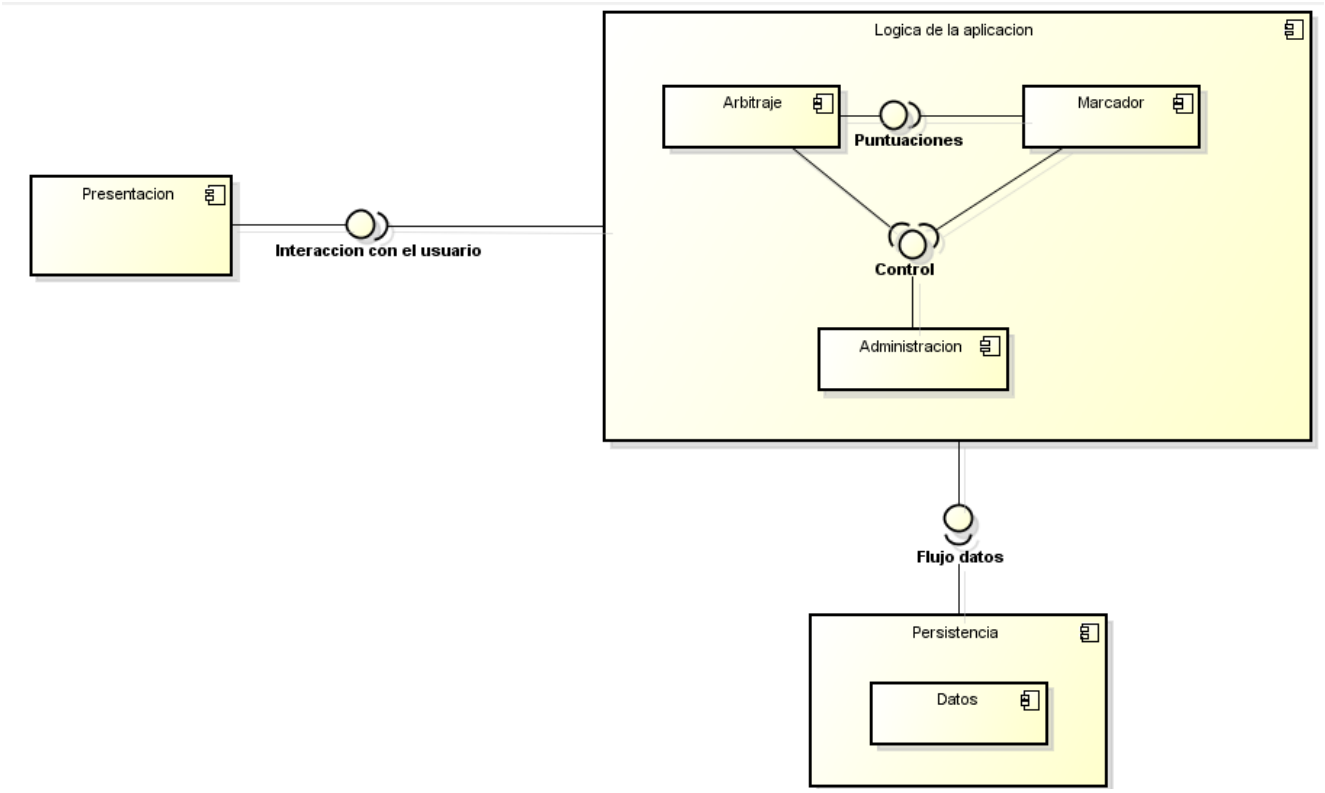


Figura 13.2 Diagrama de componentes

A continuación detallamos cada uno de los componentes que aparecen en la figura 13.2

Presentación Se encarga de interactuar con el usuario de la aplicación. Se comunica con la lógica de la aplicación informando de la petición que el cliente lanza e informando al usuario una vez se haya procesado dicha petición.

Arbitraje Este componente proporciona las puntuaciones al marcador y necesita del control por parte del módulo de administración para llevar a cabo su tarea.

Marcador Consume los datos producidos por el componente de arbitraje y requiere del control del módulo de administración para ir mostrando los datos correctamente.

Administración Se encarga de controlar a los módulos de marcador y arbitraje.

Datos Este componente contiene los datos que deben mantenerse de forma persistente en la aplicación. Se encuentra dentro del componente de persistencia.

13.2.4 Flujo de control

Para la parte del sistema encargada de llevar las peticiones realizadas desde la página web por parte de los clientes se utilizará un modelo de control centralizado basado en un modelo del gestor al tratarse de un sistema concurrente.

13.2.5 Diseño físico de la arquitectura

Con el análisis a nivel conceptual obtenido de las anteriores secciones, ya disponemos de un punto de partida más cercano a la solución del problema con una serie de elementos bien diferenciados y definidos y sus interrelaciones. Intentaremos por medio de la tecnología disponible ajustarnos de la forma más precisa al problema. En el siguiente diagrama de despliegue se muestra la propuesta para los diferentes dispositivos que conformarán el sistema y los protocolos que se utilizarán para comunicarlos. Ver figura 13.3.

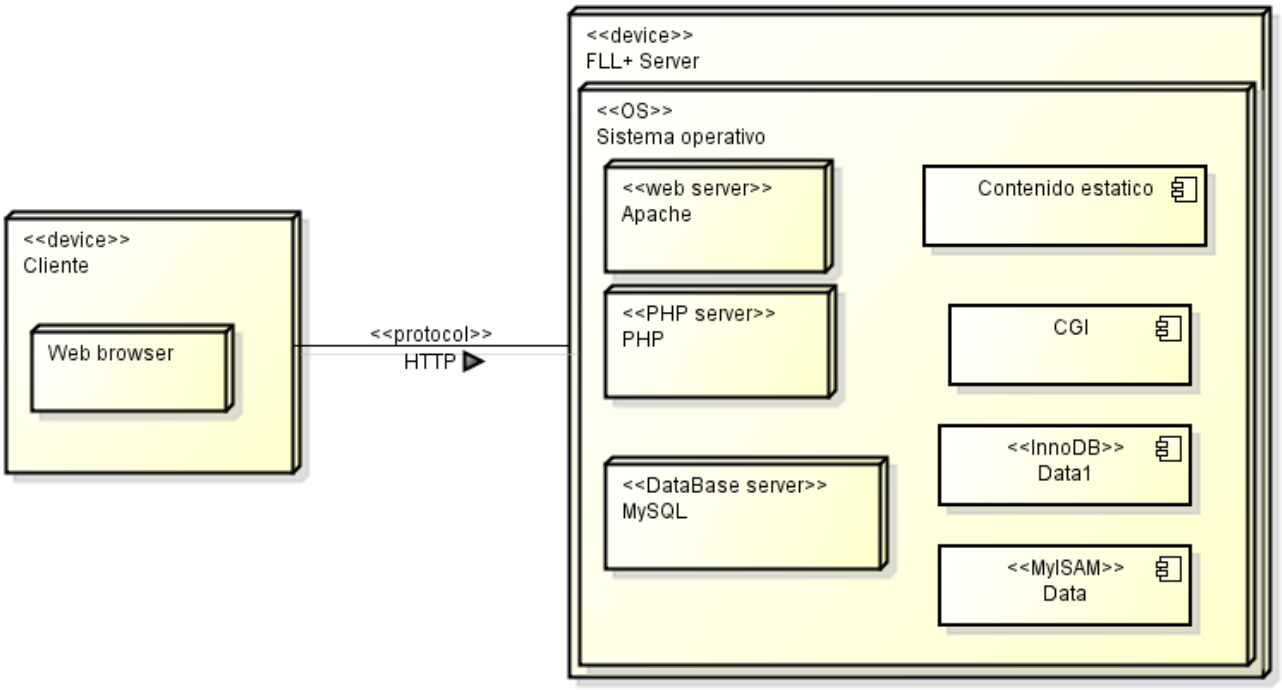


Figura 13.3 Diagrama de despliegue

A continuación pasamos a detallar los elementos presentados en la figura 13.3.

Servidor web Es el encargado de recibir las peticiones por parte del usuario y de enviar al servidor PHP los ficheros con los que tiene que trabajar.

Servidor PHP Este elemento se encargará de interpretar el código PHP para producir respuestas de forma dinámica al usuario.

Servidor de base de datos Este elemento es el encargado de almacenar y gestionar los datos de forma persistente, tales como los equipos, pruebas, puntuaciones o partidas.

Contenido estático El contenido estático es aquel que no cambiará, como por ejemplo las imágenes de la web, o las plantillas para las vistas del interfaz de usuario.

CGI Este módulo contiene los ficheros PHP que generan código HTML de forma dinámica

Datos InnoDB Este módulo contiene los datos gestionados por el motor de almacenamiento InnoDB que son aquellos que no van a ser consultados con mucha frecuencia. Principalmente son los datos de acceso al sistema.

Datos MyISAM Este módulo contiene todos los datos gestionados por el motor de almacenamiento MyISAM, que son aquellos que van a ser consultados frecuentemente. Básicamente hacen referencia a los datos necesarios para mostrar el marcador.

Nota: Vemos que en la parte del servidor hemos optado por introducir, el servidor web, el servidor PHP y el servidor MySQL en la misma máquina, sin embargo, nada impediría que esos elementos estuviesen en máquinas físicas diferentes.

Esto último requeriría de una configuración extra a la hora de instalar el sistema. Nosotros nos hemos ceñido a lo mostrado en la figura 13.3 para construir el manual de usuario que presentaremos en capítulos posteriores, sin embargo, como ya hemos mencionado, si el cliente quisiese mantener estos servidores en máquinas distintas nada lo impediría, con la salvedad de que debe tener los conocimientos necesarios para realizar una configuración a nivel software para poder llevar a cabo esos cambios.

13.3 Modelos del sistema

En primer lugar, explicaremos los patrones que hemos utilizado. Después veremos un modelo estático con las clases que formarán la aplicación para así tener un punto de partida en los modelos dinámicos, que definirán cómo se interrelacionan los objetos de estas clases en tiempo de ejecución para implementar una determinada funcionalidad.

13.3.1 Patrones utilizados

Cabe destacar en este punto los patrones que hemos utilizado para llevar a cabo la funcionalidad del sistema. A continuación enumeramos los patrones utilizados durante la etapa de diseño y damos una breve explicación de por qué hemos utilizado estos patrones:

MVC (Modelo-vista-controlador) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

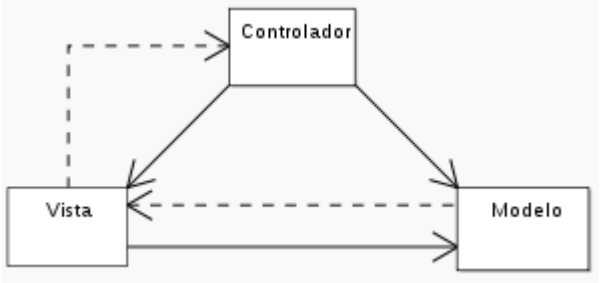


Figura 13.4 Patrón Modelo-Vista-Controlador

En la figura 13.4 podemos ver como se relacionan el modelo, la vista y el controlador. En un caso típico la secuencia de llamadas entre estos elementos es la siguiente: El usuario solicita hacer cierta acción, el controlador recibe esta petición y llama a la función del modelo que corresponda para después pasar a la vista los datos obtenidos desde el modelo.

Hemos utilizado este patrón de diseño ya que es uno de los más utilizados para la creación de páginas web y además nos permite diferenciar perfectamente entre el modelo de dominio, la persistencia de los datos y la interfaz gráfica de usuario.

Controlador frontal Es un patrón de diseño que se basa en usar un controlador como punto inicial, que denominaremos controlador frontal, para la gestión de las peticiones. El controlador frontal gestiona estas peticiones, y realiza algunas funciones como: comprobación de restricciones de seguridad, manejo de errores, mapear y delegación de las peticiones a otros componentes de la aplicación que se encargarán de generar la vista adecuada para el usuario. La figura 13.5 muestra un esquema de ello.

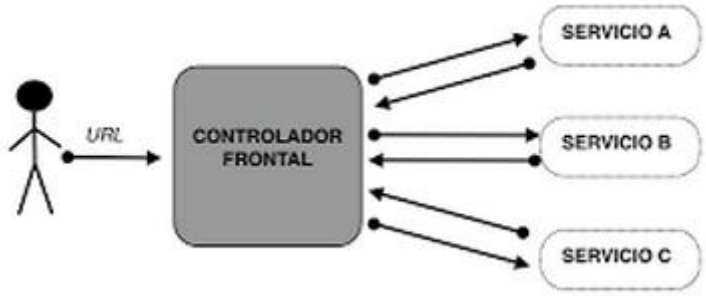


Figura 13.5 Patrón Controlador Frontal

Hemos optado por utilizar este patrón ya que nos permite tener un punto centralizado desde donde manejar las peticiones, aumentamos la reusabilidad del código y mejoramos la gestión de la seguridad.

DataMapper El patrón datamapper es un patrón que nos permite separar de forma clara, el modelo de dominio de la persistencia de datos. Se basa en la idea de que cada clase del modelo de dominio es representada por una clase datamapper para gestionar los datos entre la capa de lógica de la aplicación y la persistencia de los datos. Las clases datamapper son las encargadas de insertar, modificar, recuperar… objetos desde la base de datos hacia la memoria de la aplicación.

Para entender mejor este último párrafo pondremos un sencillo ejemplo. Tenemos una clase llamada Usuario que modela la información de un usuario de la aplicación. Al tener esta clase tendremos otra que podemos llamar DataMapperUsuario que contendrá métodos que permitan mover objetos de tipo Usuario desde/hacia la base de datos. La clase Usuario no es consciente de que existe una clase que está moviendo objetos de su tipo desde o hacia la base de datos y es esto precisamente lo que da potencia a este patrón, ya que estamos separando la lógica de la aplicación del acceso a la capa de persistencia de datos. El datamapper es el encargado de que exista una coherencia entre los objetos en memoria y la base de datos.

En la figura 13.6 vemos un diagrama que muestra el concepto del patrón DataMapper.

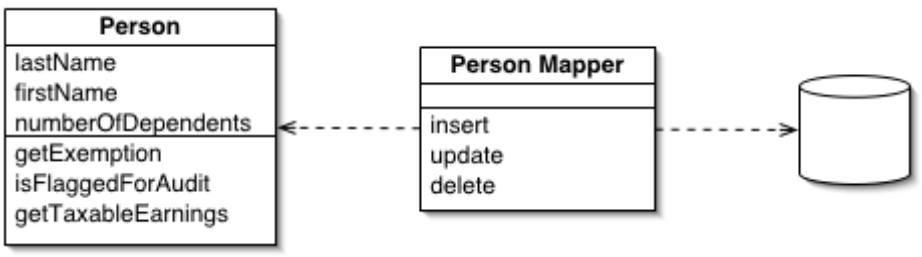


Figura 13.6 Patrón DataMapper

13.3.2 Modelos estáticos

En este apartado vamos a ver los diagramas de clases de diseño. Como vemos, los diagramas son considerablemente más grandes que los vistos en el bloque de análisis, por ello hemos dividido el diagrama en dos, en la figura 13.7 vemos la

relación entre las clases de dominio y sus datamappers y en la figura 13.16 vemos la relación entre los controladores, la lógica de dominio y la capa de presentación.

Cada clase del modelo de dominio tiene su clase datamapper para mover objetos desde/hacia la base de datos, salvo las clases ResumenPartida y Clasificacion ya que son clases que se construyen a partir de otros objetos, por ello, no tienen su clase datamapper asociada.

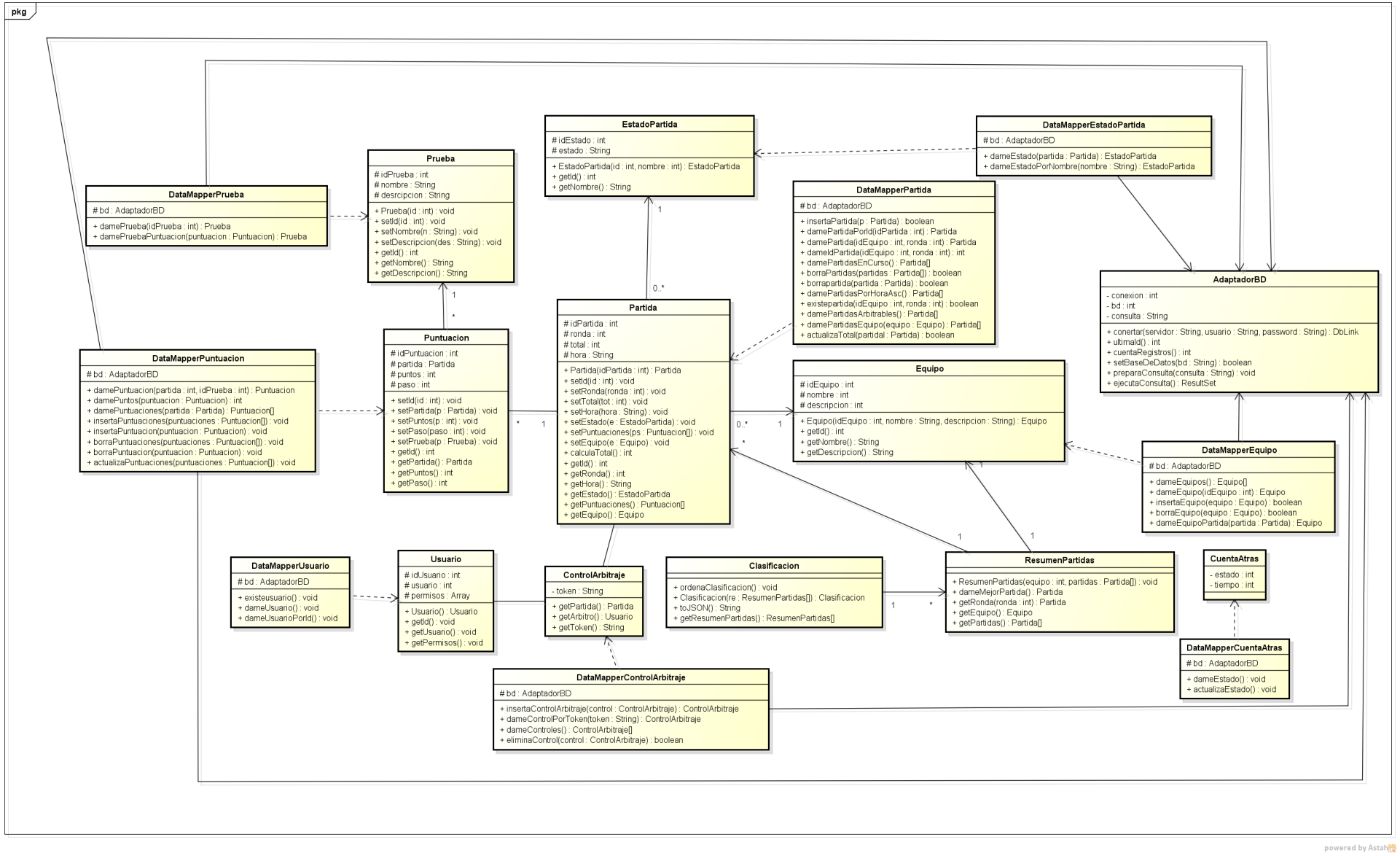


Figura 13.7 Diagrama de clases de diseño

Podemos ver que todas las clases datamapper contienen un atributo “bd”. Este atributo es de tipo “AdaptadorBD” y permite al datamapper realizar la comunicación con la base de datos.

A continuación pasamos a detallar cada una de las clases datamapper del diagrama 13.7.

13.3.2.1 DataMapperPartida

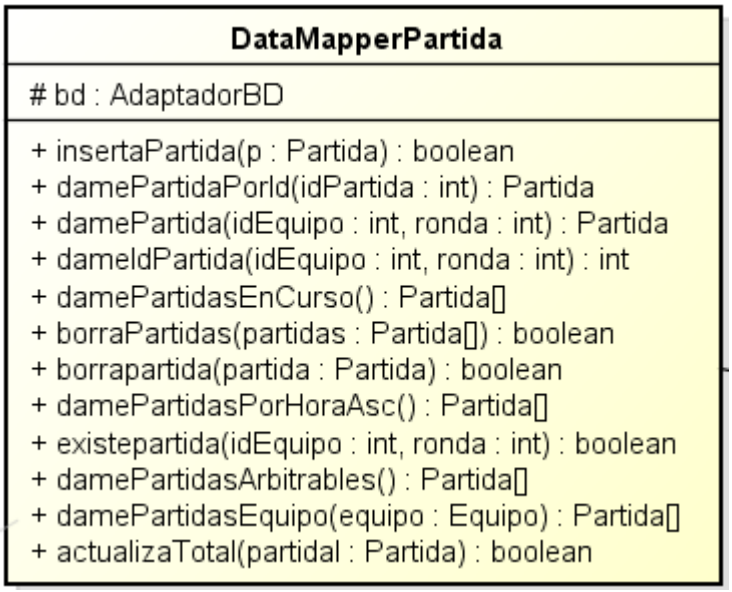


Figura 13.8 Clase “DataMapperPartida”

La clase DataMapperPartida es la encargada de mover objetos de tipo Partida hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo Partida.

A continuación pasamos a explicar los métodos de este datamapper:

+ insertaPartida(p : Partida) : boolean Este método introduce en la BBDD la partida pasada por argumento

+ damePartidaPorId(idPartida : int) : Partida Devuelve un objeto partida a partir del id pasado por parámetro

+ damePartida(idEquipo : int, ronda : int) : Partida Devuelve un objeto de tipo Partida a partir del idEquipo que la jugó y la ronda.

+ dameIdPartida(idEquipo : int, ronda : int) : int Devuelve el id de una partida a partir del id del Equipo que la jugó y la ronda.

+ damePartidasEnCurso() : Partida[] Devuelve el array de partidas que están en el estado “En curso”.

+ borraPartidas(partidas : Partida[]) : boolean Borra de la BBDD las partidas pasadas por argumento

+ borrapartida(partida : Partida) : boolean Borra de la BBDD la partida pasada por argumento.

+ damePartidasPorHoraAsc() : Partida[] Devuelve todas los objetos de tipo Partida ordenados por hora de planificación.

+ existepartida(idEquipo : int, ronda : int) : boolean Devuelve verdadero si la partida jugada por el equipo y ronda pasados por parámetro existe, falso en caso contrario.

+ damePartidasArbitrables() : Partida[] Devuelve un array de Partidas arbitrables, es decir, aquellas que estén “En curso” y que no estén siendo arbitradas.

+ damePartidasEquipo(equipo : Equipo) : Partida[] Devuelve todas las partidas jugadas por el equipo pasado por parámetro.

+ actualizaTotal(partida : Partida) : boolean Actualiza el total en la BBDD de la partida pasada por parámetro.

13.3.2.2 DataMapperEquipo

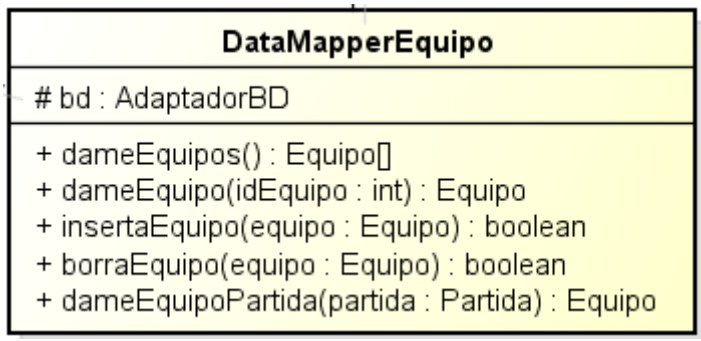


Figura 13.9 Clase "DataMapperEquipo"

La clase DataMapperEquipo es la encargada de mover objetos de tipo Equipo hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo Equipo.

A continuación pasamos a explicar los métodos de este datamapper:

+ dameEquipos() : Equipo[] Devuelve todos los equipos existentes en la BBDD.

+ dameEquipo(idEquipo : int) : Equipo Devuelve el equipo con el id pasado por parámetro.

+ insertaEquipo(equipo : Equipo) : boolean Inserta el equipo pasado por parámetro en la BBDD.

+ borraEquipo(equipo : Equipo) : boolean Borra el equipo pasado por parámetro de la BBDD.

+ dameEquipoPartida(partida : Partida) : Equipo Devuelve el equipo que jugo la partida pasada por parámetro.

13.3.2.3 DataMapperEstadoPartida

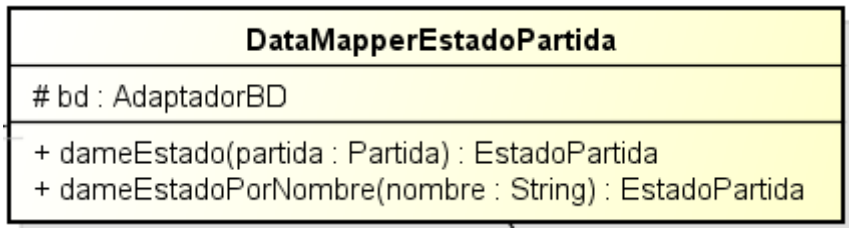


Figura 13.10 Clase "DataMapperEstadoPartida"

La clase DataMapperEstadoPartida es la encargada de mover objetos de tipo EstadoPartida hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo EstadoPartida.

A continuación pasamos a explicar los métodos de este datamapper:

+ dameEstado(partida : Partida) : EstadoPartida Devuelve el estado de la partida pasada por parámetro.

+ dameEstadoPorNombre(nombre : String) : EstadoPartida Devuelve un objeto EstadoPartida a partir del nombre pasado por parámetro.

13.3.2.4 DataMapperPrueba

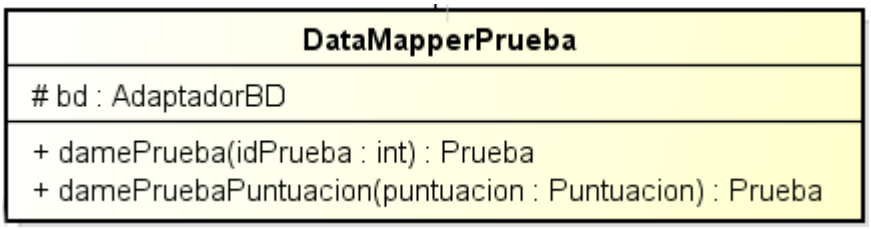


Figura 13.11 Clase "DataMapperPrueba"

La clase DataMapperPrueba es la encargada de mover objetos de tipo Prueba hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo Prueba.

A continuación pasamos a explicar los métodos de este datamapper:

+ damePrueba(idPrueba : int) : Prueba Devuelve la prueba con el id pasado por parámetro.

+ damePruebaPuntuacion(puntuacion : Puntuacion) : Prueba Devuelve la prueba asociada al objeto Puntuacion pasado por parámetro.

13.3.2.5 DataMapperPuntuacion

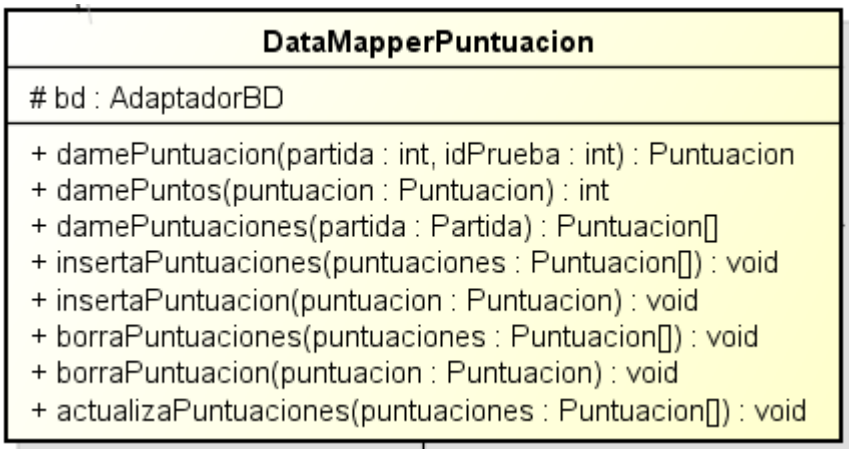


Figura 13.12 Clase "DataMapperPuntuacion"

La clase DataMapperPuntuacion es la encargada de mover objetos de tipo Puntuacion hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo Puntuacion.

A continuación pasamos a explicar los métodos de este datamapper:

+ damePuntuacion(partida : int, idPrueba : int) : Puntuacion Devuelve la puntuación asociada a la partida y prueba pasada por parámetro.

+ damePuntos(puntuacion : Puntuacion) : int Devuelve los puntos del objeto Puntuacion pasado por parámetro.

+ damePuntuaciones(partida : Partida) : Puntuacion[] Devuelve todos los objetos Puntuacion asociados a la partida pasada por parámetro.

+ insertaPuntuaciones(puntuaciones : Puntuacion[]) : void Inserta en la BBDD los objetos Puntuacion pasados por parámetro.

+ insertaPuntuacion(puntuacion : Puntuacion) : void Inserta en la BBDD el objeto Puntuacion pasado por parámetro.

+ borraPuntuaciones(puntuaciones : Puntuacion[]) : void Elimina de la BBDD los objetos Puntuacion pasados por parámetro.

+ borraPuntuacion(puntuacion : Puntuacion) : void Elimina de la BBDD el objeto Puntuacion pasado por parámetro.

+ actualizaPuntuaciones(puntuaciones : Puntuacion[]) : void Actualiza en la BBDD los objetos Puntuacion pasados por parámetro.

13.3.2.6 DataMapperUsuario

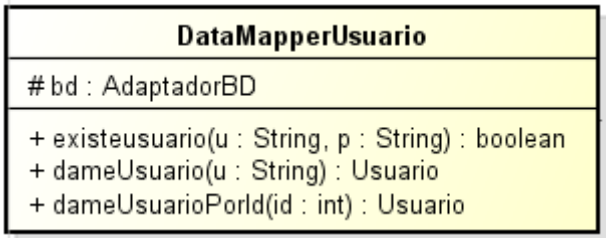


Figura 13.13 Clase "DataMapperUsuario"

La clase DataMapperUsuario es la encargada de mover objetos de tipo Usuario hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo Usuario.

A continuación pasamos a explicar los métodos de este datamapper:

+ existeusuario(u : String, p : String) : Boolean Devuelve verdadero si existe en BBDD el usuario con la contraseña pasados por parámetro.

+ dameUsuario(u : String) : Usuario Devuelve el objeto Usuario con el nombre de usuario pasado por parámetro.

+ dameUsuarioPorId(id : int) : Usuario Devuelve el objeto Usuario asociado al id pasado por parámetro.

13.3.2.7 DataMapperControlArbitraje

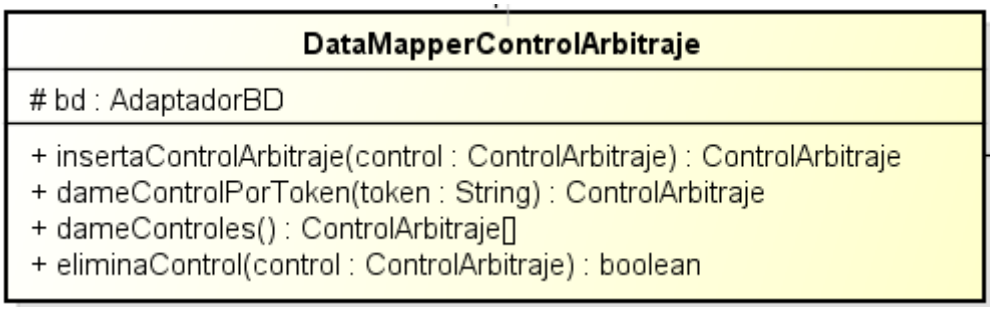


Figura 13.14 Clase "DataMapperControlArbitraje"

La clase DataMapperControlArbitraje es la encargada de mover objetos de tipo ControlArbitraje hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo ControlArbitraje.

A continuación pasamos a explicar los métodos de este datamapper:

+ insertaControlArbitraje(control : ControlArbitraje) : ControlArbitraje Introduce en la BBDD el objeto ControlArbitraje pasado por parámetro.

+ dameControlPorToken(token : String) : ControlArbitraje Devuelve un objeto ControlArbitraje asociado al token pasado por parámetro.

+ dameControles() : ControlArbitraje[] Devuelve todos los objetos ControlArbitraje presentes en la BBDD.

+ eliminaControl(control : ControlArbitraje) : boolean Elimina de la BBDD el control de arbitraje pasado por parámetro.

13.3.2.8 DataMapperCuentaAtras

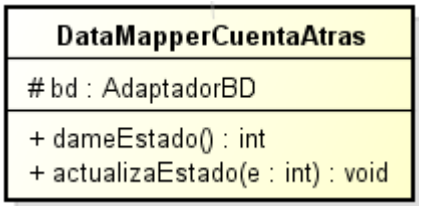


Figura 13.15 Clase "DataMapperCuentaAtras"

La clase DataMapperCuentaAtras es la encargada de mover objetos de tipo CuentaAtras hacia/desde la base de datos y, en general, realizar acciones que requieren acceso a la base de datos para objetos de tipo CuentaAtras.

A continuación pasamos a explicar los métodos de este datamapper:

+ dameEstado() : int Devuelve el estado de la cuenta atrás.

+ actualizaEstado(e : int) : void Actualiza en BBDD el estado de la cuenta atrás.

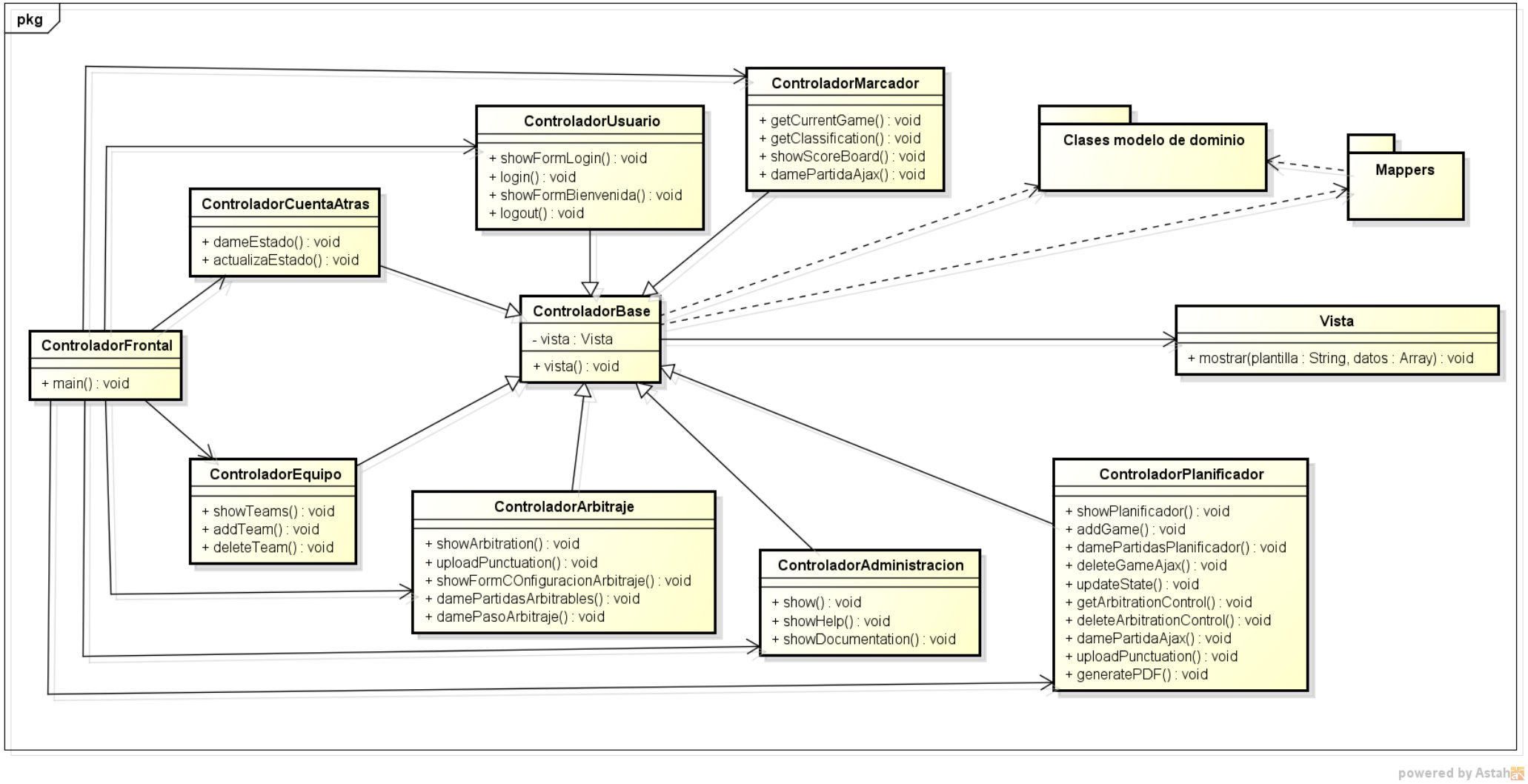


Figura 13.16 Diagrama de clases (Controladores)

Nota:

En el diagrama de la figura 13.16 podemos ver como cada controlador hereda de la clase "ControladorBase" heredando así el atributo vista, el cual es de tipo Vista y contiene un método al que se le puede pasar la ruta de una vista y los datos para ser mostrados al usuario.

Por otra parte, la clase "ControladorFrontal" está asociada a todos los controladores, delegando así, las peticiones del usuario al controlador que corresponda. También podemos ver que los controladores dependen del modelo y de los datamappers, ya que tienen que consultarlos para poder obtener los datos que pasará a la vista.

En conclusión, los diagramas mostrados en las figuras 13.7 y 13.16, muestran el modelo estático del sistema y se puede observar cómo se han llevado a cabo los patrones modelo-vista-controlador, datamapper y controlador frontal.

A continuación vamos a dar una breve descripción de cada uno de los controladores que forman el sistema:

13.3.2.9 ControladorUsuario

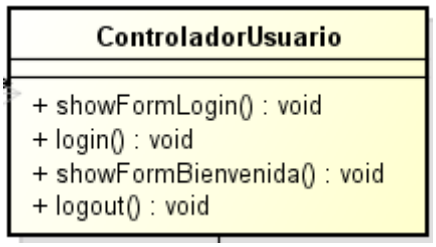


Figura 13.17 Clase "ControladorUsuario"

Es el controlador encargado de gestionar aquellas peticiones relacionadas con los usuarios del sistema.

Breve descripción de los métodos:

+ showFormLogin() : void Muestra el formulario de acceso al sistema FLL+.

+ login() : void Comprueba que los datos introducidos en el formulario de login y da paso al sistema en caso de que los datos sean correctos.

+ showFormBienvenida() : void Muestra la pantalla de bienvenida al usuario, desde donde puede acceder a aquellas partes del sistema para las que tenga permiso.

+ logout() : void Cierra la sesión del usuario.

13.3.2.10 ControladorEquipo

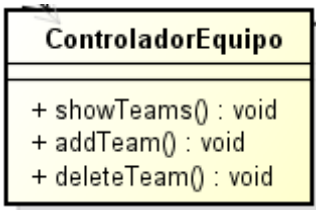


Figura 13.18 Clase "ControladorEquipo"

Este controlador es el encargado de manejar las peticiones relacionadas con la gestión de los equipos, tales como, añadir o eliminar equipos.

+ showTeams() : void Muestra los equipos existentes en el sistema.

+ addTeam() : void Añade un equipo al sistema.

+ deleteTeam() : void Borra un equipo del sistema.

13.3.2.11 ControladorArbitraje

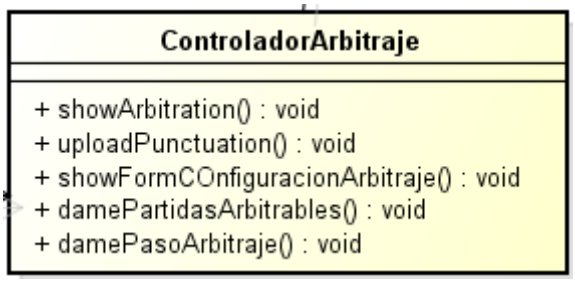


Figura 13.19 Clase "ControladorArbitraje"

Es el controlador que se encarga de manejar las peticiones enviadas por el usuario relacionadas con el arbitraje de las partidas.

Estos son sus métodos:

+ showArbitration() : void Muestra el panel de arbitraje.

+ uploadPunctuation() : void Inserta una puntuación para una prueba y partida concretas.

+ showFormConfiguracionArbitraje() : void Muestra el formulario de configuración de arbitraje, es decir, el formulario desde donde el árbitro selecciona la partida que va a arbitras.

+ damePartidasArbitrables() : void Muestra las partidas arbitrables, que son aquellas que están marcadas con el estado "En curso" y no están siendo arbitradas por ningún árbitro.

+ damePasoArbitraje() : void Comprueba que la partida que ha seleccionado el árbitro no está siendo arbitrada por otro y genera un token de arbitraje que le permitirá arbitrar la partida si procede.

13.3.2.12 ControladorCuentaAtras

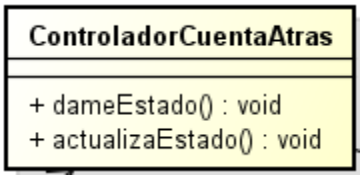


Figura 13.20 Clase "ControladorCuentaAtras"

Es el controlador encargado de gestionar las peticiones relacionadas con el estado de la cuenta atrás.

Sus métodos son los siguientes:

+ dameEstado() : void Muestra el estado de la cuenta atrás.

+ actualizaEstado() : void Actualiza el estado de la cuenta atrás.

13.3.2.13 ControladorAdministracion

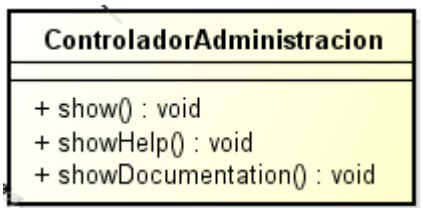


Figura 13.21 Clase "ControladorAdministracion"

Este controlador es el encargado de gestionar las peticiones generales del panel de administración.

A continuación pasamos a hacer una breve descripción de sus métodos:

+ show() : void Muestra la pantalla de bienvenida del panel de administración

+ showHelp() : void Muestra la ayuda en el panel de administración

+ showDocumentation() : void Muestra la documentación del panel de administración

13.3.2.14 ControladorMarcador

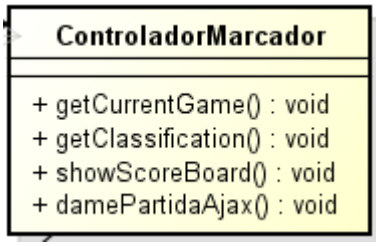


Figura 13.22 Clase "ControladorMarcador"

Esta clase es la encargada de manejar las peticiones relacionadas con el marcador.

Breve descripción de sus métodos:

+ getCurrentGame() : void Muestra la/s partidas que están en estado "En curso".

+ getClassification() : void Muestra la clasificación general del evento FLL.

+ showScoreBoard() : void Muestra el marcador.

+ damePartidaAjax() : void Muestra una partida concreta a través de una petición Ajax.

13.3.2.15 ControladorPlanificador

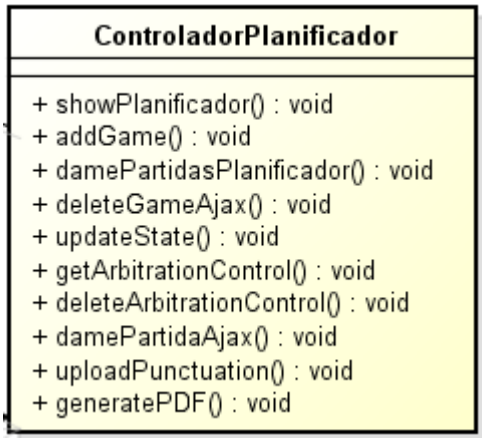


Figura 13.23 Clase "ControladorPlanificador"

Este controlador se encarga de gestionar las peticiones del usuario relacionadas con el panel de planificación, tales como, insertar una partida, modificar una puntuación…

+ **showPlanificador() : void** Muestra el menú de planificación dentro del panel del administrador.

+ **addGame() : void** Añade una nueva partida.

+ **damePartidasPlanificador() : void** Muestra todas las partidas para ser mostradas en el menú de planificación.

+ **deleteGameAjax() : void** Elimina una partida concreta.

+ **updateState() : void** Actualiza el estado de una partida concreta.

+ **getArbitrationControl() : void** Muestra todos los controles de arbitraje. Recordamos que la clase ControlArbitraje relacionaba las partidas con los árbitros, para evitar que varios árbitros puedan arbitrar la misma partida y que el jefe de árbitros pueda llevar un control relativo al arbitraje desde el panel de administración.

+ **deleteArbitrationControl() : void** Borra un control de arbitraje concreto.

+ **damePartidaAjax() : void** Muestra una partida concreta a través de una petición Ajax.

+ **uploadPunctuation() : void** Actualiza una puntuación de una prueba y partida concretas.

+ **generatePDF() : void** Genera un informe PDF de una partida concreta.

13.3.3 Modelos dinámicos

En este apartado vamos a describir el sistema desde una perspectiva dinámica, es decir, mostraremos que secuencia de mensaje desatan las peticiones de los actores y las clases que lo forman para dar la respuesta adecuada a cada una de estas peticiones

Vamos a ver los diagramas de secuencia de los casos de uso principales de la aplicación. En estos diagramas se muestra cómo se comunican o relacionan las clases para construir una respuesta ante una petición por parte del usuario.

13.3.3.1 Diagrama de secuencia (Insertar equipo)

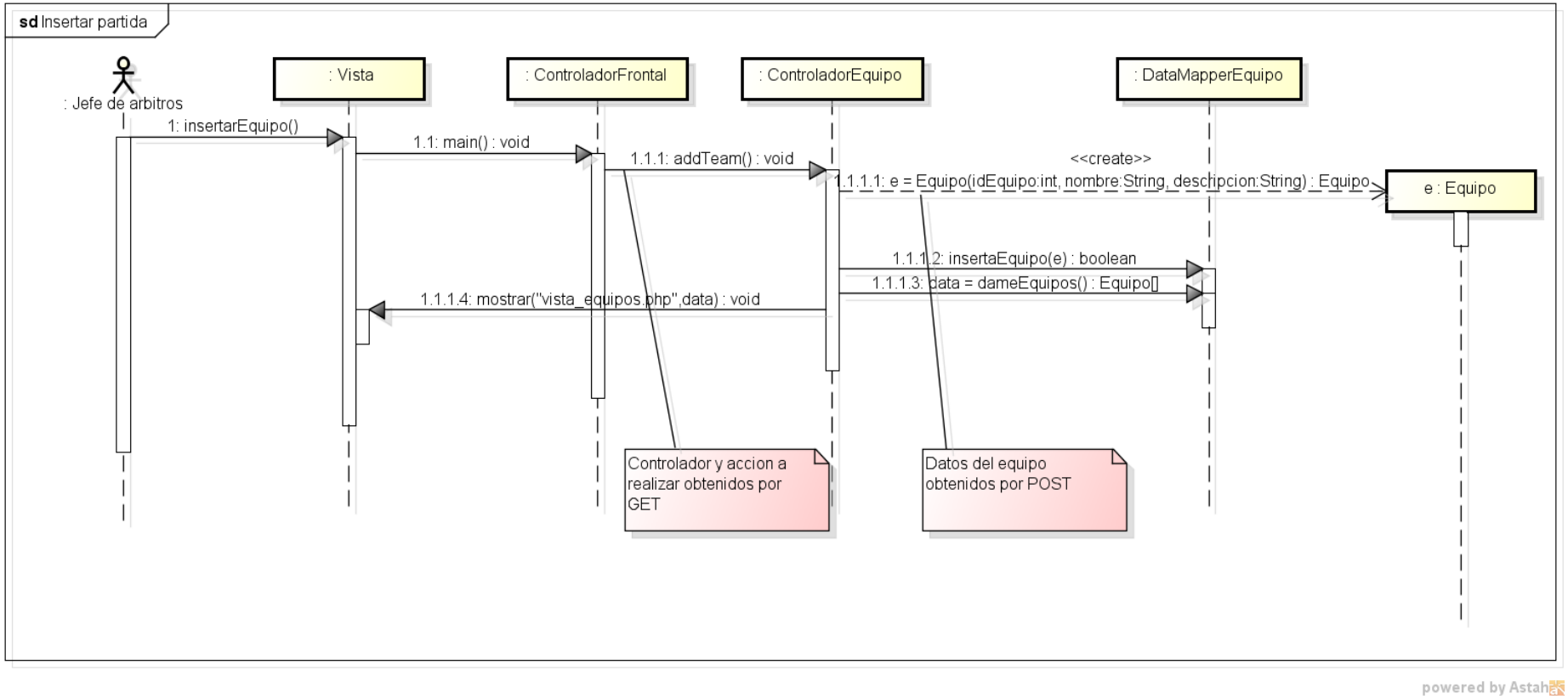


Figura 13.24 Diagrama de secuencia "Insertar equipo"

13.3.3.2 Diagrama de secuencia (Eliminar equipo)

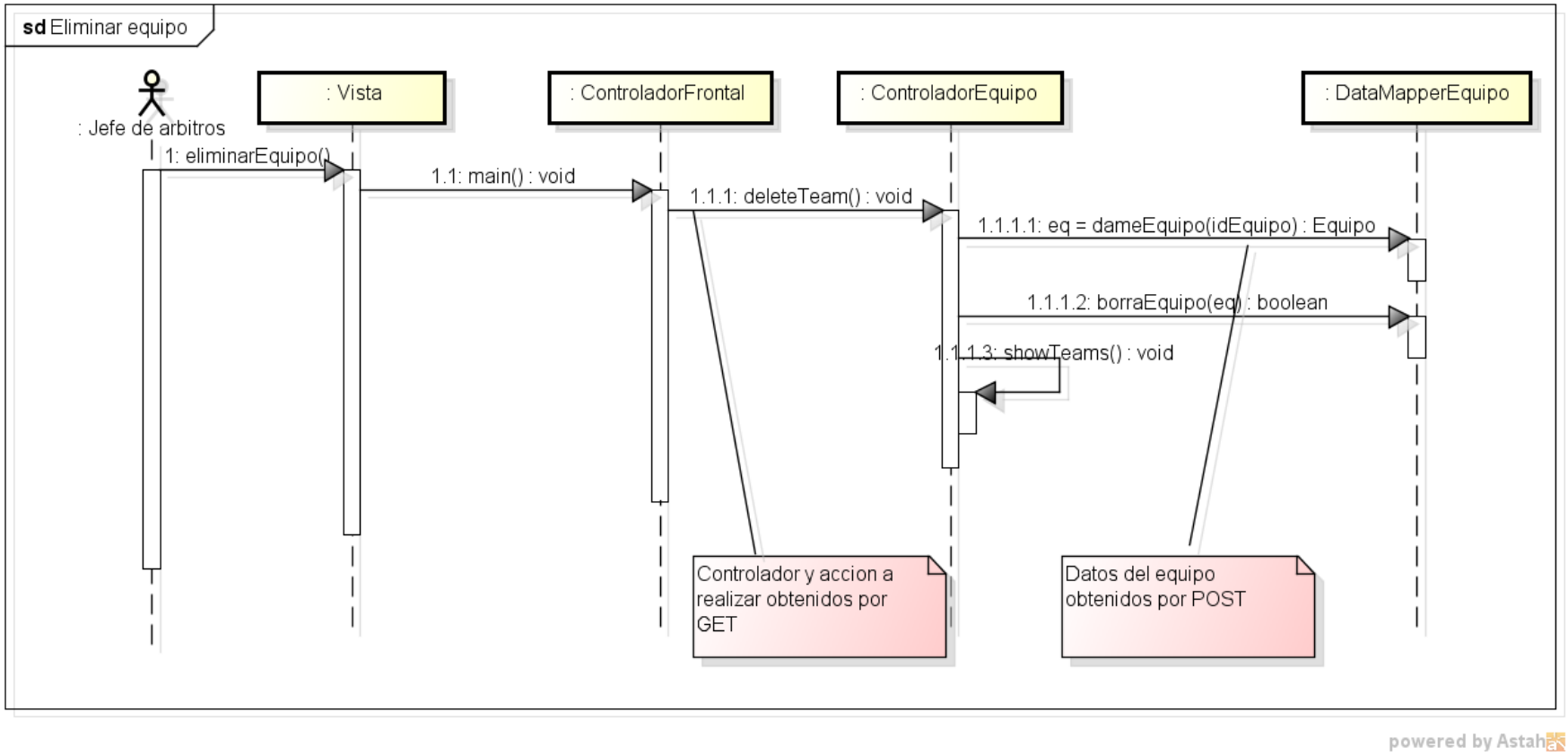


Figura 13.25 Diagrama de secuencia "Eliminar equipo"

13.3.3.3 Diagrama de secuencia (Exportar partida)

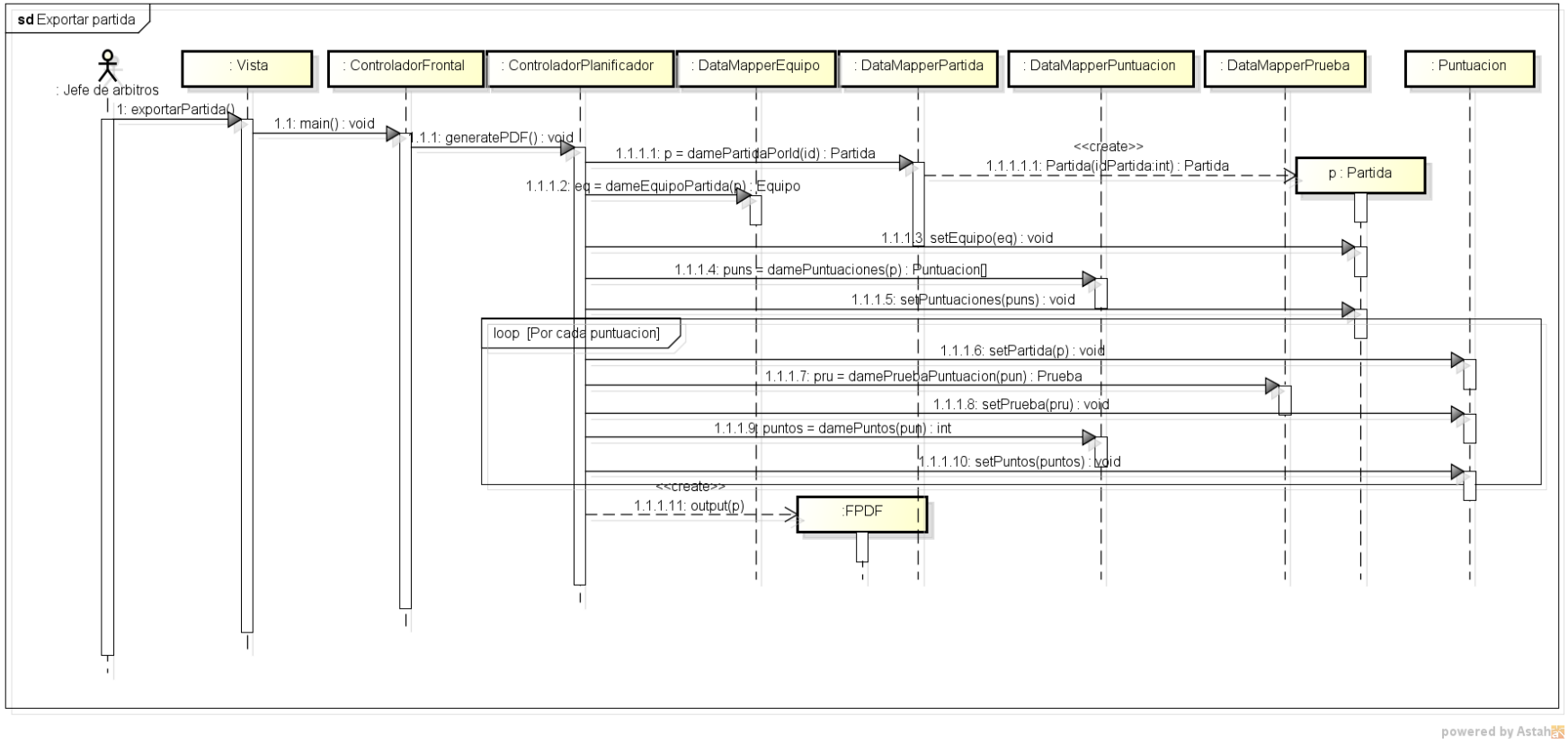


Figura 13.26 Diagrama de secuencia "Exportar partida"

13.3.3.4 Diagrama de secuencia (Planificar partida)

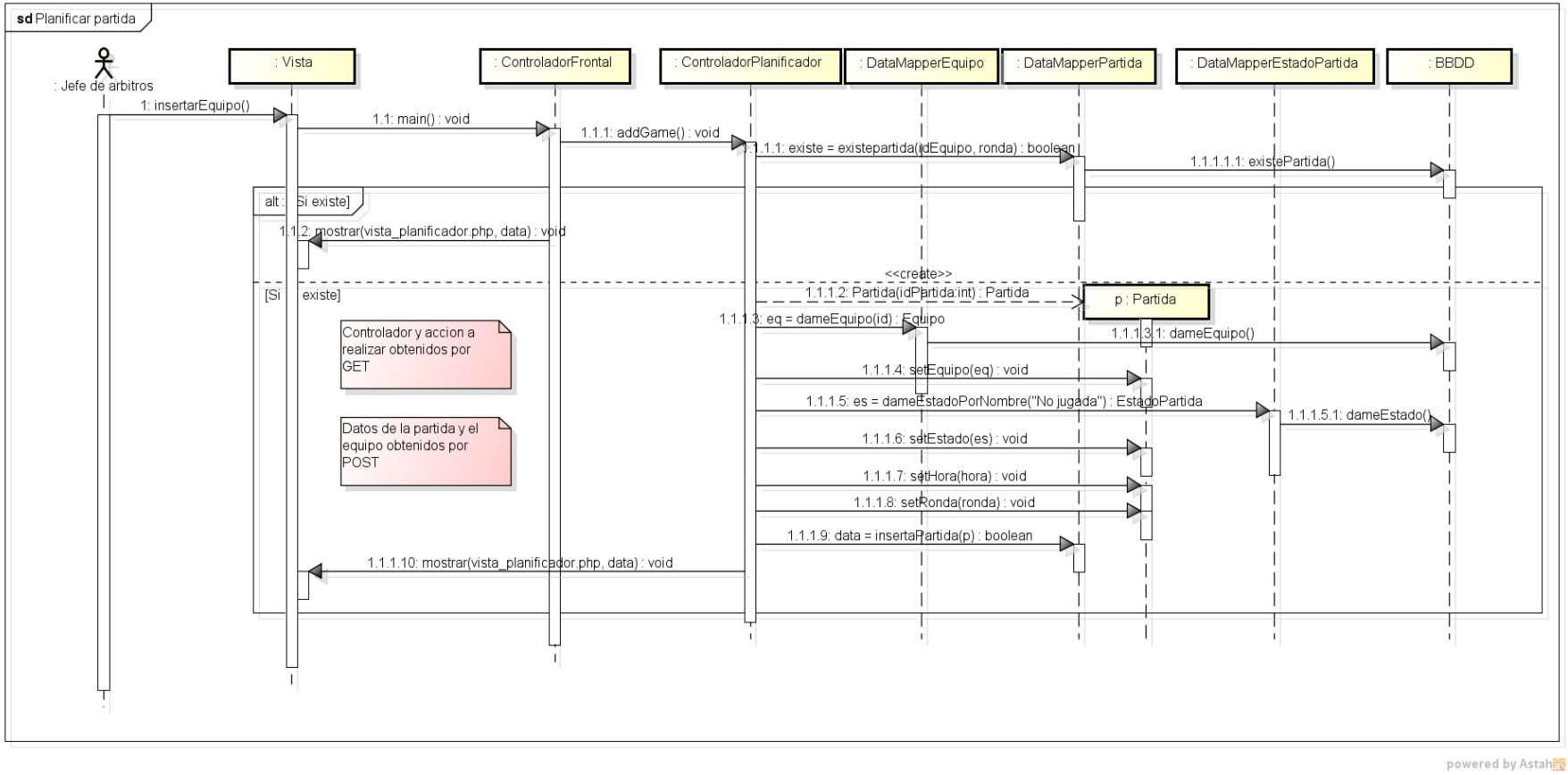


Figura 13.27 Diagrama de secuencia "Planificar partida"

13.3.3.5 Diagrama de secuencia (Ver clasificación)

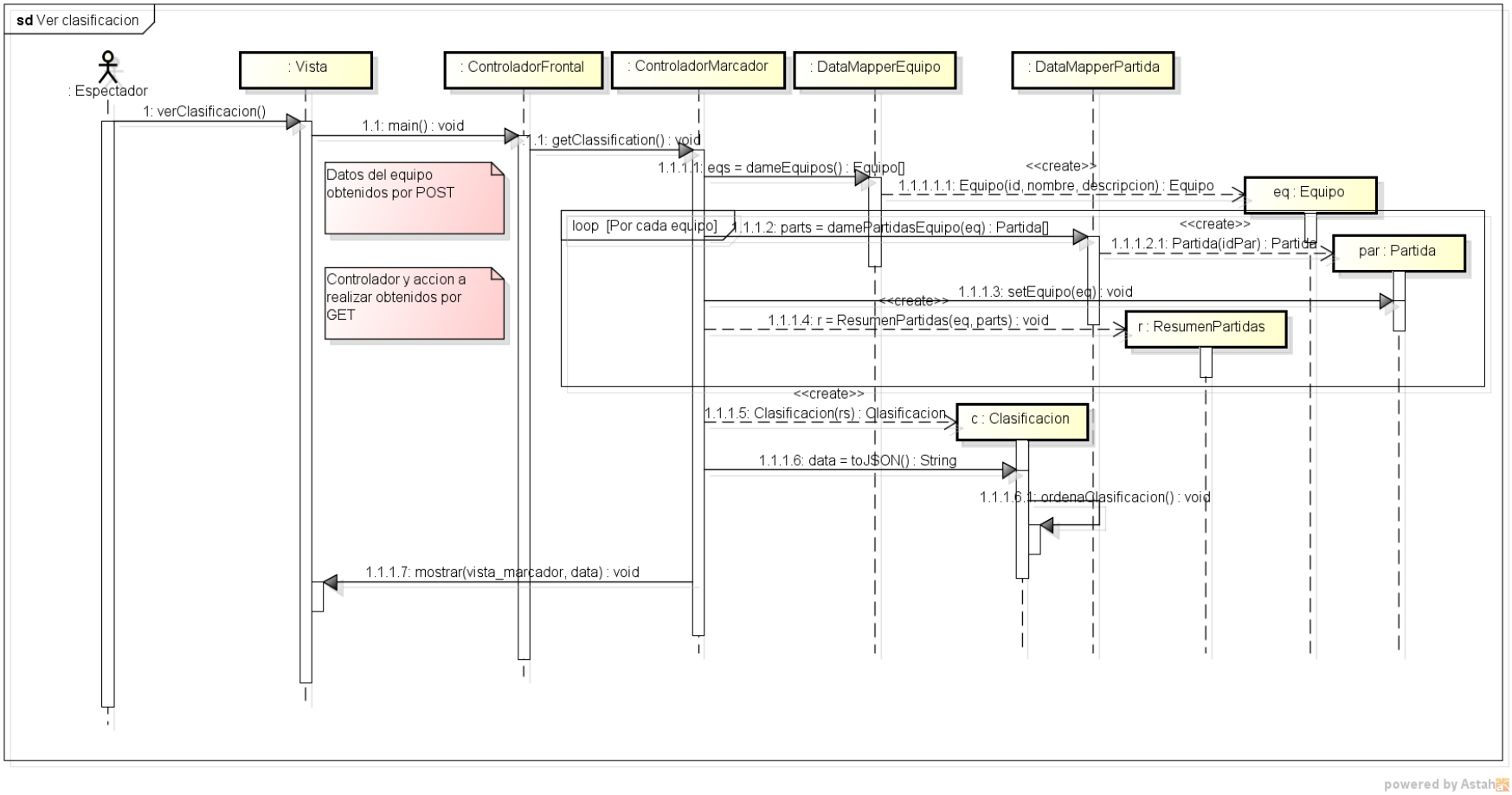


Figura 13.28 Diagrama de secuencia "Ver clasificación"

13.3.3.6 Diagrama de secuencia (Ver resultados partida en curso)

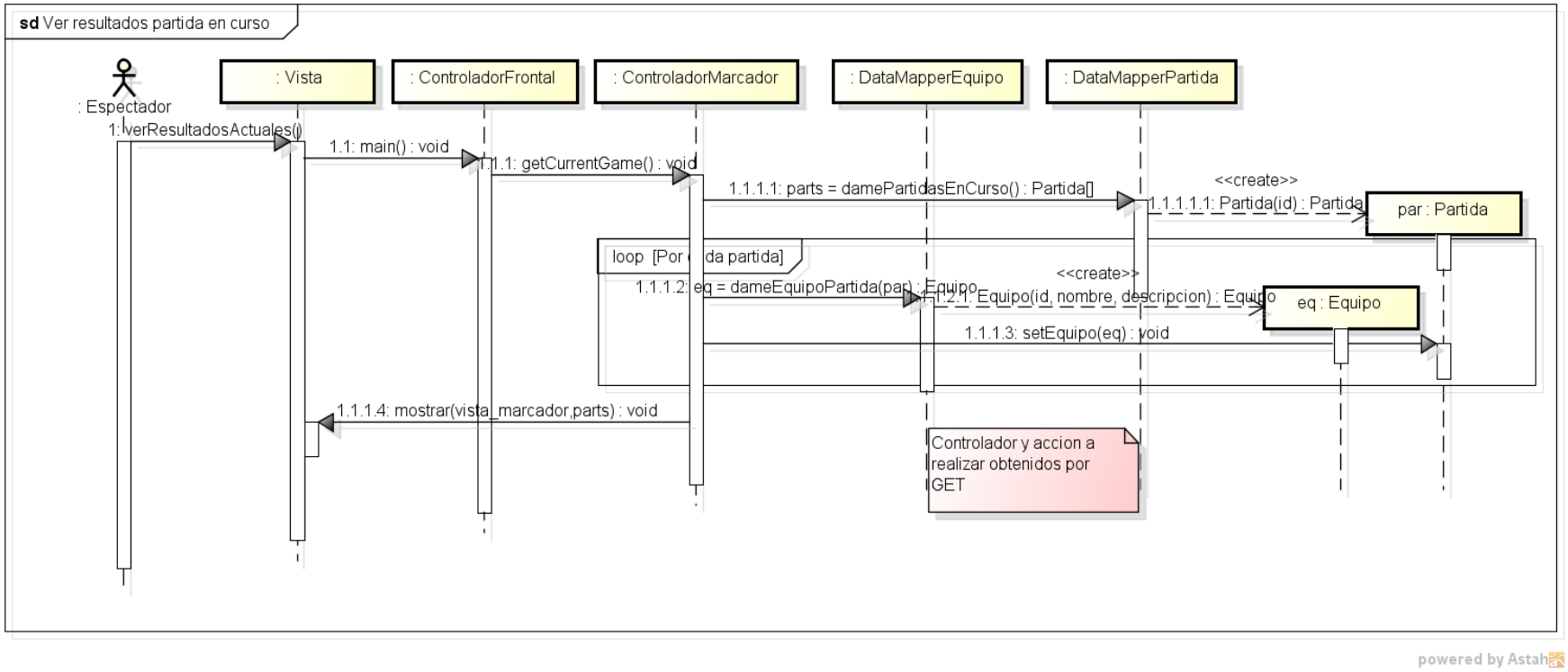


Figura 13.29 Diagrama de secuencia "Ver resultados partida en curso"

13.3.3.7 Diagrama de secuencia (Eliminar partida)

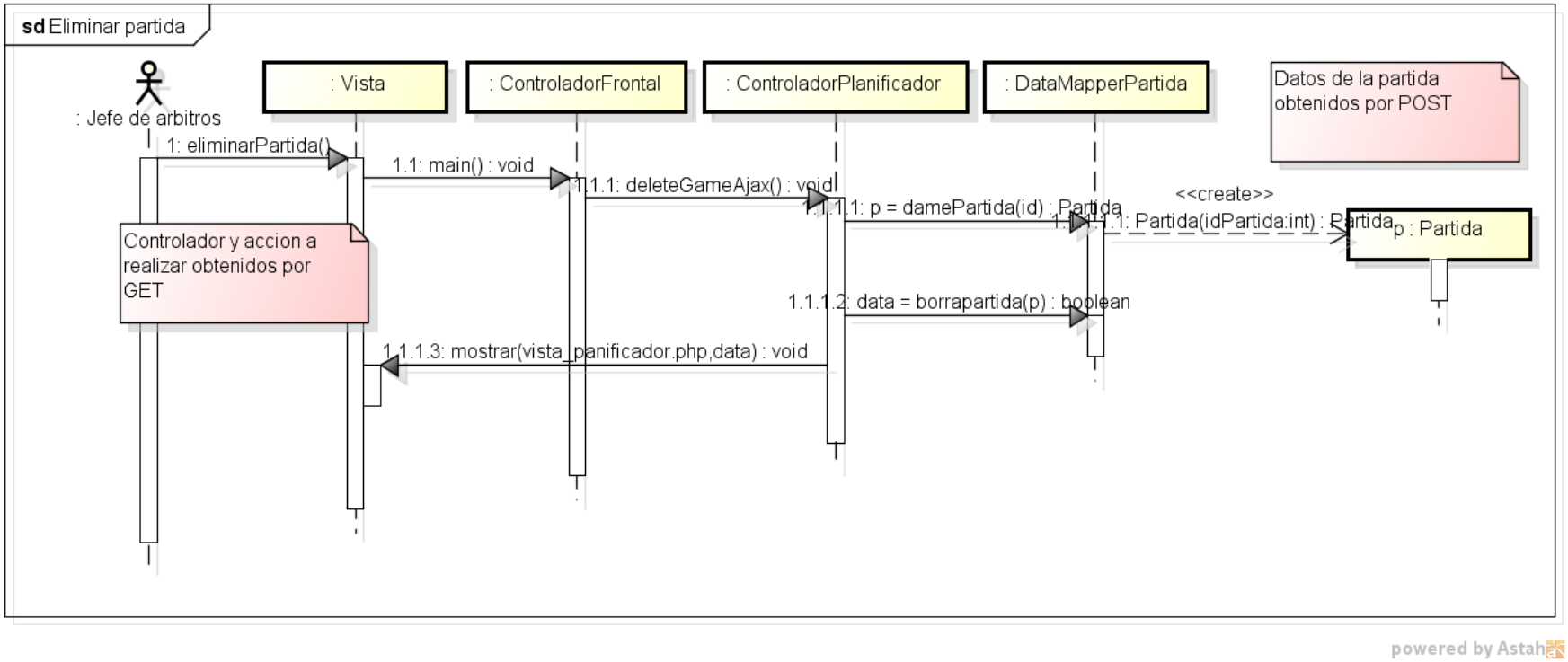


Figura 13.30 Diagrama de secuencia "Eliminar partida"

13.3.3.8 Diagrama de secuencia (Introducir puntuación prueba)

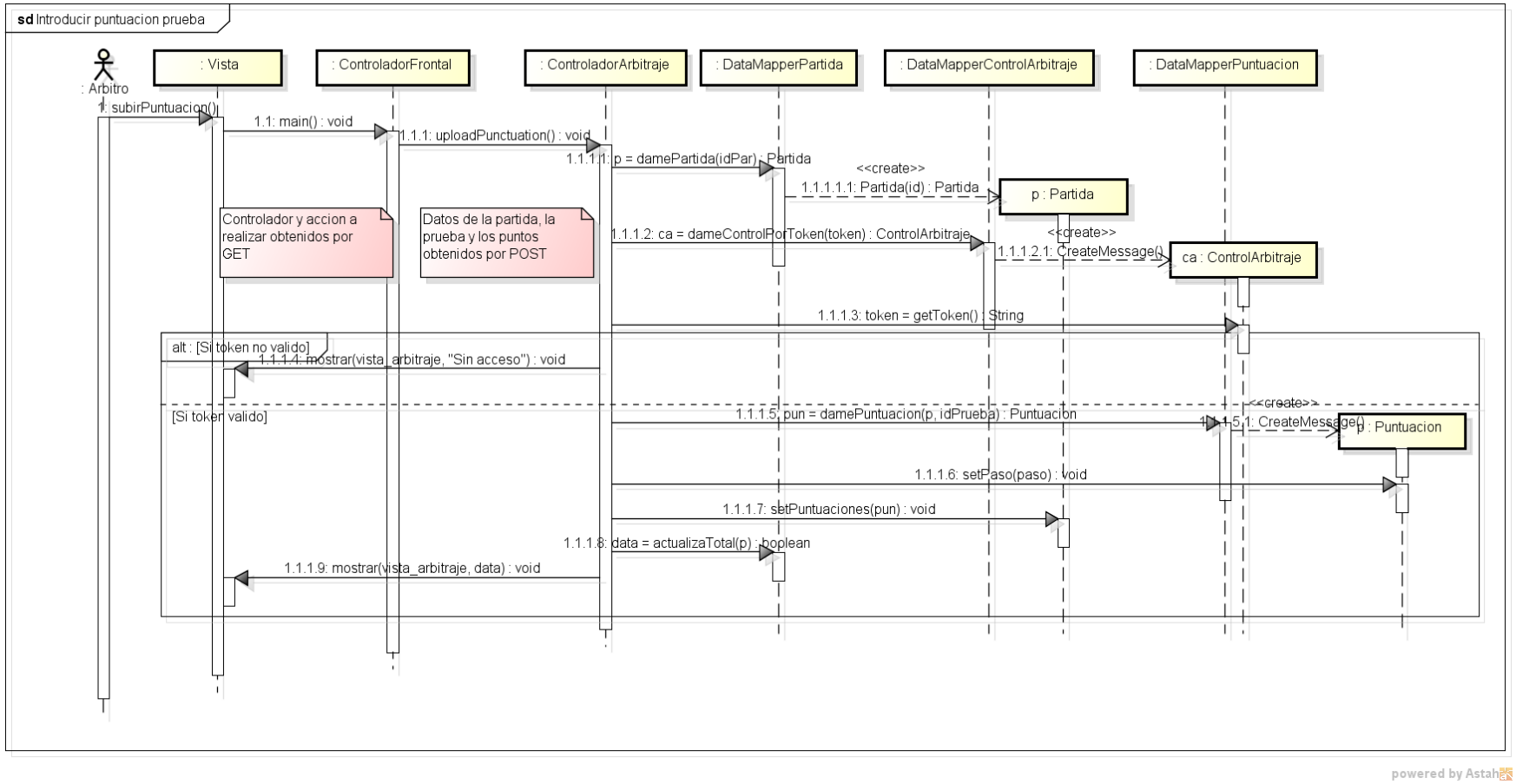


Figura 13.31 Diagrama de secuencia "Introducir puntuación prueba"

Bloque V. Implementación

Capítulo 14. Introducción a la implementación

En este bloque describiremos brevemente cómo hemos convertido todas las especificaciones descritas en los bloques anteriores en un sistema software real. La idea es no alargarse mucho en este bloque, ya que la implementación propiamente dicha está adjunta en el CD que acompaña esta memoria y además el código tiene gran cantidad de comentarios que facilitan la depuración y compresión del mismo.

El lenguaje de programación utilizado en el servidor para generar páginas web de forma dinámica, ha sido PHP, ya que es una de los más utilizados en sistemas web y también uno de los más rápidos y que menos recursos consumen.

En la parte de presentación al usuario o interfaz gráfica, se ha utilizado JavaScript, que es un lenguaje interpretado en el cliente que nos permite obtener ciertas ventajas, como acceder fácilmente a la estructura del documento HTML, crear pequeñas animaciones…

En la parte de persistencia, se ha utilizado el sistema gestor de bases de datos MySQL, ya que es uno de los más utilizados, gratuito y con buenas prestaciones. Además incluye los dos motores de almacenamiento que mencionamos en el bloque de diseño, “InnoDB” y “MyISAM”.

Para escribir el código PHP, JavaScript y HTML se ha utilizado la herramienta Dreamweaver, la cual, está especializada en la creación de aplicaciones web.

Capítulo 15. Tecnologías utilizadas

15.1 Ajax

Las tecnologías utilizadas para implementar el sistema desde un punto estático ya se han mencionado en capítulos anteriores, sin embargo, el principal reto que se presenta en la parte de implementación es que tecnología utilizar para dotar al sistema de un interfaz que muestre los datos en tiempo real a los usuarios.

Las dos principales opciones para esto último son node.js y Ajax (Asynchronous JavaScript And XML). Finalmente hemos optado por utilizar Ajax, ya que node.js, es una tecnología aún muy poco madura y que además no consta de mucha documentación.

Con la tecnología Ajax hemos conseguido realizar llamadas a web services cada cierto tiempo, sin necesidad de que el usuario tenga que recargar la página ni realizar ninguna otra acción, de ésta forma hacemos que el usuario disponga de un interfaz en el que se van mostrando los datos en tiempo real.

15.2 Web services

Los web services que hemos creado son llamados a través de la tecnología Ajax cada cierto tiempo. Estos web services devuelven datos en formato JSON a partir de las peticiones Ajax. Una vez esos datos han llegado al navegador, son interpretados por el framework jQuery y presentados en el lugar que corresponda.

A continuación, pasamos a describir los web services que existen en el sistema FLL+.

15.2.1 Web service (Partida actual)

La función que tiene este web service es devolver los datos de la partida actual, es decir, la partida que se esté jugando en un instante dado, en formato JSON.

En la figura 15.1 podeos ver un ejemplo de la respuesta obtenida de este servicio web.

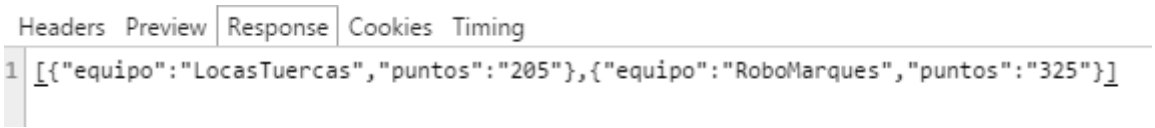


Figura 15.1 Respuesta web service (Partida actual)

Como vemos, el JSON que devuelve, está formado por un array de elementos. Cada uno de estos elementos contiene:

-El nombre del equipo

-Los puntos obtenidos hasta el momento.

A este servicio web se puede acceder desde la siguiente url:

http://sudominio.es/fll+/marcador /index.php?controlador=Marcador&accion=getCurrentGame

15.2.2 Web service (Clasificación)

La función del web service de clasificación es obtener la lista de los equipos ordenada según la puntuación que han obtenido en cada una de las rondas.

En la figura 15.2 se muestra un ejemplo de la respuesta de este web service.


```
[
    {
        "equipo": "RoboMarques",
        "total": "325",
        "partidas": [
            {
                "ronda": "1",
                "puntos": "0",
                "estado": "2",
                "id": "8"
            },
            {
                "ronda": "2",
                "puntos": "325",
                "estado": "3",
                "id": "9"
            },
            {
                "ronda": "3",
                "puntos": "225",
                "estado": "1",
                "id": "10"
            }
        ]
    },
    {
        "equipo": "Robot Sabios",
        "total": "308",
        "partidas": [
            {
                "ronda": "1",
                "puntos": "0",
                "estado": "2",
                "id": "45"
            },
            {
```


Figura 15.2 Respuesta web service (Clasificación)

Como vemos, este servicio web nos devuelve un array de elementos, cada uno de ellos contiene:

-El nombre del equipo.

-Su puntuación total.

-Un array de partidas, cada una de la cuales contiene:

-La ronda

-Los puntos obtenidos en esa ronda

-El estado de esa partida.

-El identificador único de esa partida.

Para acceder a este servicio web basta con acceder a la siguiente url:

http://sudominio.es/fll+/marcador/index.php?controlador=Marcador&accion=getClassification

15.2.3 Web Service (Actualización de puntuación)

Este servicio web se encarga de incluir las puntuaciones recibidas a través del método POST devolviendo la puntuación total de la partida o error en caso de que no se haya podido llevar a cabo la actualización.

Un ejemplo de la respuesta lo podemos ver en la figura 15.3.

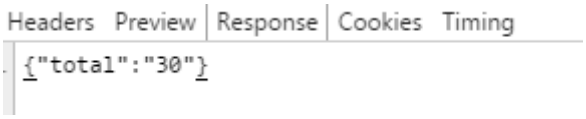


Figura 15.3 Respuesta web service (Actualización puntuación)

Como vemos, únicamente devuelve la puntuación total, para que, una vez haya sido recibida en el navegador del usuario, esta sea mostrada de forma automática.

15.3 jQuery

En la parte de presentación, como ya hemos indicado, se ha utilizado JavaScript, sin embargo, cabe destacar que hemos hecho uso del framework jQuery.

Este framework nos facilita mucho la tarea, ya que se puede obtener más funcionalidad con menos líneas si se usa jQuery comparado con la utilización nativa de JavaScript.

Además jQuery dispone de toda una comunidad de desarrolladores, que implementan diferentes plugins para este framework. Estos plugins se incluyen en las páginas en las que se deseen utilizar y ofrecen una amplia gama de funcionalidades, que abarcan desde la creación automática de galerías de imágenes, hasta la personalización de los menús contextuales.

En nuestro caso hemos utilizado varios de estos plugins que comentaremos en el siguiente capítulo.

15.1 FPDF

Otro de los retos que se nos han planteado en la fase de implementación ha sido que tecnología utilizar para generar informes de forma automática de las partidas del torneo FLL.

Tras documentarnos sobre varias, hemos elegido FPDF.

FPDF constituye una fenomenal herramienta para desarrolladores web con la que podemos programar impresiones de documentos en formato PDF.

Representa una clase escrita en PHP, cuya función es facilitar la creación de documentos en este lenguaje de programación

que den como resultado un documento en PDF.

FPDF permite, entre otras cosas, elegir la unidad de medida, el formato de página y los márgenes; gestionar la cabecera y el pie de página; establecer saltos de líneas y justificar textos de forma automática.

Por otro lado, admite imágenes en formato JPEG y PNG; y soporta los tipos de fuente TrueType y Type1.

Capítulo 16. Plugins jQuery utilizados

En este capítulo vamos a describir brevemente los plugins para jQuery que hemos utilizado y con qué propósito se ha hecho.

16.1 Plugin Counter

Este plugin nos permitirá crear una cuenta regresiva muy fácilmente con solo jQuery.

Para ello muestra un reloj estilo antiguo con paletas que cambian sus números con el paso del tiempo. Viene con una variedad de opciones y permite su uso básico y avanzado dependiendo lo que se necesite.

La forma de utilizarlo es la siguiente:

```
$('#counter').countdown({
stepTime: 60,
format: 'hh:mm:ss',
startTime: "12:32:55",
digitImages: 6,
digitWidth: 53,
digitHeight: 77,
timerEnd: function() { alert('end!!'); },
image: "digits.png"
});
```

Como vemos, con este plugin podemos especificar el tiempo de actualización de la cuenta atrás, el formato, el tiempo de inicio, las dimensiones de los dígitos e incluso podemos indicarle que ejecute cierta función al terminar la cuenta atrás. Esto es muy útil en nuestro caso, ya que queremos que una vez haya llegado la cuenta atrás a 0, empiece a contar los dos minutos y medio que dura una partida FLL.

En la figura 16.1 vemos un ejemplo de este plugin en funcionamiento.

Figura 16.1 Plugin counter

16.2 Plugin FancyBox

FancyBox es una de los mejores plugins para abrir en un popup imágenes, contenido HTML y multimedia al estilo "lightbox" flotante sobre la página web.

En nuestro caso, lo hemos utilizado para que el jefe de árbitros pueda modificar la puntuación de una partida sin necesidad de salir de la página actual, simplemente cargando un div y superponiéndolo al resto de la página con ayuda de este plugin para jQuery.

Para utilizar este plugin lo primero que hemos hecho es descargarlo dese la página oficial de FancyBox.

En la carpeta "source" tenemos los estilos y scripts del plugin además de las imágenes que emplea. Esa carpeta debemos incorporarla a nuestro proyecto para poder utilizarlo.

Invocamos a la librería de la siguiente forma:

```
<script type="text/javascript" src="fancybox/jquery.fancybox.js?v=2.1.5"></script>
```

Seguidamente crearemos nuestro contenedor que hará de popup con el contenido que deseemos, en nuestro ejemplo:

```
<div id="popup" style="display: none; width: 560px;">
    <h2>Contenido de fancybox</h2>
    Ejemplo de contenido
</div>
```

Y finalmente deberemos incluir antes de la etiqueta de cierre </body> o dentro de la cabecera <head> la llamada al fancybox y utilizar la función trigger() de jQuery para lanzar el evento.

```
<script>
    $(document).ready(function () {
      $(".fancybox").fancybox();
      $(".fancybox").trigger('click');
    });
</script>
```

En la figura 16.2, podemos ver un ejemplo del plugin fancybox en funcionamiento.

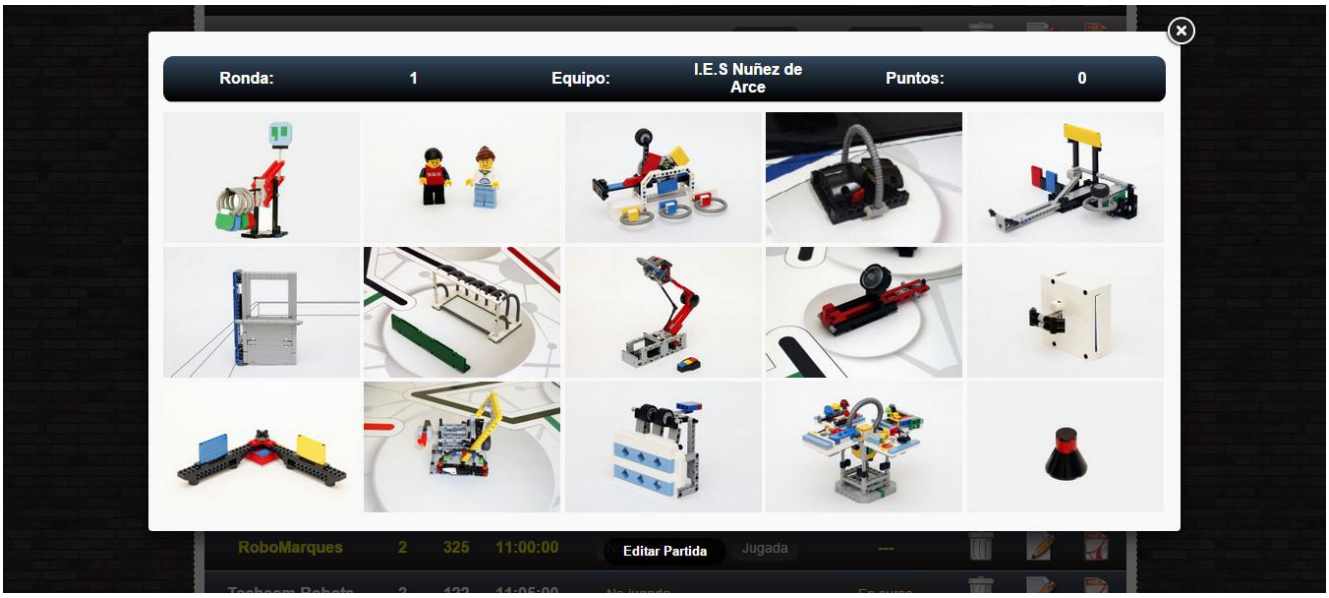


Figura 16.2 Plugin FancyBox

Capítulo 17. Estructura de directorios

En este capítulo vamos a describir como hemos estructurado los directorios de la página web. La organización de los mismos se ha llevado a cabo intentando mantener una organización lógica, a su vez, se ha hecho teniendo en cuenta los módulos que definimos en la etapa de análisis.

En la figura 17.1 podemos ver esta estructura.

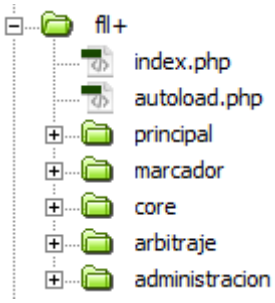


Figura 17.1 Estructura de directorios general

La función que realiza el fichero index.php es incluir el fichero autoload.php, que a su vez, incluye todas las clases del modelo de dominio y los datamappers. Index.php también incluye al controlador frontal e invoca a su método main(), delegando en él el procesamiento de la petición del usuario.

A continuación pasamos a describir el contenido de cada uno de los subdirectorios y la funcionalidad que esta implementada dentro de cada uno.

13.1 Marcador

En la figura 17.2, podemos ver los ficheros y directorios incluidos dentro de la carpeta Marcador. Aquí se encuentran las clases que ofrecen la funcionalidad del marcador dentro del sistema.

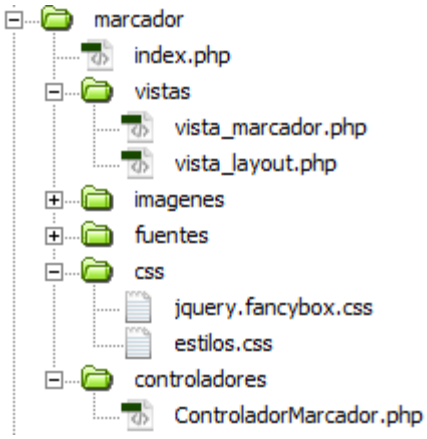


Figura 17.2 Estructura de directorios del marcador

-El fichero “index.php”, del mismo modo que ocurría en el nivel superior, se encarga de incluir el controlador frontal y de invocar al método “main()”, delegando así, la petición del usuario a este controlador.

-El subdirectorio vistas contiene los ficheros “vista_marcador.php”, que contiene los datos necesarios para poder mostrar al usuario la vista del marcador. También contiene el fichero “vista_layout.php” que contiene el esqueleto de la vista del marcador.

-El directorio imágenes contiene las imágenes del marcador, tales como el logotipo de Lego entre otras.

-El directorio “fuentes”, contiene la fuente de Lego en varios formatos, para poder mostrar los textos del marcador con la fuente oficial de Lego.

-El directorio “css”, contiene los ficheros de estilo que maquetan el marcador.

-Por último el directorio “controladores”, contiene el controlador del marcador, que como vimos en el bloque de diseño es el encargado de gestionar las peticiones relacionadas con el marcador.

17.2 Principal

En la figura 17.3 podemos ver los ficheros y directorios contenidos en la carpeta “principal”. Este directorio contiene las clases que ofrecen el acceso al sistema y muestran al usuario las acciones que puede realizar en función de los permisos que tenga otorgados en la base de datos.

Figura 17.3 Estructura de directorios módulo principal

-El fichero “index.php” se encarga de llamar al controlador frontal para delegar en él la gestión de la petición del usuario.

-En la carpeta vista encontramos los interfaces de acceso a la aplicación y el menú de bienvenida.

-El directorio “imágenes” contiene los logos de FLL entre otras cosas.

-En la carpeta “css” se encuentran definidos los diferentes estilos para maquetar este apartado del sistema.

-Por último, en la carpeta “Controladores” se encuentra la clase “ControladorUsuario” encargada principalmente de gestionar el acceso de los usuarios al sistema.

17.3 Arbitraje

En la figura 17.4 se muestra la estructura de ficheros y directorios existentes en la carpeta arbitraje.

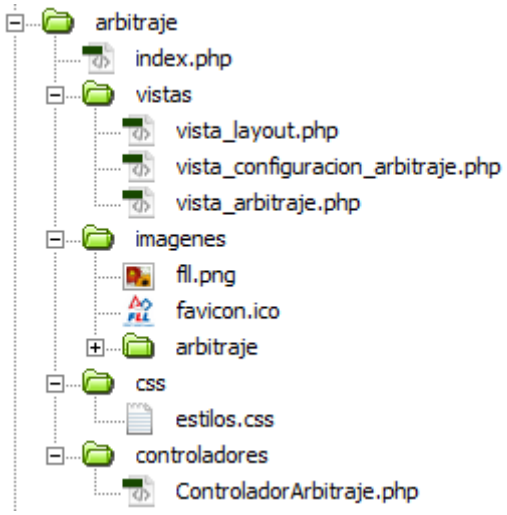


Figura 17.4 Estructura de directorios módulo arbitraje

En este directorio se encuentran todos los ficheros encargados de la parte de arbitraje, más en concreto, aquí podemos encontrar:

-El fichero “index.php” se encarga de incluir el controlador frontal e invocar a su método “main()” delegando en él, la petición realizada por el usuario.

-La carpeta “vistas” contiene todas las vistas relacionadas con el arbitraje, como pueden ser, la vista de configuración de arbitraje, en la que el árbitro escoge la partida que va a arbitrar o la propia vista de arbitraje.

-En la carpeta imágenes se encuentran las imágenes del módulo de arbitraje. Aquí, entre otras, podemos encontrar las todas las imágenes de cada una de las pruebas de las que consta un torneo FLL.

-El directorio “css” contiene el fichero que maqueta esta parte de la web.

-Por último en el directorio “controladores” encontramos la clase “ControladorArbitraje” encargada de gestionar las peticiones relacionadas con el arbitraje en el sistema.

17.4 Administración

En la figura 17.5 observamos la estructura de directorios de la parte de administración del sistema FLL+.

Este directorio contiene todas las clases relacionadas con la parte de administración del sistema. Son, por tanto, las que ofrecen toda la funcionalidad al actor jefe de árbitros.

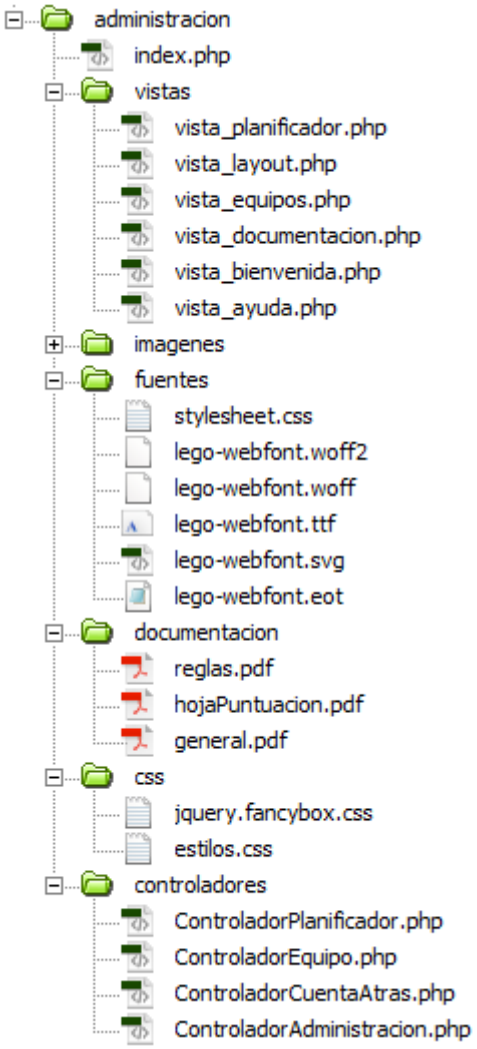


Figura 17.5 Estructura de directorios módulo administración

-El fichero "index.php", como en los casos anteriores, se encarga de incluir el controlador frontal en el cual delegará la gestión de la petición del usuario.

-En la carpeta vista encontramos la vista de la parte de administración, como por ejemplo, la vista del planificador de partidas o la vista de gestión de equipos.

-En la carpeta imágenes encontramos las imágenes utilizadas en la administración, como pueden ser, los iconos de modificar, eliminar o insertar partidas.

-En la carpeta fuentes esta la fuente oficial de Lego en varios formatos para asegurar que la web se visualiza correctamente en distintos navegadores web.

-En el directorio "documentación" se encuentran los ficheros PDF con la documentación oficial de la FLL.

-En la carpeta "css" se encuentran los estilos que maquetan la parte de administración del sistema FLL+.

-Por último en la carpeta "controladores" encontramos los controladores encargados de gestionar los diferentes tipos de peticiones por parte de los usuarios.

17.1 Core

En este directorio encontramos el núcleo del sistema FLL+. El esquema de directorios del core se muestra en la figura 17.6.

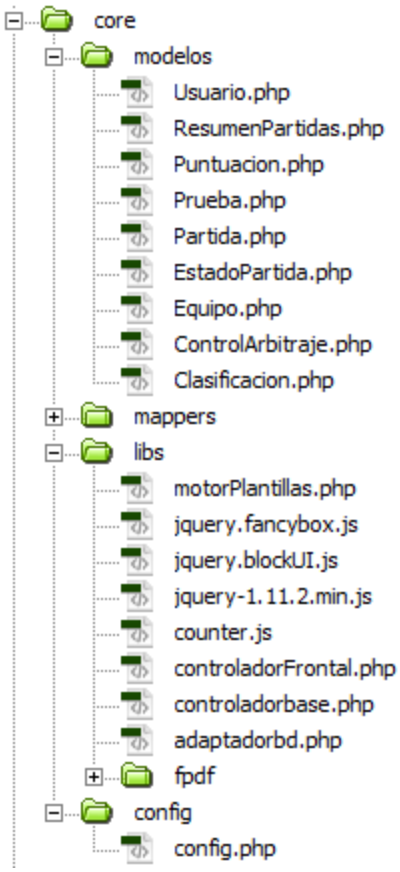


Figura 17.6 Estructura de directorios del core

-En el directorio “modelos” encontramos todas las clases de la lógica de la aplicación.

-En el directorio “mappers” se encuentran todas las clases datamapper de la aplicación.

-En la carpeta “libs” encontramos algunas clases de propósito general, como el controlador frontal, la clase “Vista”, las librerías de los plugins utilizados o las clases de FPDF, que como hemos mencionado anteriormente , es el framework que hemos utilizado para la generación automática de PDFS.

-Por último, en la capeta “config” está ubicado el fichero de configuración de FLL+, donde se definen configuraciones como el nombre de usuario y contraseña de acceso al servidor de bases de datos o el número de pruebas de las que consta el torneo.

Bloque V. Pruebas y resultados

Capítulo 18. Pruebas

18.1 Introducción

En este bloque vamos a explicar las pruebas que hemos realizado al sistema una vez lo hemos implementado por completo. Estas pruebas nos ayudarán a conocer si hemos alcanzado los objetivos que nos hemos marcado en etapas anteriores y a comprobar que el sistema tiene la funcionalidad esperada.

A grandes rasgos, las pruebas que hemos realizado sobre el sistema las podemos clasificar en dos tipos:

Pruebas de funcionalidad Estas pruebas están orientadas a conocer si el sistema hace lo que debería hacer y como lo debería hacer.

Pruebas de rendimiento Las pruebas de rendimiento nos ayudarán a conocer cómo se comporta el sistema cuando el nivel de peticiones por parte de los usuarios crece, haciendo así, una emulación de la carga real que podría tener el sistema en producción.

A continuación pasamos a detallar estos dos tipos de pruebas.

18.2 Pruebas de funcionalidad

Para llevar a cabo las pruebas de funcionalidad, hemos utilizado dos ordenadores. Desde uno, realizamos diferentes acciones, mientras que desde el otro comprobamos que el sistema se está comportando como debería hacerlo en tiempo real. Por ejemplo, desde el primero actualizamos una puntuación y desde el segundo visualizamos el marcador, comprobando que los resultados obtenidos son los esperados.

A continuación, pasamos a enumerar los pasos que hemos ido probando y los resultados que hemos ido obteniendo en cada uno de ellos:

-Actualización de la puntuación de una partida desde el panel de administración.

La puntuación se almacena correctamente y el marcador muestra de forma automática el resultado actualizado.

-Planificación de una partida indicando el equipo, la ronda y la hora.

El sistema registra la partida en la base de datos y desde el panel del marcador se muestra una nueva ronda con el estado “No jugada”.

-Inserción de un nuevo equipo, indicando su nombre y descripción.

El sistema almacena el nuevo equipo, y este, es mostrado en el marcador.

-Exportación de una partida con puntuaciones cargadas.

El sistema devuelve un fichero en formato PDF con la información correctamente actualizada.

-Iniciar y parar la cuenta atrás desde el panel de administración.

El marcador comienza y se detiene de forma correcta.

-Marca una partida con el estado “En curso”.

El sistema cambia el color del registro donde se muestra la partida y desde el panel de selección de partida utilizado por los árbitros se muestra una nueva partida lista para ser arbitrada.

-Iniciar el arbitraje de una partida.

El sistema asigna un token al árbitro correspondiente y elimina automáticamente la partida del panel de selección de partidas listas para ser arbitradas.

-Iniciar el arbitraje de una partida y posteriormente eliminar el acceso al árbitro que inicio este arbitraje.

El sistema no deja seguir arbitrando informando al árbitro de ello.

18.3 Pruebas de rendimiento

El objetivo de las pruebas de rendimiento es ver cómo se comporta el sistema emulando las cargas que tendría el sistema en producción, es decir, lo que queremos saber es si el sistema responde a las peticiones de los usuarios en un tiempo razonable cuando la concurrencia de las peticiones se eleva hasta el punto que podría alcanzar cuando se haga un uso real de la aplicación.

Para llevar a cabo estas pruebas, hemos usado la herramienta “Apache benchmark” ya que es una herramienta que permite hacer pruebas de carga a servidores web Apache.

Con “Apache benchmark” podemos lanzar las peticiones que queramos sobre la URL que deseemos con el nivel de concurrencia que le indiquemos. Por ejemplo, si queremos lanzar 10 peticiones de 2 en 2 (nivel de concurrencia 2), sobre la URL “vps150302.ovh.net/arbitraje/principal/index.php” debemos escribir el comando que se muestra en la figura 18.1.

```
C:\xampp\apache\bin>ab -n 10 -c 2 http://vps150302.ovh.net/arbitraje/principal/i
ndex.php
This is ApacheBench, Version 2.3 <$Revision: 1604373 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Figura 18.1 Ejemplo de uso del comando “ab”

18.3.1 Definición del sistema donde se han hecho las pruebas

Como es lógico, antes de mostrar los resultados obtenidos debemos explicar las características de la máquina sobre la que se han hecho las pruebas, ya que esta variable es de vital importancia. Un equipo con mejores prestaciones obtendrá mejores resultados y viceversa.

La máquina sobre la que se han hecho las pruebas, es una máquina virtual alquilada personalmente por el alumno a la empresa de hosting OVH. Las características de esta máquina virtual son las mostradas en las figuras 18.2 y 18.3.

```
root@vps150302:/proc# cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 16
model           : 2
model name      : AMD Opteron(tm) Processor 4284
stepping        : 3
cpu MHz         : 2999.999
cache size      : 2048 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 5
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp
 lm constant_tsc rep_good tsc_reliable nonstop_tsc unfair_spinlock pni cx16 x2ap
ic popcnt hypervisor lahf_lm cmp_legacy extapic cr8_legacy abm sse4a misalignsse
 3dnowprefetch osvw
bogomips        : 5999.99
TLB size        : 1536 4K pages
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:
```

Figura 18.2 Procesador de la máquina sometida a pruebas

```
root@vps150302:/proc# cat /proc/meminfo
MemTotal:        1048576 kB
MemFree:          849296 kB
Cached:            56868 kB
Buffers:               0 kB
Active:            76020 kB
Inactive:         102640 kB
Active(anon):      54372 kB
Inactive(anon):    70604 kB
Active(file):      21648 kB
Inactive(file):    32036 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:        131072 kB
SwapFree:          71752 kB
Dirty:                24 kB
Writeback:             0 kB
AnonPages:        124976 kB
Shmem:              3184 kB
Slab:              20600 kB
SReclaimable:       6088 kB
SUnreclaim:        14512 kB
```

Figura 18.3 Memoria RAM de la máquina sometida a pruebas

Como vemos, se trata de una maquina con un procesador de 2 cores a 3GHz y 1GB de memoria RAM sobre el que hay instalado un sistema operativo Debian.

Esta máquina no tiene unas prestaciones especialmente buenas, de hecho, son más bien mediocres, por ello, si las pruebas que haremos a continuación dan buenos resultados, implicará que en un servidor de gama superior, los usuarios de la aplicación podrían experimentar unos resultados por parte del sistema mejores.

18.3.2 Determinación de las métricas

Las métricas se centrarán en definir el rendimiento del servidor web en cuanto al servicio de respuesta de peticiones se refiere. Se clasificarán las métricas siguiendo un esquema:

Petición correctamente servida

Petición incorrectamente servida (error)

Petición no servida de ninguna forma (fallo, caída del servidor web)

Se asignarán un conjunto de métricas para cada uno de estos tipos. Para observar el rendimiento ofrecido al servir peticiones correctamente, interesa medir la velocidad en la que se sirven: el tiempo en el que se sirven las peticiones (tiempo de respuesta), esto es, el tiempo transcurrido desde que se envía la petición hasta que se recibe (no interesa el tiempo que emplea el navegador en interpretar el código HTML devuelto por el servidor, ya que deja de ser competencia del sistema).

Por otra parte, interesa conocer la productividad al servir un número elevado de peticiones (número de peticiones servidas por unidad de tiempo, datos transferidos por unidad de tiempo).

El último campo de estudio de rendimiento será la disponibilidad, esto es, el porcentaje de situaciones en la que la petición no ha sido resuelta de ninguna forma por caída o fallo del sistema.

En este campo interesa conocer el tiempo que lleva volver a poner el servidor en funcionamiento, o el porcentaje de tiempo que se estima que el servidor esté disponible en un determinado rango de tiempo.

En la figura 18.4 se resumen las métricas escogidas para evaluar el rendimiento del sistema:

Tipo	Nombre	Unidad de medida
Métricas de velocidad	Tiempo de respuesta	Segundo
	Peticiones servidas por unidad de tiempo	Peticiones/segundo
	Throughput	Bytes/segundo
	Utilización	%uso de los recursos del sistema
Métricas de fiabilidad	Tasa de error	% de errores
Métricas de disponibilidad	Tasa de fallo	% de fallos

Figura 18.4 Métricas escogidas

18.3.3 Justificación de las métricas escogidas

Tiempo de respuesta medio (s). El tiempo de respuesta medio es una métrica muy significativa, puesto que es el periodo de tiempo que podría esperar un cliente desde que lanza la petición hasta que obtiene la respuesta por parte del servidor. Un tiempo de respuesta aceptable evita que los usuarios abandonen el sitio web mientras que un tiempo de respuesta excesivo provoca, en general, que los usuarios se desesperen y abandonen la página.

Peticiones servidas por unidad de tiempo (peticiones/s). Hemos elegido esta métrica puesto que representa el rendimiento que es capaz de ofrecer el sistema, es decir, cuantas peticiones puede resolver por unidad de tiempo.

Productividad (B/s). Esta métrica representa otro punto de vista del rendimiento del sistema, muestra la cantidad de información que es capaz de generar y transmitir a los clientes. Resumiendo, es la cantidad de datos que puede servir.

Tasa de error. Hemos elegido esta métrica por que representa el número de transacciones erróneas entre el número total de transacciones, es decir, esta métrica representa la fiabilidad de nuestra sistema y la probabilidad que podríamos esperar de que una transacción devuelva una respuesta errónea.

Tasa de fallo. Esta métrica muestra el número de transacciones fallidas entre el número total de transacciones, es decir, esta métrica representa la disponibilidad que podemos esperar de nuestro sistema, por ello se ha elegido como métrica representativa de la disponibilidad del sistema.

18.1 Caracterización de la carga

La caracterización de la carga consiste en definir qué recursos del sistema serán solicitados y con qué frecuencia, es decir, la idea de la caracterización de la carga es describir con qué nivel de trabajo se va a topar nuestro servidor en una situación real.

Normalmente, la forma de obtener la caracterización de la carga se realiza mediante la monitorización de las peticiones recibidas durante un periodo de tiempo, de tal forma, que después de esa monitorización podamos saber qué porcentaje de peticiones recaen sobre qué tipos de ficheros o URLs.

En nuestro caso, esta monitorización no ha sido posible, sin embargo, tanto el tutor del proyecto como yo sabemos bien cómo funcionan los eventos FLL ya que hemos participado activamente en la organización de los mismos.

Podemos llegar a la conclusión de que las peticiones realizadas por los árbitros y el/los jefe/jefes de árbitros son despreciables ya que son acciones poco frecuentes en comparación con las realizadas por los espectadores. De la misma forma, las peticiones sobre ficheros que no sean PHP también son despreciables, ya que la web cuenta con muy pocos contenidos de otro tipo en el módulo del marcador.

Para recalcar el hecho de que las acciones de los árbitros y el jefe de árbitros son despreciables pondremos un ejemplo:

Por cada vez que un árbitro actualiza una puntuación de una partida o un jefe de árbitros cambia el estado de una partida, se efectúan decenas o incluso centenas de peticiones por parte de los espectadores para consultar el estado de la partida actual o la clasificación.

Por ello, nuestro estudio del rendimiento recaerá sobre las peticiones que hacen los espectadores, que son, ver la clasificación y ver el estado de la partida actual. Habitualmente, el número de espectadores suele ser de entre 100 y 200.

La implementación del marcador, está programada de tal forma que se consulte la clasificación mediante una petición Ajax cada 5 segundos y el estado de la partida actual cada 4 segundos.

En conclusión, nuestra caracterización de la carga se puede resumir así:

-Decenas de peticiones concurrentes sobre el web service que devuelve el estado de la partida actual.

-Decenas de peticiones concurrentes sobre el servicio web que devuelve el estado de la clasificación.

Capítulo 19. Resultados obtenidos

19.1 Resultados

Como ya hemos comentado vamos a lanzar cargas de trabajo con la ayuda de la herramienta Apache Benchmark previamente instalada, variando el nivel de concurrencia y el número de peticiones en cada una de las pruebas, a los dos web services del marcador, para ver de qué forma influye en el rendimiento del sistema.

La sintaxis del comando para lanzar peticiones es:

$ ab –n X –c Y http://vps150302.ovh.net/arbitraje

Donde X será el número de peticiones e Y el nivel de concurrencia (número de peticiones que se envían simultáneamente).

A continuación pasamos a mostrar los resultados obtenidos (figuras 19.1 y 19.2) para cada una de las cargas lanzadas a los servicios web del sistema FLL+:

Número de peticiones	Nivel de concurrencia	Tiempo de respuesta (ms)	Peticiones/segundo	Tasa de error	Tasa de fallo	Throughput (kB/s)
150	1	69,004	14,49	0	0	5,07
	2	77,764	25,72	0	0	7,93
	3	131,268	22,85	0	0	10,49
	4	145,635	27,47	0	0	10,41
	5	224,746	22,25	0	0	10,21
	10	368,888	27,11	0	0	12,44
	15	513,029	29,24	0	0	13,42
	20	894,318	22,36	0	0	10,26
	25	902,385	27,7	0	0	12,72
	50	1.832,438	27,29	0	0	12,52
	100	3.876,221	25,8	0	0	11,84
	150	5.651,323	26,54	0	0	12,18

Figura 19.1 Resultados web service (Partida actual)

Número de peticiones	Nivel de concurrencia	Tiempo de respuesta (ms)	Peticiones/segundo	Tasa de error	Tasa de fallo	Throughput (kB/s)
150	1	98,919	10,11	0	0	33,14
	2	101,846	19,64	0	0	64,38
	3	107,226	27,98	0	0	91,72
	4	157,076	25,47	0	0	83,48
	5	208,712	23,96	0	0	78,54
	10	369,421	27,07	0	0	88,74
	15	556,432	26,96	0	0	88,38
	20	741,242	26,98	0	0	88,45
	25	964,722	25,91	0	0	84,96
	50	1.942,444	25,74	0	0	84,39
	100	4.234,243	23,62	0	0	77,42
	150	6.799,389	22,06	0	0	72,32

Figura 19.2 Resultados web service (Clasificación)

En las siguientes figuras vemos los datos de las tablas 19.1 y 19.2 en forma de gráfico.

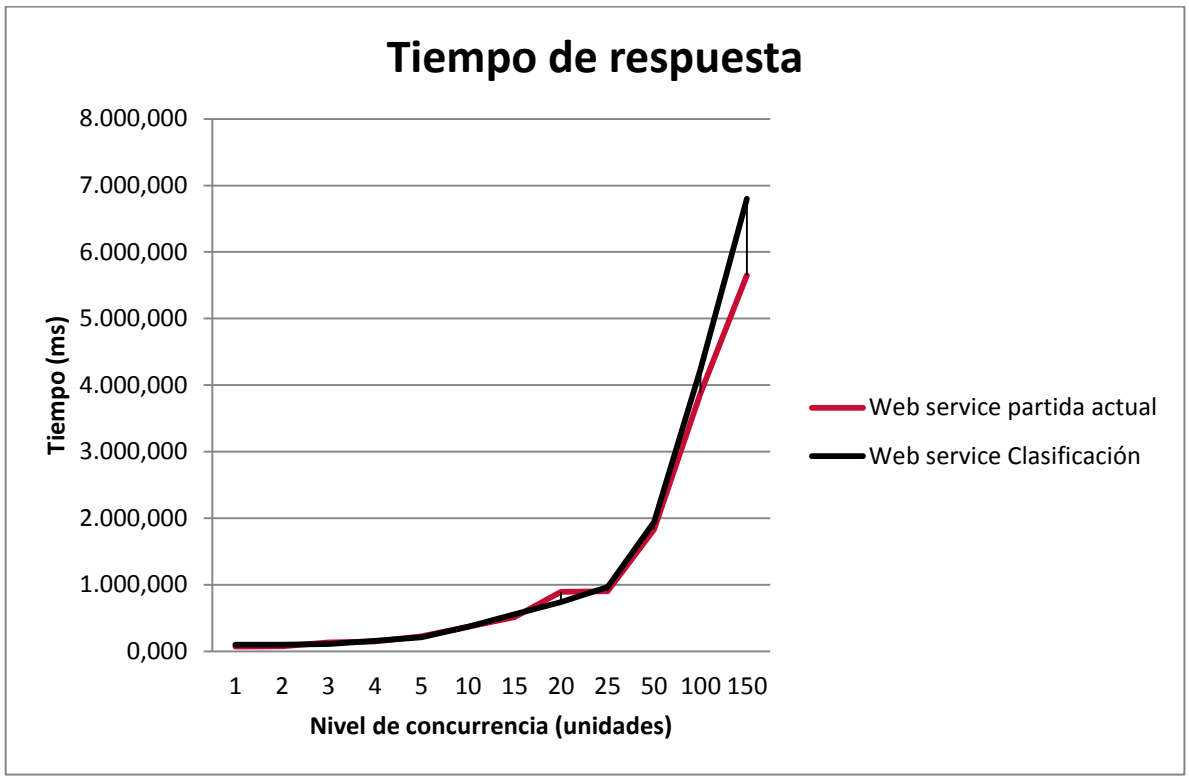


Figura 19.3 Tiempo de respuesta

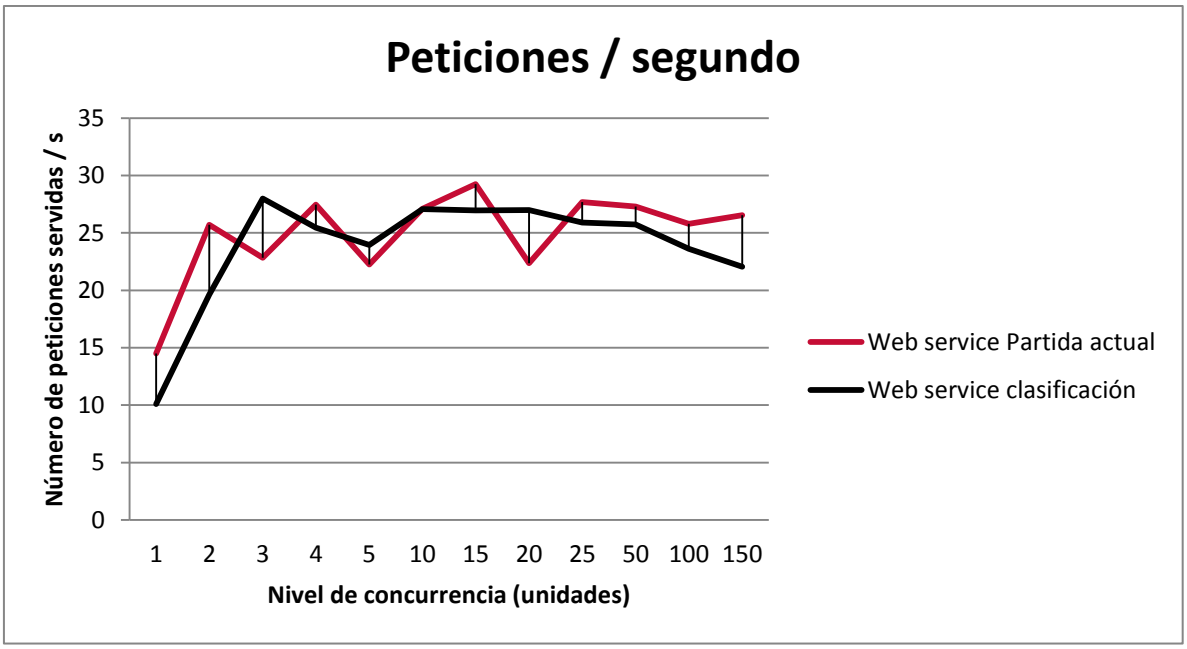


Figura 19.4 Peticiones/segundo

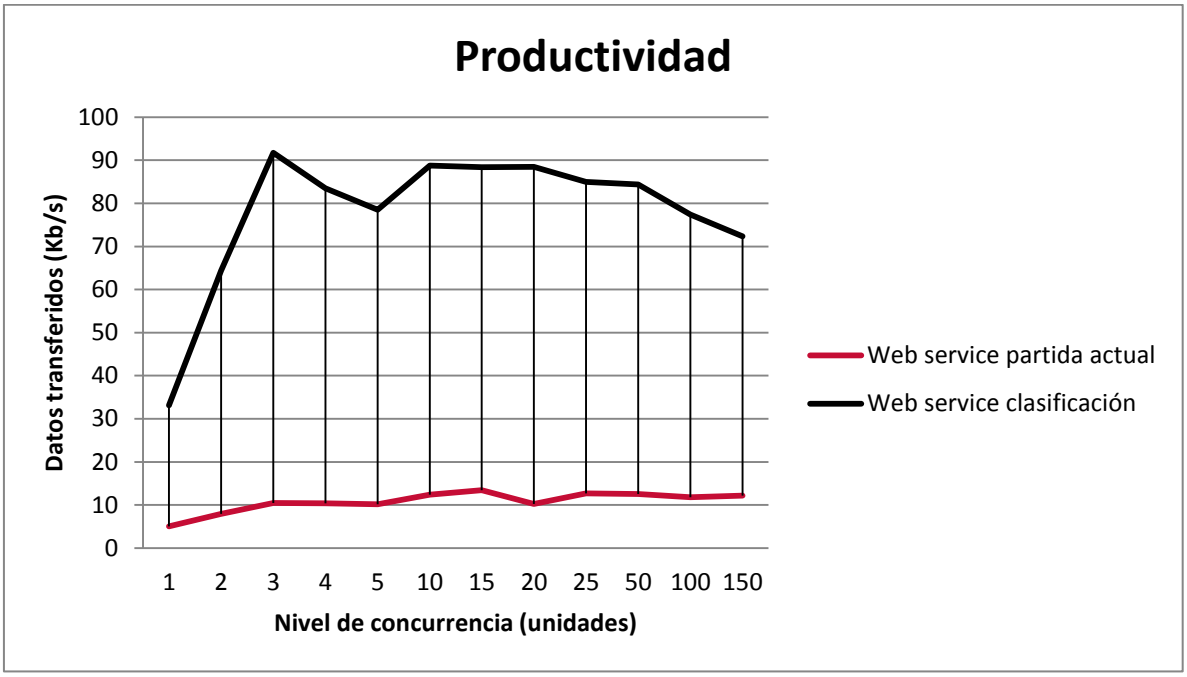


Figura 19.5 Productividad

19.2 Análisis de los resultados

19.2.1 Tiempo de respuesta

Para la métrica de tiempo de respuesta en función del número de peticiones simultáneas, se observa como el tiempo de respuesta crece al aumentar el nivel de concurrencia. Entre el dato para concurrencia 25 y 150, se aprecia un ligero cambio de pendiente, lo que quiere decir que el tiempo de respuesta aumenta conforme aumenta el tiempo de concurrencia creciendo de forma más rápida el tiempo de respuesta que la concurrencia. También podemos observar que el tiempo de respuesta de los dos web services es muy similar

Como sabemos que acuden habitualmente entre 100 y 200 personas a un evento FLL y suponiendo que una de cada dos personas utilice un dispositivo para conectarse al servidor y que las peticiones a los web services se realizan 1 vez cada 5 segundos para el caso de la clasificación y 1 vez cada 4 segundos para el caso de la partida actual, obtenemos que el servidor recibirá 100/4 = 25 peticiones por segundos para el web service de la partida actual y 100/5 = 20 peticiones por segundo para el servicio web de la clasificación.

Si observamos la figura 19.3 vemos que para 25 y 20 peticiones concurrentes el servidor es capaz de responder en algo menos de un segundo, lo cual es un tiempo de respuesta más que aceptable y que producirá una buena experiencia de usuario para los espectadores. Si a esto le sumamos que las prestaciones del servidor donde se han realizado las pruebas son medias-bajas el dato obtenido es más positivo aún, ya que utilizando un servidor más potente tanto en velocidad de CPU como de acceso a disco podríamos experimentar tiempos aún más reducidos.

19.2.2 Peticiones por segundo

En el caso de las peticiones por segundo (Figura 19.4) se aprecia que entre el valor de concurrencia 1 y 2 hay un aumento notable de las peticiones que se sirven por unidad de tiempo, y que este valor va oscilando manteniéndose de una forma más o menos estable. Esto es porque el servidor dispone de dos núcleos de procesamiento, con lo cual es posible servir dos peticiones al mismo tiempo y así sacar más rendimiento a la máquina.

19.1 Productividad

Respecto a la productividad, podemos ver en la figura 19.5 que el servicios web de clasificación produce más Kb/s dado que el JSON que devuelve es más largo que en el caso del servicio web de clasificación.

Estos niveles de Kb/s son muy pequeños si tenemos en cuenta la capacidad de subida de una conexión moderna, por lo tanto, no sería necesario más de 1 Mbit/s de velocidad de subida para poder atender a las peticiones.

Bloque VI. Manual

Capítulo 20. Proceso de instalación

20.1 Requisitos previos a la instalación

Este software, como se ha indicado, está escrito en lenguaje PHP, se ejecuta bajo un servidor web Apache y requiere un servidor MySQL para poder almacenar de forma persistente los datos, por ello es necesario disponer de un sistema que al menos tenga instalado un servidor web, un intérprete de PHP y un sistema gestor de bases de datos MySQL.

Dado que Windows y Linux son dos de los sistemas operativos más utilizados, presentaremos a modo de ejemplo como instalar todos los elementos mencionados en el párrafo anterior tanto para Windows como para Linux (en su distribución Debian). Si ya se dispone de un servidor web, de un intérprete de PHP y de un sistema gestor de bases de datos no es necesario que siga leyendo este punto.

Existe una herramienta denominada XAMPP que incluye entre otras cosas Apache, PHP y MySQL. Para instalarla debemos visitar su página oficial y acceder al apartado de descargas. Una vez ahí seleccionamos la última versión y la descargamos.

20.1.1 Instalación de Xampp para Windows

En este apartado vamos a ver cómo realizar la instalación de Xampp para sistemas operativos Windows. Lo primero que debemos hacer es descargar el instalador de Xampp.

La URL de descarga es la siguiente:

https://www.apachefriends.org/es/download.html

Figura 20.1 Descarga XAMPP

Una vez descargado ejecutaremos el archivo con extensión .exe Nos aparecerá el siguiente menú:

Figura 20.2 Primer menú instalación XAMPP

Pulsamos en next y se mostrará el siguiente menú:

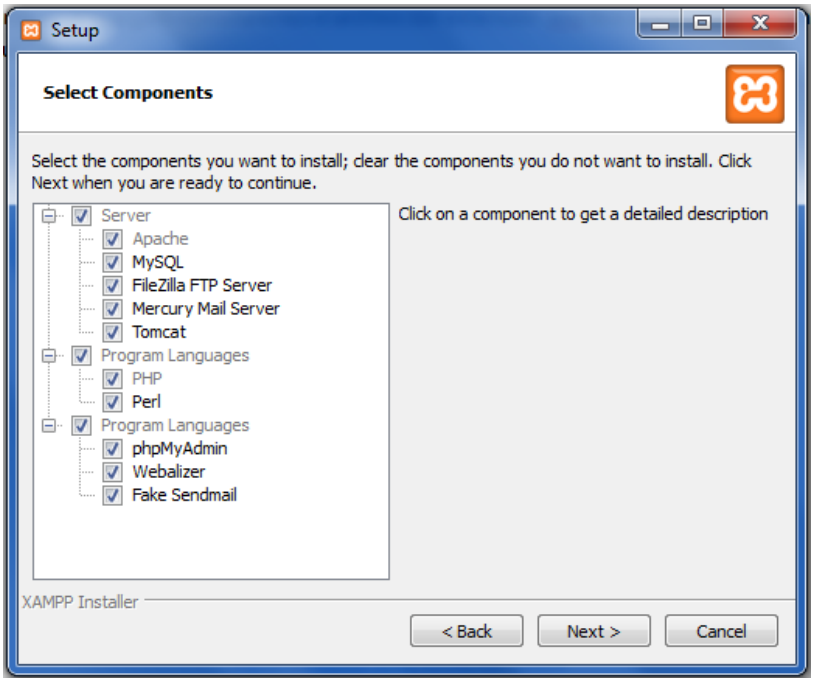


Figura 20.3 Segundo menú instalación XAMPP

Como vemos aparecen los distintos componentes que podemos instalar, los necesarios, como ya indicamos, son Apache, PHP y MySQL, sin embargo es recomendable marcar todos estos componentes para poder tener una mayor funcionalidad como por ejemplo poder trabajar con la base de datos desde phpMyAdmin, que es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web y que nos facilitará la tarea en pasos posteriores. A continuación pulsamos en el botón “next”. El siguiente menú que nos muestra sirve para indicar la ubicación donde se instalara XAMPP:

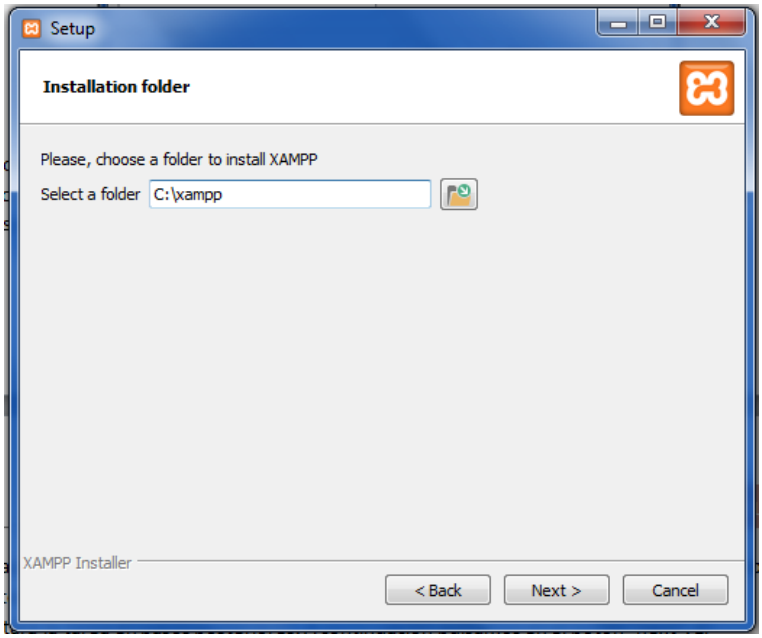


Figura 20.4 Tercer menú instalación XAMPP

Elegimos una ruta que esté vacía, es decir, que no contenga ningún fichero ni carpeta y pulsamos en el botón "Next".

En la figura siguiente vemos el menú que nos mostrará en el instalador. Aquí simplemente se nos informa de la posibilidad de instalar algunos de los gestores de contenidos más populares de forma automatizada como pueden ser joomla, wordpress o drupal. Como para nuestra instalación esto no nos interesa, desmarcamos la casilla "Learn more about Bitnami for Xampp.

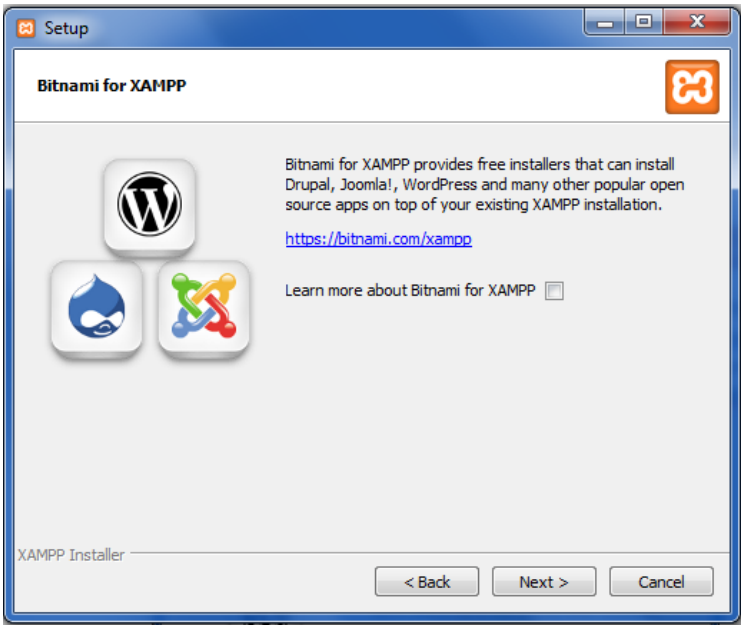


Figura 20.5 Cuarto menú instalación XAMPP

Pulsamos sobre el botón "Next" y el instalador nos mostrará el menú de la figura 20.6, indicándonos que ya está todo preparado para iniciar la instalación de Xampp.

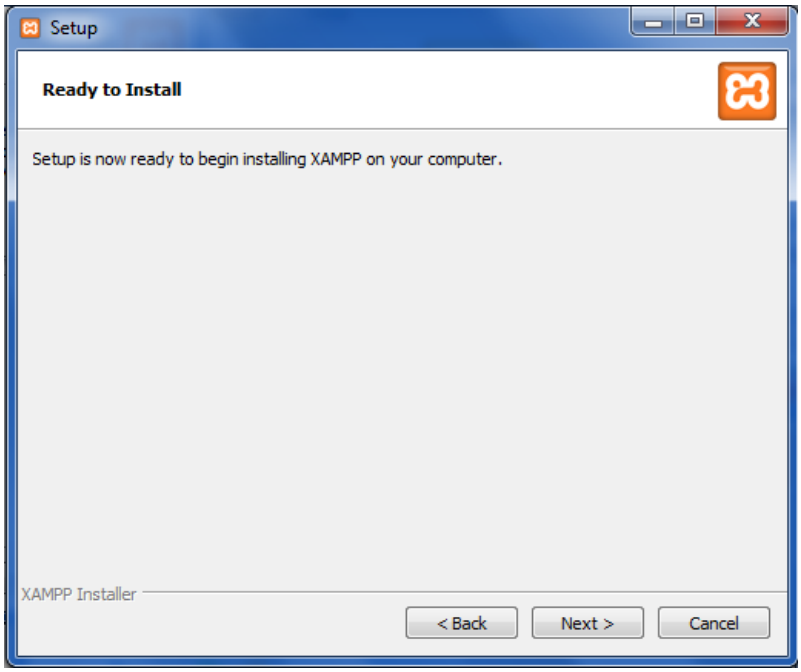


Figura 20.6 Quinto menú instalación XAMPP

Pulsamos una vez más en el botón "Next" y la instalación comenzará. Una vez completada ya dispondremos en nuestra máquina del servidor web Apache, el intérprete de PHP y el sistema gestor de base de datos MySQL.

Es importante destacar en este punto que a través del panel de control de Xampp podemos administrar todos los componentes mencionados en el párrafo anterior de una forma cómoda y sencilla. Algunas de las acciones que podemos realizar son parar, iniciar o reiniciar los servidores, y también, abrir los ficheros de configuración de cada uno de ellos para poder ajustar las opciones a nuestro gusto. En el caso del sistema FLL+ con la configuración que viene por defecto en Xampp es suficiente para que todo funcione correctamente.

Para asegurarnos de que todo está preparado debidamente, debemos abrir el panel de control de Xampp e iniciar el servidor Apache y MySQL pulsado en el botón "Start" en ambos.

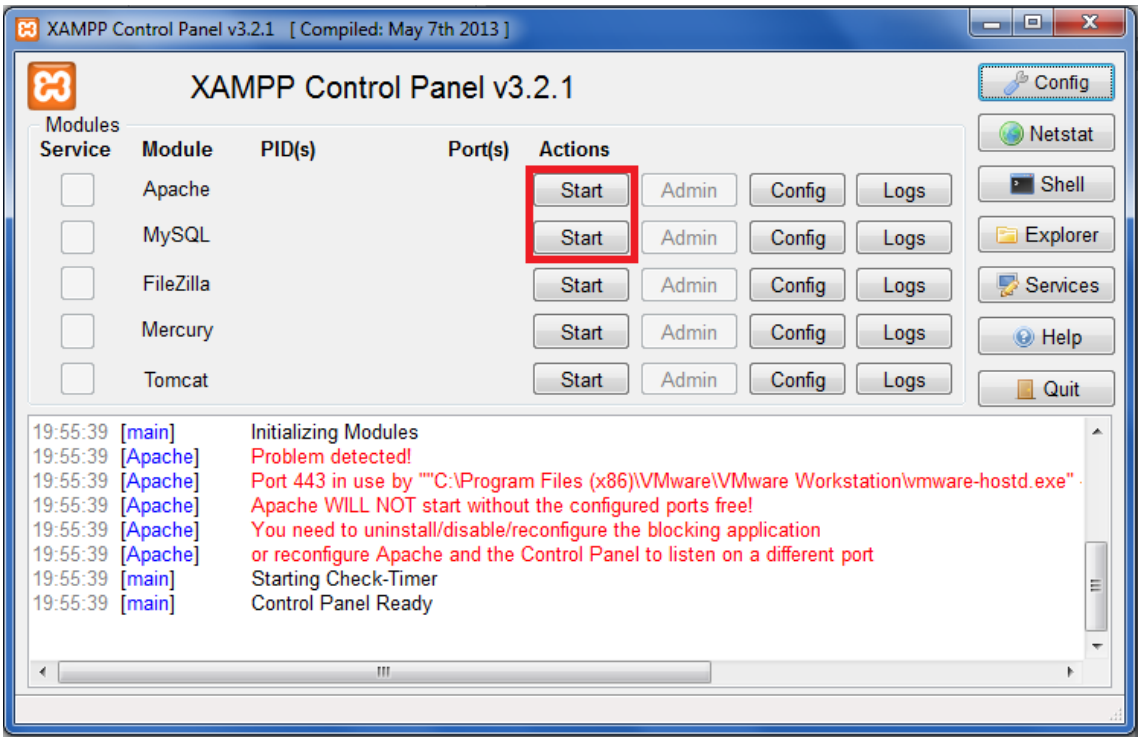


Figura 20.7 Panel de control XAMPP Windows

Una vez hemos iniciado estos dos servicios el panel de control deberá tener un aspecto como el que se muestra en la figura 20.8.

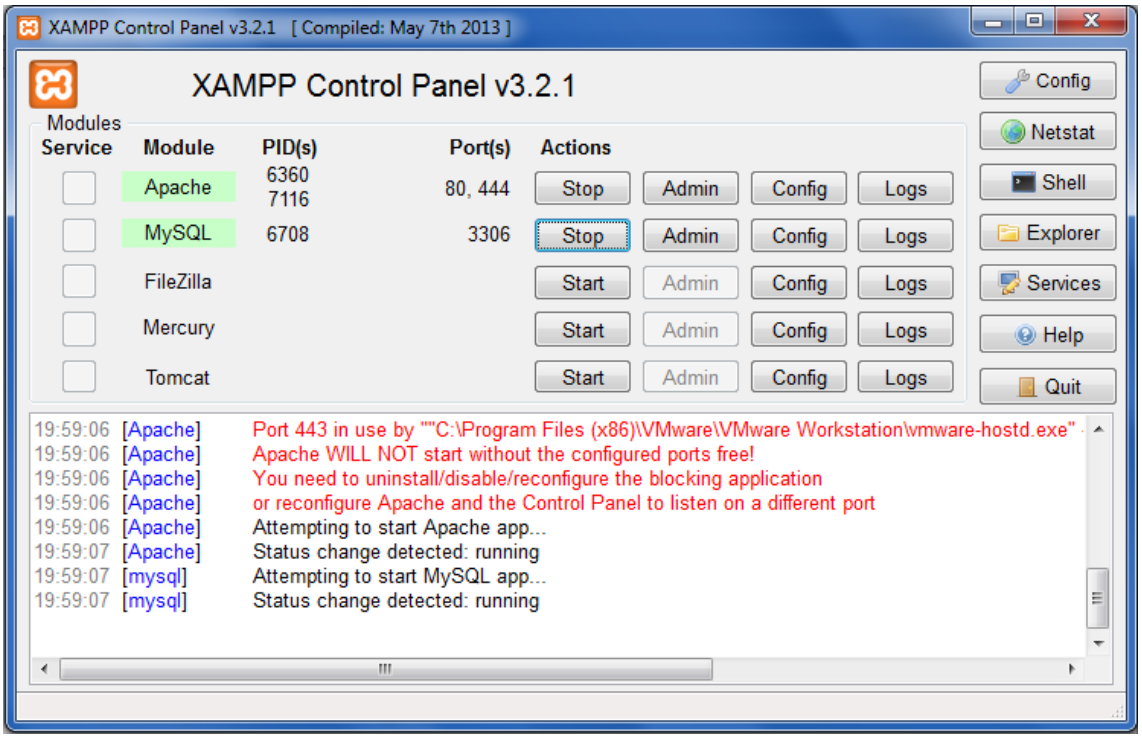


Figura 20.8 Panel de control XAMPP Windows (Servicios iniciados)

Llegado este punto podemos dar por concluidos los requisitos previos a la instalación de FLL+ para sistemas operativos Windows y estamos preparados para instalar el software FLL+.

20.1.2 Instalación de Xampp para Linux

En este apartado vamos a explicar cómo realizar la instalación para sistemas operativos Linux, más en concreto para la distribución Debian, ya que es una de las más utilizadas, sin embargo podemos extrapolar esta sección a cualquier distribución Linux ya que los pasos serán prácticamente los mismos. Lo primero que debemos hacer es descargar el instalador desde la página oficial de Xampp.

La URL de descarga es la siguiente:

https://www.apachefriends.org/es/download.html

Después debemos seleccionar el instalador de 32 o 64 bits en función del tipo de maquina en la que deseemos realizar la instalación. Si no sabemos qué tipo de maquina tenemos podemos ejecutar el comando "uname –a" tal y como se muestra en la figura 20.9 desde un terminal.

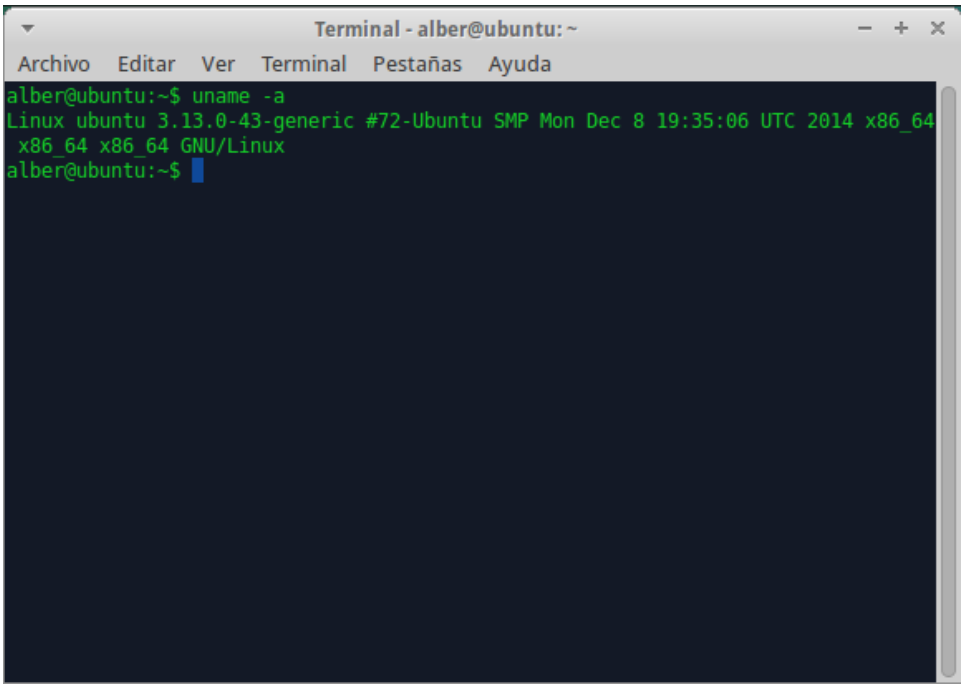


Figura 20.9 Comando uname –a Linux

Si en la salida del comando aparece "x86_64 GNU/Linux" estaremos frente a una máquina de 64 bits, sin embargo, si la salida es "i386/i486/i586/i686" la máquina será de 32 bits.

Una vez conozcamos los bits de nuestro procesador procederemos a descargar el instalador correspondiente.

El siguiente paso es cambiar los permisos al instalador que acabamos de descargar, a través del siguiente comando:

xampp-linux-x64-5.6.8-0-installer.run

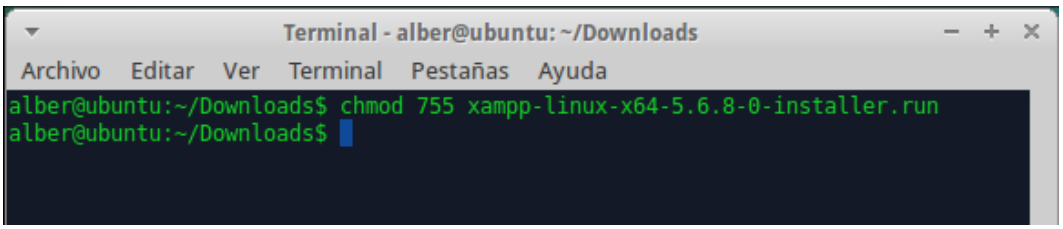


Figura 20.10 Cambiar permisos instalador XAMPP Linux

Después ejecutamos el instalador con el comando mostrado en la figura 20.11.

sudo ./ xampp-linux-x64-5.6.8-0-installer.run

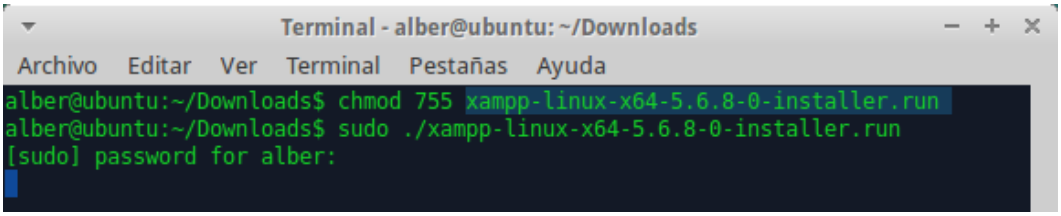


Figura 20.11 Comando ejecución instalador XAMPP Linux

Con la ejecución del comando aparecerá un menú con aspecto muy similar a los descritos en la sección de instalación de Xampp para Windows, por ello no indicaremos con recursos gráficos cada uno de los menús que nos irán apareciendo, tan solo mostraremos los más significativos. Lo único que debemos hacer es pulsar sucesivamente en el botón next hasta que comience el proceso de instalación tal y como se muestra en la figura 20.12.

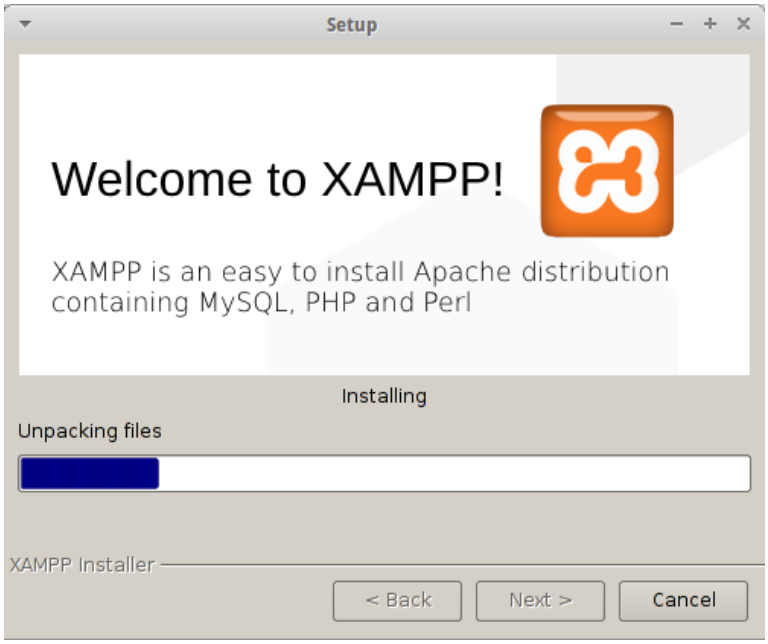


Figura 20.12 Progreso de instalación XAMPP Linux

Una vez haya concluido el proceso de instalación pulsamos sobre el botón “Finish” y el panel de control de Xampp se ejecutara de forma automática.

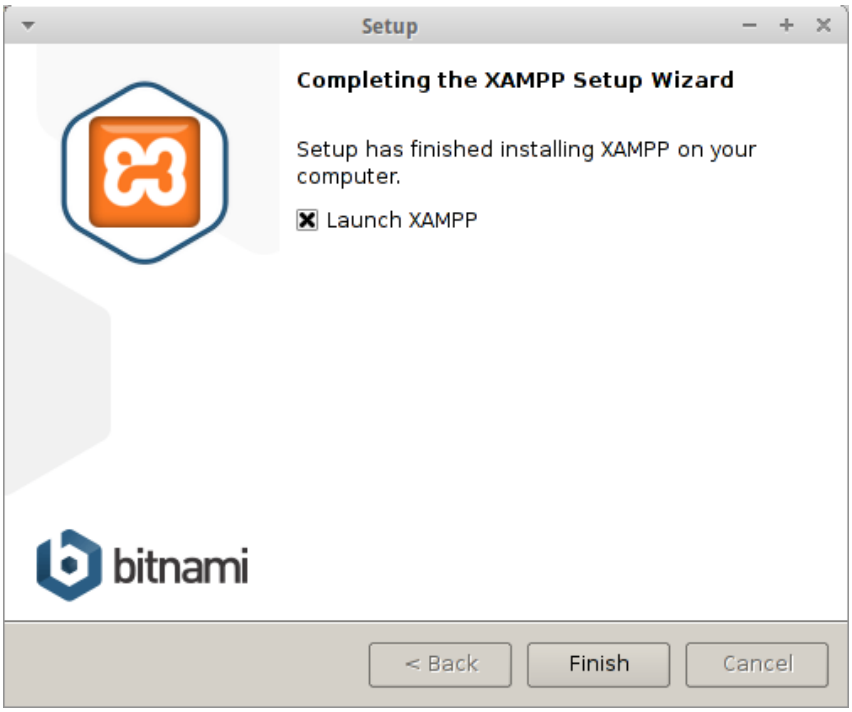


Figura 20.13 Ultimo paso instalación XAMPP linux

Una vez iniciado el panel de control, abrimos la pestaña "manage servers" e iniciamos Apache y MySQL. El aspecto que debe tener este panel se muestra en la figura 20.14.

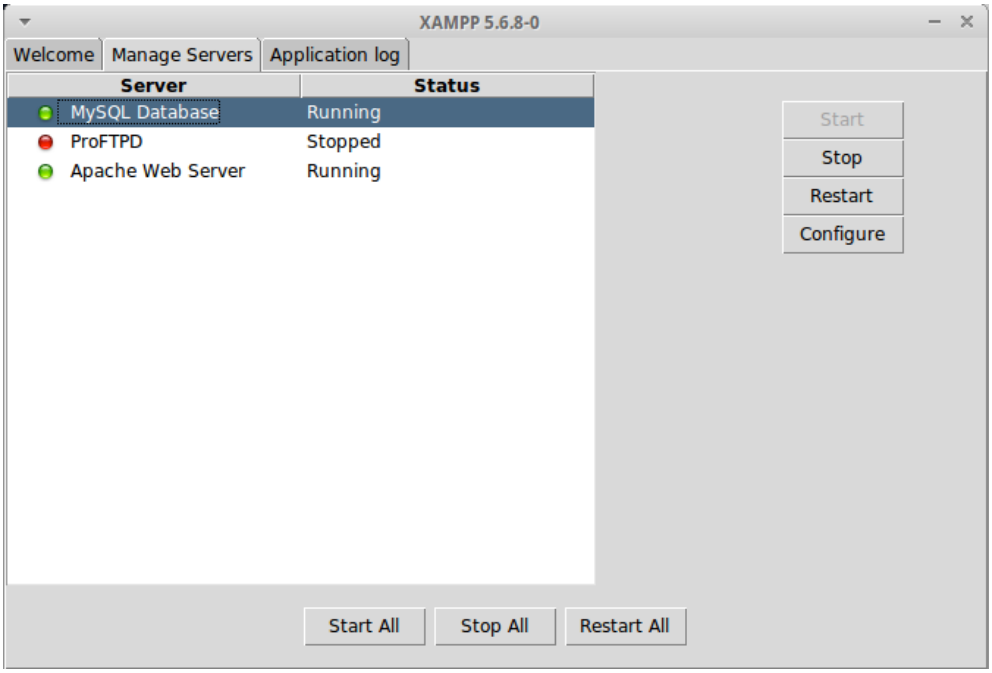


Figura 20.14 Panel de control XAMPP linux

Para comprobar que la instalación de Xampp se ha realizado correctamente, abrimos un navegador web e introducimos la siguiente URL:

Http://localhost/

Si todo ha ido bien el navegador nos mostrará una página similar a la indicada en la figura 20.15.

Figura 20.15 Interfaz de bienvenida XAMPP

Llegado a este punto ya podemos proceder a instalar el software FLL+ sobre nuestro servidor web recién creado.

20.2 Instalación de FLL+

Como en el caso de la instalación de Xampp, en este apartado vamos a explicar cómo instalar FLL+ para sistemas Windows y Linux. En ambos casos el proceso es muy similar, sin embargo existen algunas diferencias que merecen ser tomadas en cuenta y por tanto a continuación mostraremos dos apartados diferentes en los que explicaremos el proceso de instalación para cada uno de los dos sistemas operativos.

20.2.1 Instalación de FLL+ en Windows

El primer paso para instalar FLL+ es obtener desde el disco adjunto el fichero fll+.rar ubicado dentro de directorio "instalación". Después, nos ubicamos en el directorio donde hemos instalado Xampp y accedemos a la carpeta "htdocs". Borramos todo el contenido que este ubicado en ese directorio que en el caso de no haber modificado la ruta de instalación por defecto al instalar Xampp será el siguiente:

C:\xampp\htdocs

Copiamos en este directorio el fichero FLL+.rar y lo descomprimimos. Veremos que aparece una carpeta llamada "fll+" y un archivo llamado "fll+.sql". Una vez descomprimido el fichero en el directorio raíz de nuestro servidor web, accederemos a la siguiente URL con ayuda de un navegador web:

http://localhost/phpmyadmin

El navegador deberá mostrarnos una página similar a la mostrada en la figura 20.16

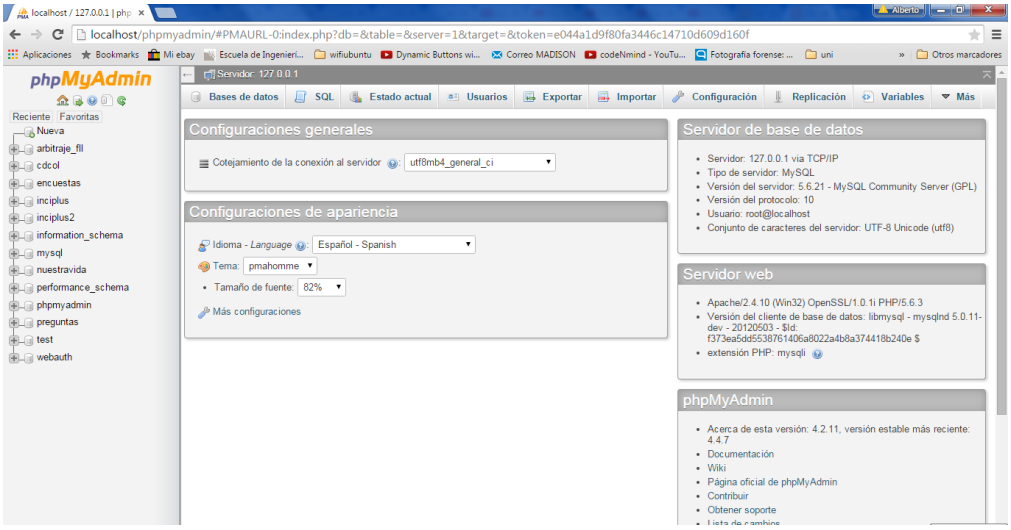


Figura 20.16 PHPMyAdmin

A continuación vamos a crear la base de datos que albergara la información persistente del software FLL+. Para ellos pulsamos en "Nueva" tal y como se muestra en la figura 20.17.

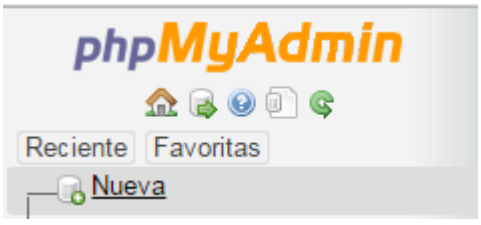


Figura 20.17 Detalle agregar nueva base de datos

Escribimos como nombre de base de datos FLL+ y elegimos en el desplegable “Cotejamiento” la opción “utf8_general_ci” tal y como se muestra en la figura 20.18. Esto último es importante, ya que sino lo realizamos correctamente los caracteres de la aplicación FLL+ no se verán de forma bien.

Figura 20.18 Crear base de datos relleno

Pulsamos en el botón crear y ¡listo! Ya tenemos creada la base de datos que albergara los datos de FLL+.

El siguiente paso es crear las estructuras de las tablas y los datos iniciales que usará FLL+. Para ello nos vamos a la pestaña “Importar” y en el campo “Seleccionar archivo” escogemos el fichero “fll+.sql” que habíamos descomprimido anteriormente. También debemos seleccionar en el campo “Conjunto de caracteres” la opción “utf-8”. Al final nos deberá quedar algo parecido a lo que muestra la figura 20.19.

Figura 20.19 Importar tablas FLL+

Pulsamos en el botón “continuar” y “phpmyadmin” comenzará a enviar al servidor MySQL las consultas necesarias para crear la estructura de tablas e información inicial. Este proceso puede durar varios segundos.

Una vez haya concluido este proceso al seleccionar la base de datos “fll+” nos aparecerán las tablas que se muestran en la figura 20.20.

Figura 20.20 Tablas FLL+

Ya solo nos queda un último paso para completar la instalación de FLL+. A continuación nos dirigimos a la siguiente ruta:

C:\xampp\htdocs\fll+\core\config

En esa ruta encontraremos el fichero "config.php". Lo abrimos con cualquier editor de texto plano, como por ejemplo el bloc de notas de Windows y haremos las siguientes modificaciones:

-Buscamos la línea donde pone "define("HOSTBD","TU_SERVIDOR");" y sustituimos TU_SERVIDOR por localhost.

-Buscamos la línea donde pone "define("USERBD","TU_USUARIO");" y sustituimos TU_USUARIO por root.

NOTA: La instalación del servidor Xampp crea un usuario con login "root" y contraseña vacía para MySQL. Es extremadamente recomendable crear una contraseña para el usuario root con la finalidad de proteger nuestra base de datos ante posibles atacantes. Otra opción es crear otro usuario con privilegios de lectura y escritura para la base de datos fll+ que creamos anteriormente.

-Buscamos la línea donde aparece "define("PASSBD","TU_PASSWORD");" y sustituimos TU_PASSWORD por la contraseña que hayamos definido para el usuario root o lo borramos si no hemos establecido una contraseña para el root (NO RECOMENDABLE).

-Buscamos la línea donde aparece "define("BD","TU_BASE_DE_DATOS");" y sustituimos TU_BASE_DE_DATOS por "fll+"

Después de estas modificaciones nuestro fichero de configuración deberá tener un aspecto similar al que muestra la figura 20.21.

```php
<?php
define("HOSTBD","localhost");

define("USERBD","root");

define("PASSBD","");

define("BD","fll+");

define("DIRCONTROLADORES","controladores/");

define("DIRVISTAS","vistas/");

define("DIRMODELOS","../core/modelos/");

define("DIRMAPPERS","../core/mappers/");

//numero de pruebas incluidas penalizaciones
define("NUMPRUEBAS",15);
?>
```

Figura 20.21 Fichero de configuración FLL+

Guardamos los cambios y ¡listo! Ya hemos realizado la instalación de FLL+. Para comprobar que todo ha ido bien nos dirigimos a la siguiente URL con un navegador web:

http://localhost/fll+

Si todo ha ido bien el navegador mostrará una web como la que aparece en la figura 20.22 ¡Enhorabuena! Has instalado FLL+. Si no fuese ese el caso, por favor, repite los pasos explicados anteriormente, ya que alguno de los pasos no los has realizado correctamente.

Figura 20.22 Login FLL+

20.1 Instalación de FLL+ en Linux.

Para no alargar demasiado este apartado indicaremos simplemente las diferencias que existen entre la instalación de FLL+ en Windows y Linux.

Básicamente la principal diferencia radica en la ruta donde se aloja el directorio raíz del servidor web Apache y en la forma de descomprimir el fichero donde se alojan los ficheros que forman el software FLL+.

Nos dirigimos al directorio "/opt/lampp/htdocs" y copiamos ahí el fichero fll+.rar que podemos encontrar en el cd adjunto dentro de la carpeta Linux. Una vez copiado lo descomprimimos con el siguiente comando:

unrar fll+.tar.gz

Después realizamos la carga de las estructuras de las tablas y los valores iniciales de la base de datos tal y como se indica en el apartado anterior para Windows.

De igual manera procedemos a modificador el fichero de configuración de la misma forma que hicimos para el caso de Windows.

Capítulo 21. Manual de usuario

21.1 Creación de usuarios para FLL+

El sistema FLL+ dispone de un sistema de autenticación y acceso a los diferentes recursos basado en roles, es decir, cada usuario existente en el sistema FLL+ pertenece a un rol. Este rol es el que definirá a que recursos puede acceder cada usuario.

FLL+ trae configurado tres tipos de roles diferentes, estos son:

-Administrador. El administrador puede acceder a todos los recursos del sistema, es decir, puede arbitrar, ver el marcador y por supuesto, acceder al panel de administración.

-Arbitro. Este rol tan solo permite acceder al sistema de arbitraje propiamente dicho, es decir, solo permite arbitrar cada una de las diferentes partidas.

-Público. Este rol está pensado para aquellas personas que asistan a un torneo FLL en calidad de público, por tanto, el rol de público solo permitirá acceder al marcador en tiempo real.

Dicho esto, pasamos a enumerar los pasos necesarios para crear un usuario dentro del sistema FLL+:

-El primer paso que debemos realizar es acceder a phpmyadmin a través de la siguiente URL:

http://localhost/phpmyadmin

-Una vez dentro, acedemos a nuestra base de datos “FLL+”

-Después, seleccionamos la tabla “usuarios”

-Hacemos clic en la pestaña “Insertar” y podremos ver un interfaz similar al de la figura 21.1.

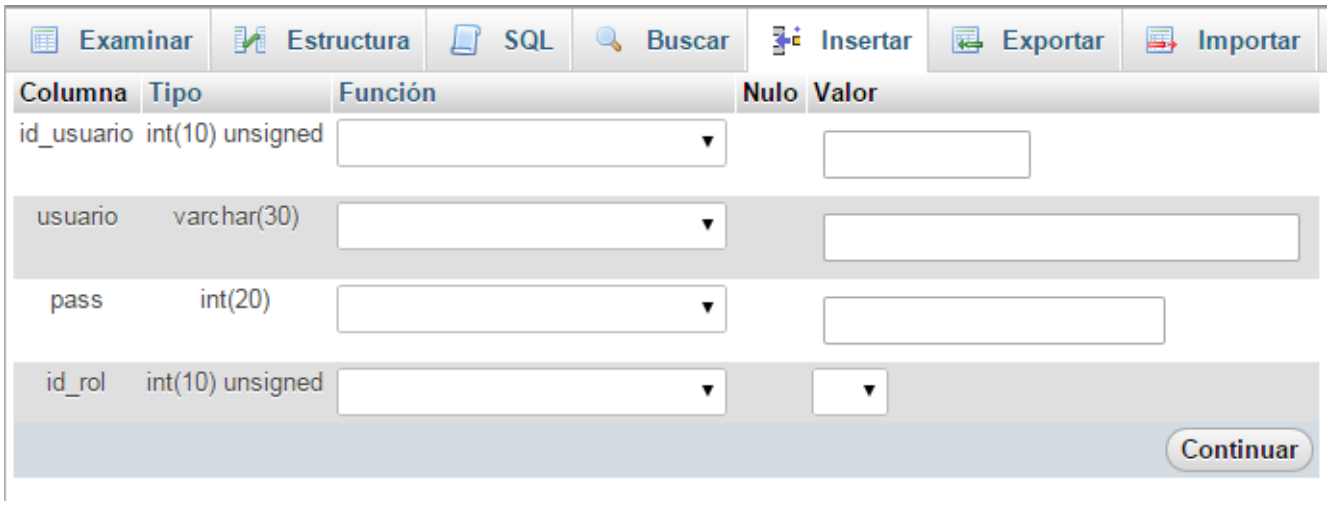


Figura 21.1 Insertar nuevo usuario

En el campo "id_usuario" no tenemos que insertar nada, ya que se trata de un campo autoincrementable y por tanto MySQL se encargará de darle un valor automáticamente.

En el campo "usuario" introducimos el nombre de usuario que deseemos darle al nuevo usuario.

En el campo "pass" introducimos la contraseña para el nuevo usuario.

Finalmente en el campo "id_rol" tenemos 3 opciones:

-Marcar la opción "1" que corresponde al rol de árbitro.

-Marcar la opción "2" que corresponde al rol de administrador.

-Marcar la opción "3" que corresponde al rol público.

Un ejemplo de cómo introducir un nuevo usuario con el rol árbitro lo podemos ver en la figura 21.2.

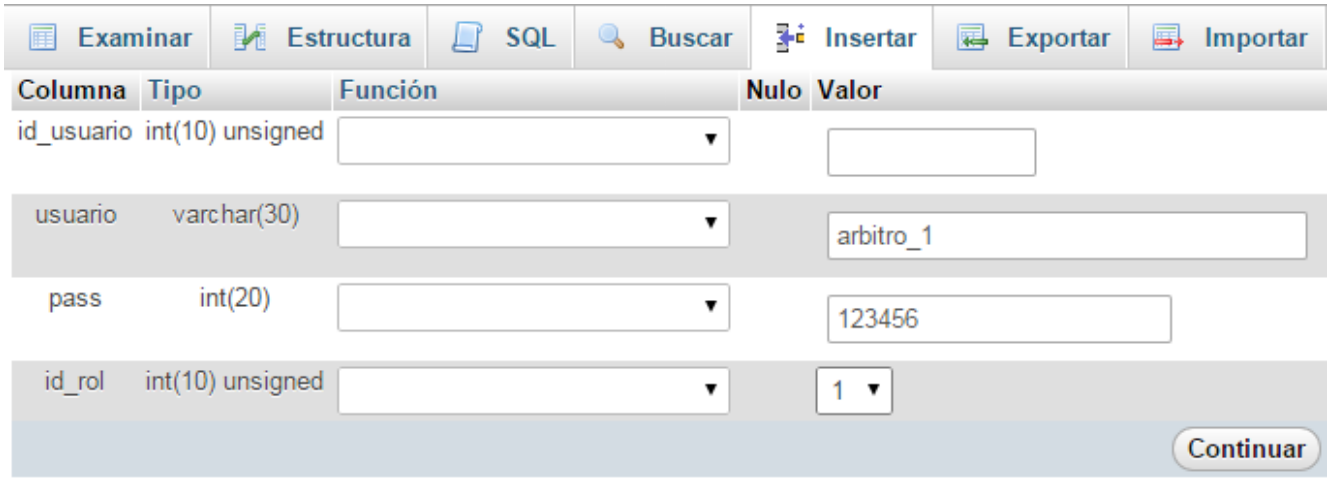


Figura 21.2 Insertar nuevo usuario relleno

-Por último hacemos clic en el botón continuar y ¡listo! ya hemos creado un nuevo usuario en el sistema FLL+.

21.2 Acceder al sistema FLL+

Si ya dispones de una cuenta en FLL+, acceder al sistema es tan sencillo como visitar el sitio web donde este alojada la aplicación FLL+, introducir sus datos de acceso y pulsar en el botón continuar. Si quieres acceder al sistema desde la propia máquina donde está instalado el servidor, tan solo es necesario que accedas a la siguiente URL:

http://localhost/fll+

Figura 21.3 Acceso a FLL+

21.3 Crear un nuevo rol

Como ya indicamos anteriormente FLL+ trae pre configurados 3 tipos de roles, el administrador, el público y árbitro, sin embargo, puedes insertar un nuevo rol con los permisos que desees.

Para hacerlo tan solo debes seguir estos sencillos pasos:

-Acceder a la URL donde está alojado “phpmyadmin”, si lo haces desde la máquina donde está instalado el servidor esta URL será la siguiente:

http://localhost/phpmyadmin/

-Seleccionamos la base de datos “FLL+”.

-Hacemos clic en la tabla “roles”.

-Nos vamos a la pestaña insertar y podremos visualizar un interfaz similar al mostrado en la figura 21.4.

Figura 21.4 Crear nuevo rol

En el campo "id_rol" no tenemos que insertar nada, ya que se trata de un campo autoincrementable y por tanto MySQL se encargará de darle un valor automáticamente.

En el campo "nombre" introducíos el nombre del nuevo rol.

Si queremos que el rol que estamos creando tenga permisos para arbitrar, insertaremos un "1" en el campo "permiso_arbitraje", en caso contrario un "0".

Si queremos que el rol que estamos creando tenga permisos para ver el marcador, insertaremos un "1" en el campo "permiso_marcador", en caso contrario un "0".

Si queremos que el rol que estamos creando tenga permisos de administración, insertaremos un "1" en el campo "permiso_administracion", en caso contrario un "0".

-Pulsamos el botón "continuar" y habremos concluido el proceso de creación del nuevo rol.

21.4 Acceder al marcador

Para acceder al marcador, previamente tenemos que haber accedido al sistema FLL+. Véase la sección 8.4.

Una vez hayamos accedido al sistema se mostrará un panel con las acciones que podemos realizar en función del rol de nuestra cuenta de usuario. Un ejemplo de este panel para el rol de "administrador" se muestra en la figura 21.5.

Figura 21.5 Panel de acciones

Ya en el panel de bienvenida, seleccionamos la opción "Marcador" y se abrirá una web donde se mostrarán todos los resultados en tiempo real del torneo FLL. Véase figura 21.6.

21.5 ¿Qué datos muestra el marcador?

A grandes rasgos, podemos decir que los datos que muestra el marcador son, obviamente, los datos en tiempo real del torneo FLL, sin embargo, cabe destacar la forma en que se disponen y configuran los datos dentro de la interfaz del marcador dentro del sistema FLL+.

Básicamente tenemos 3 elementos:

-El primero de ellos, situado en la parte superior izquierda muestra los resultados en tiempo real de la partida actual, es decir, la partida que se está jugando en ese preciso instante. Ver figura 21.6:

Figura 21.6 Panel de resultados en directo con partidas en curso

Si en ese momento no se estuviese jugando ninguna partida, este elemento mostrará un aspecto como el de la figura 21.7.

Figura 21.7 Panel de resultados en directo sin partidas en curso

-El segundo elemento es la cuenta atrás el cual se ubica en la zona superior derecha. La cuenta atrás es iniciada o detenida por el actor jefe de árbitros, la cual durara 10 segundos. Una vez llegue a 0 comenzará los dos minutos y medio que es la duración establecida para una partida en el reglamento de FLL. Ver figura 21.8.

Figura 21.8 Cuenta atrás

-El último elemento del marcador es la clasificación, la cual muestra en orden descendente la mejor puntuación obtenida por cada equipo en cada una de sus rondas, tal y como está definido en el reglamento FLL. Aquellas partidas que aún no se han jugado se mostrarán con 3 guiones consecutivos, las que se estén jugando en un momento dado se mostrarán con el texto “En curso” y las partidas que ya se hayan jugado mostrarán la puntuación que se obtuvo. Ver figura 21.9.

POS	EQUIPO	RONDA 1	RONDA 2	RONDA 3	TOTAL
1	ROBOMARQUES	0	325	---	325
2	LOCASTUERCAS	0	205	---	205
3	LEGOBOT	40	196	---	196
4	MARISTAS PALENCIA	0	---	---	192
5	ROBOT SABIOS	0	---	---	90
6	TECHCOM ROBOTS	0	65	---	65
7	BOTRONICA	0	---	---	0

Figura 21.9 Clasificación

21.6 Acceder al panel de arbitraje

Para acceder al panel de arbitraje, previamente tenemos que haber accedido al sistema FLL+ con una cuenta con permisos para arbitrar. Véase la sección 8.4.

Una vez estemos ubicados en pantalla de bienvenida seleccionamos la opción “Arbitrar”.

El sistema mostrará una lista que contendrá aquellas partidas que el jefe de árbitros haya marcado como “En curso” y que no estén siendo arbitradas por otros árbitros. Ver figura 21.10.

Figura 21.10 Panel selección de partida a arbitrar

Este panel tiene una característica o funcionalidad bastante interesante. Supongamos que un árbitro está visualizando el panel mostrado en la figura 21.10 y otro árbitro selecciona arbitrar a un equipo en concreto, pues bien, el panel automáticamente eliminara el equipo elegido por ese segundo árbitro, de tal forma que una partida de un equipo concreto no pueda ser arbitrada por dos árbitros simultáneamente, evitando así, problemas debidos a falta de comunicación entre los miembros de la organización.

Una vez el árbitro tenga claro que equipo y ronda tiene que arbitrar, tan solo tendrá que pulsar en el botón "Arbitrar" que corresponda para acceder al panel de arbitraje.

21.7 ¿Cómo se arbitra una partida?

Antes de poder arbitrar una partida de un equipo concreto, debemos haber accedido al panel de arbitraje tal y como se ha explicado en el apartado anterior.

Una vez hayamos accedido al panel de arbitraje podremos ver una interfaz como la que muestra la figura 21.11.

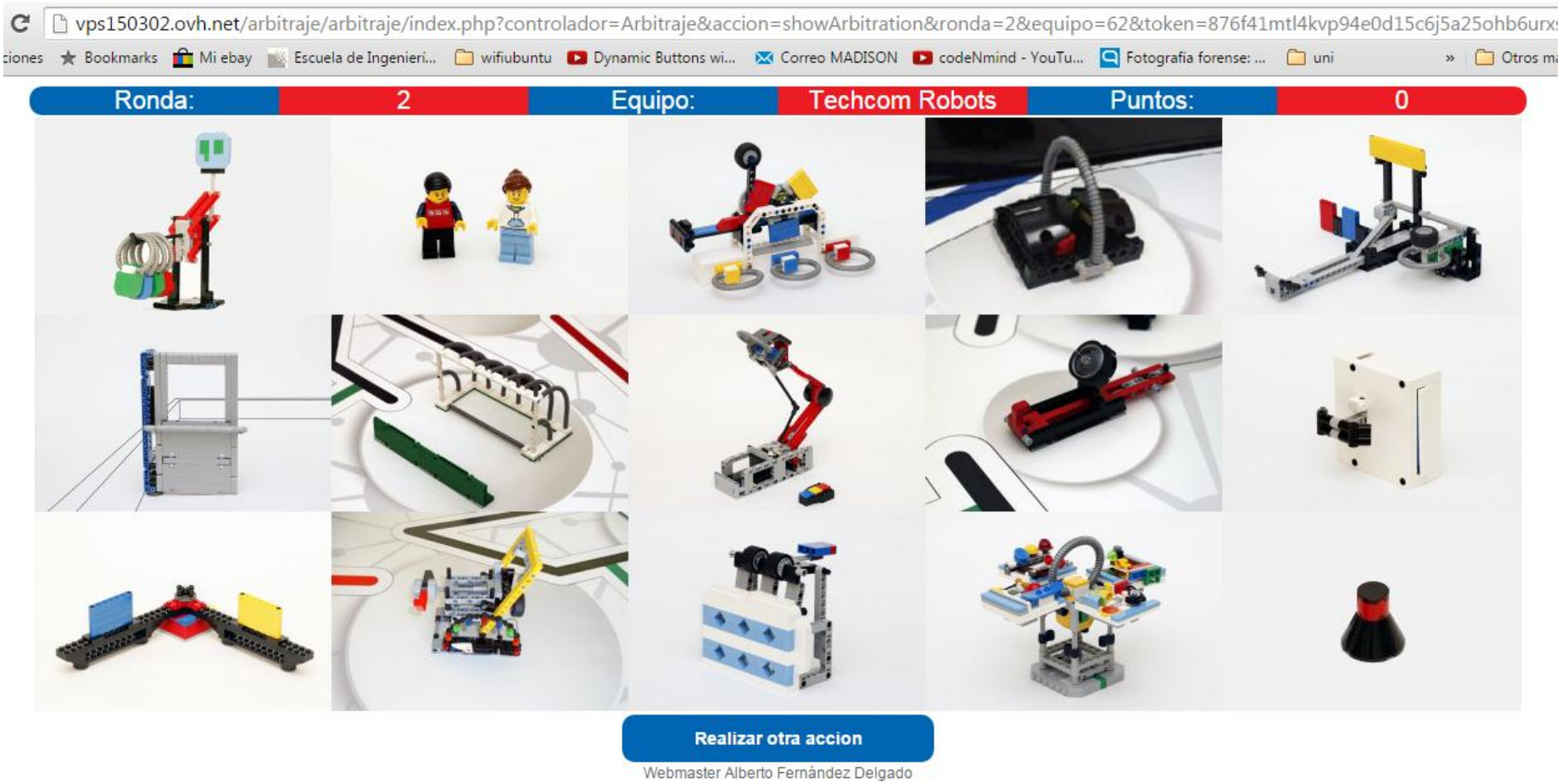


Figura 21.11 Panel de arbitraje

Cada una de las diferentes pruebas de las que consta el torneo FLL está representado por una imagen diferente tal y como se muestra en la figura 21.11. Las imágenes que se cargan por defecto al entrar al panel de arbitraje muestran el estado de cada prueba en su estado inicial. Por ejemplo, la prueba "Apertura de puertas" comienza con la puerta cerrada, por ello, la imagen inicial muestra la puerta en ese estado.

Para cambiar el estado de cada una de las pruebas tan solo debemos hacer clic sobre su imagen, y esta, cambiará a su estado siguiente. Por ejemplo, en la prueba "Motor de búsqueda" tenemos 3 estados posibles. El primero de ellos es cuando el robot no ha interactuado con la prueba, el segundo sucede cuando el robot empuja la palanca y hace girar la rueda y el tercero sucede cuando el robot retira el asa correspondiente al color que ha obtenido al girar la ruleta. En las figuras 21.12, 21.13 y 21.14 se muestra la sucesión de cada una de las imágenes para estas pruebas cuando el árbitro hace clic sobre ella.

Figura 21.12 Prueba en estado inicial

Figura 21.13 Prueba en paso 2

Figura 21.14 Prueba superada

Si por equivocación el árbitro pulsa en una prueba errónea, no hay problema, ya que al volver a pulsar sobre la imagen que se muestra en la figura 21.14, la prueba volvería a mostrarse en su estado original.

Por supuesto, cada vez que el estado de una prueba cambia como consecuencia de que el árbitro haga clic sobre ella, se actualizará la puntuación en la parte superior derecha del panel de arbitraje.

Llegado este punto y una vez que hemos explicado cómo se arbitra una prueba de forma general, vamos a explicar 3 casos particulares que afectan a las pruebas "Captar la atención", "Aprendizaje basado en proyectos" y a las penalizaciones. A continuación pasamos a detallar como se arbitran cada una de ellas.

-“Aprendizaje basado en proyectos”. Esta prueba consiste en depositar el mayor número de asas sobre un soporte, hasta un máximo de 8. Dado que es relativamente fácil que el árbitro se equivoque y que la prueba tiene 9 estados posibles, que van desde el inicial (el soporte no tiene ningún asa) hasta que el soporte tiene 8 asas, la prueba está programada de tal forma que al darle a la parte superior de la imagen se incrementa en uno las asas que cuelgan del soporte y al hacer clic en la parte inferior este número se decrementa tal y como se muestra en la figura 21.15.

Figura 21.15 Arbitrar “Aprendizaje basado en proyectos”

La zona verde suma 1 asa y la zona roja lo resta.

Al llegar al número 8, la prueba vuelve a su estado inicial, de esta forma evitamos tener que hacer clic varias veces en caso de error, esto es importante ya que las partidas transcurren de forma rápida y el árbitro no se puede demorar en exceso ya que impediría que su atención siguiese centrada en las acciones que el robot va realizando.

-“Captar la atención”. Esta prueba contiene un gran número de estados posibles, más en concreto 51. Por tanto sería una pérdida de tiempo muy significativa para el árbitro tener que hacer 20 o 30 clics sobre una misma prueba, por ello, la prueba está programada para que al hacer clic sobre la zona superior izquierda sume seis, al hacer clic sobre la zona inferior izquierda reste seis, al pulsar sobre la zona superior derecha sume uno y al hacer clic sobre la zona inferior derecha reste 1. De esta forma conseguimos que el árbitro pueda evaluar esta prueba de una forma mucho más cómoda, rápida, eficaz y rápidamente. Ver figura 21.16.

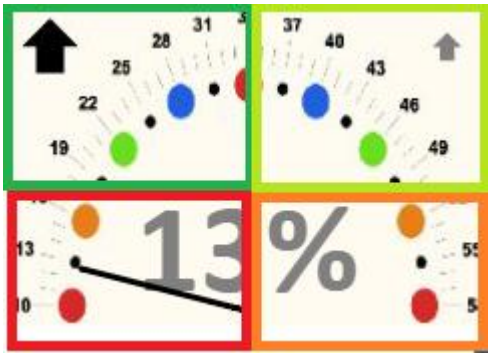


Figura 21.16 Arbitrar “Captar la atención”

-“Penalizaciones”. De forma similar a lo que sucedía en la prueba “aprendizaje basado en proyectos”, las penalizaciones también contienen un elevado número de estados, en concreto 9, por ello es muy útil que al pulsar sobre la zona superior de la imagen aumentemos en 1 la penalización y al hacer clic sobre la zona inferior ocurra lo contrario, es decir, se reste una penalización. Ver figura 21.17.

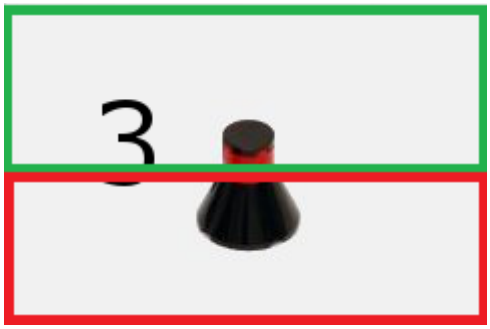


Figura 21.17 Arbitrar penalizaciones

De la misma forma que pasaba en la prueba “Aprendizaje basado en proyectos” si pulsamos sobre la prueba cuando tenemos 8 penalizaciones (número máximo de penalizaciones contemplado en el reglamento de la FLL), volveríamos al estado inicial, es decir, 0 penalizaciones.

Con estas pautas podemos dar por finalizado el manual de arbitraje, sin embargo, añadiré un comentario más. La mejor forma de aprender a utilizar de forma rápida y correcta el sistema de arbitraje es practicando días antes de la celebración del torneo, ya que en las partidas suceden muchas cosas de forma muy rápida y no es sencillo controlar las acciones que va realizando el robot a la vez que se van anotando las diferentes puntuaciones que se va obteniendo el equipo.

21.8 Acceder al panel de administración

Para acceder al panel de administración, previamente tenemos que haber accedido al sistema FLL+ con una cuenta con permisos de administración.. Véase la sección 8.4.

Una vez estemos ubicados en pantalla de bienvenida seleccionamos la opción “Administrar”.

21.9 ¿Qué acciones podemos hacer desde el panel de administración?

El panel de administración está pensado para ser utilizado por el jefe de árbitros, por tanto, está diseñado para ser utilizado por una única persona al mismo tiempo ya que tan solo suele existir un jefe de árbitros, sin embargo, FLL+ puede soportar un uso en paralelo de varios jefes de árbitros para supuestos eventos de mayor envergadura.

Dicho esto, vamos a pasar a enumerar cada una de las acciones que puede realizar el jefe o jefes de árbitros desde el panel de administración para luego, en los apartados siguientes, explicar cómo se deben llevar a cabo cada una de esas posibles acciones:

-Añadir un equipo.

-Eliminar un equipo.

-Planificar una partida.

-Modificar la puntuación de una partida.

-Modificar el estado de una partida. Los estados en los que puede estar una partida son:

-Jugada.

-No jugada.

-En curso.

-Exportar una partida a formato PDF.

-Controlar que árbitros están arbitrando cada una de las partidas.

-Denegar a un árbitro concreto arbitrar una partida concreta.

-Eliminar una partida.

-Iniciar la cuenta atrás.

-Parar la cuenta atrás.

-Acceder a la documentación oficial de la FLL.

Como ya hemos mencionado, a continuación pasamos a describir como llevar a cabo cada una de las acciones indicadas anteriormente. De ahora en adelante asumiremos que el jefe de árbitros se ha identificado frente al sistema FLL+ como tal y se encuentra en la página principal del panel de administración mostrado en la figura 21.18.

Figura 21.18 Bienvenida panel de administración

21.10 Añadir un equipo

Para añadir un equipo debemos pulsar en la pestaña equipos y FLL+ nos mostrará un interfaz como el que se muestra en la figura 21.19.

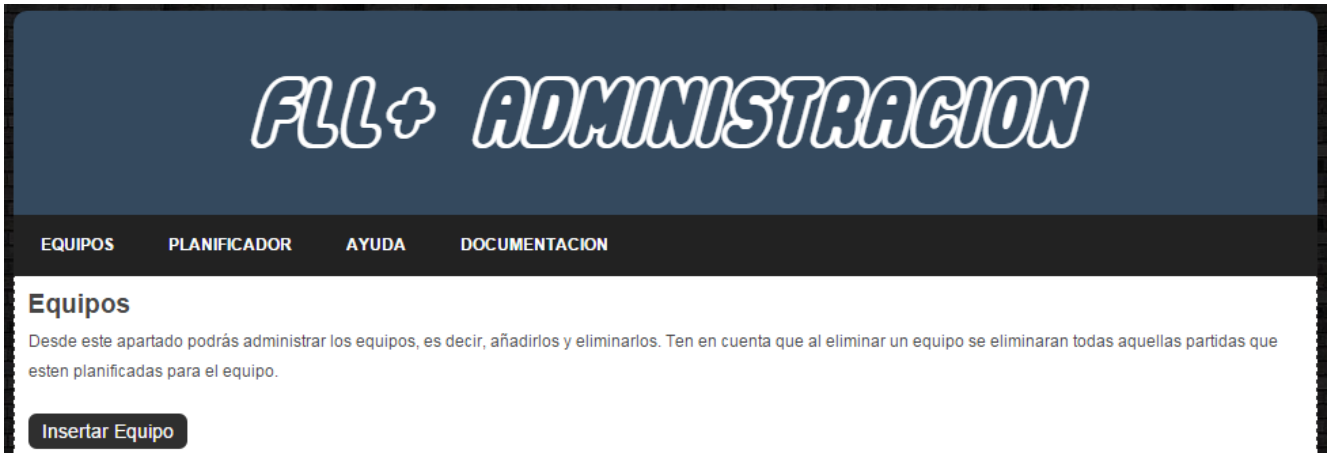


Figura 21.19 Menú Equipos

Pulsamos sobre el botón “Insertar equipo” y aparecerá un desplegable como el mostrado en la figura 21.20.

Figura 21.20 Menú insertar equipo

En el campo nombre introducimos el nombre del equipo que queremos crear.

En el campo descripción introduciremos la descripción del equipo, útil para ayudar al jefe de árbitros a identificar los diferentes equipos en torneos con mucha afluencia de equipos.

Por último hacemos clic en el botón "Insertar equipo" y el sistema guardará los datos de manera persistente.

21.11 Eliminar un equipo

Para eliminar un equipo debemos hacer clic en la pestaña equipos dentro del panel de administración. Una vez ahí, pulsamos sobre el texto "Eliminar" en el equipo que deseemos eliminar tal y como muestra la figura 21.21.

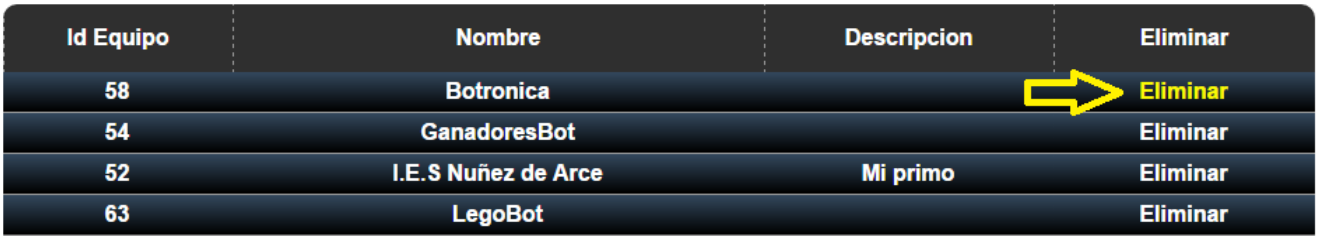


Figura 21.21 Eliminar equipo

Nota: Ten precaución al utilizar esta acción, ya que al eliminar un equipo no solamente estarás eliminando el equipo en sí, sino que, estarás eliminando todas las partidas, puntuaciones… que estuviesen asociadas a ese equipo en la base de datos, por tanto, utilice esta función solo si está realmente seguro de lo que va a hacer.

21.12 Planificar una partida

Esta acción la utilizarás frecuentemente antes del inicio del evento FLL. El objetivo es planificar una determinada partida de un determinado equipo para una determinada hora.

A continuación pasamos a detallar los pasos necesarios para planificar una partida.

-Primero hacemos clic en la pestaña "Planificador" dentro del panel de administración. El sistema FLL+ le mostrará un interfaz similar al que se muestra en la figura 21.22.

Figura 21.22 Menú planificador

-Pulsamos en el botón Insertar partida y FLL+ mostrará un desplegable como el mostrado en la figura 21.23.

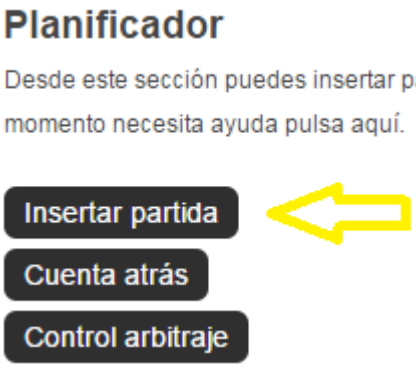


Figura 21.23 Acceder al menú planificar partida

Figura 21.24 Menú planificar partida

-En el campo “Equipo” seleccionamos el equipo para el cual queremos planificar una partida. Este campo nos mostrará por defecto una lista de los equipos que ya estén introducidos en la base de datos.

-En el campo “Ronda” seleccionamos la ronda que queremos planificar para el equipo seleccionado anteriormente.

-En el campo “Hora” introducimos la hora a la que se jugara la partida.

-Por ultimo pulsamos en el botón “Insertar partida”. Ver figura 21.25.

Figura 21.25 Menú planificar partida rellenado

-Finalmente el sistema FLL+ introducirá la partida en la base de datos y la podrás visualizar en la tabla ubicada en la parte inferior del menú de planificación. Ver figura 21.26. Las partidas introducidas se crean en el estado “No jugada” de forma predeterminada.

Figura 21.26 Nueva partida planificada

Por ultimo mencionar que si intentas planificar una partida que ya estaba introducida en la base de datos, el sistema FLL+ no te dejara llevar a cabo dicha acción y mostrará un mensaje como el que se indica en la figura 21.27.

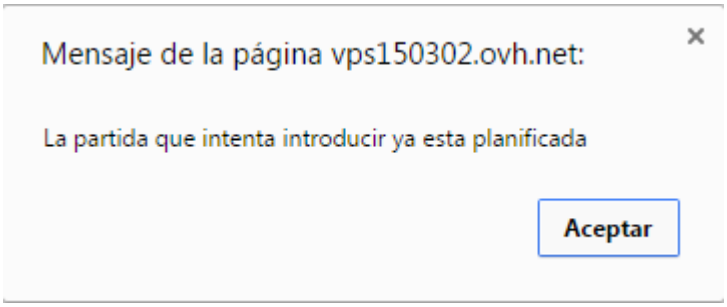


Figura 21.27 Mensaje de partida ya planificada

21.13 Modificar la puntuación de una partida

Esta acción es especialmente útil cuando una partida ha sido marcada como “Jugada” pero por motivos diversos ha existido una equivocación por parte del árbitro a la hora de puntuar una partida.

Para modificar la puntuación de una partida debemos realizar los siguientes pasos.

-En la pestaña “Planificador” busque la partida que desea modificar.

-Pulse en el icono “Editar” tal y como se muestra en la figura 21.28.

Figura 21.28 Cómo editar una partida

-En la página se superpondrá un menú como el de la figura 21.29.

Figura 21.29 Menú edición partida

-Después, tan solo tendrás que pulsar en cada una de las pruebas para modificar la puntuación de cada una de ellas. Si tienes dudas de cómo llevar a cabo este paso revisa la sección 8.9.

-Según vayas haciendo modificaciones sobre las distintas pruebas, se irá modificando la puntuación en la parte superior derecha.

-Una vez hayas concluido de editar la partida, pulsa sobre la "x" situada en la parte superior derecha y los datos quedaran actualizados en la base de datos. Ver figura 21.30.

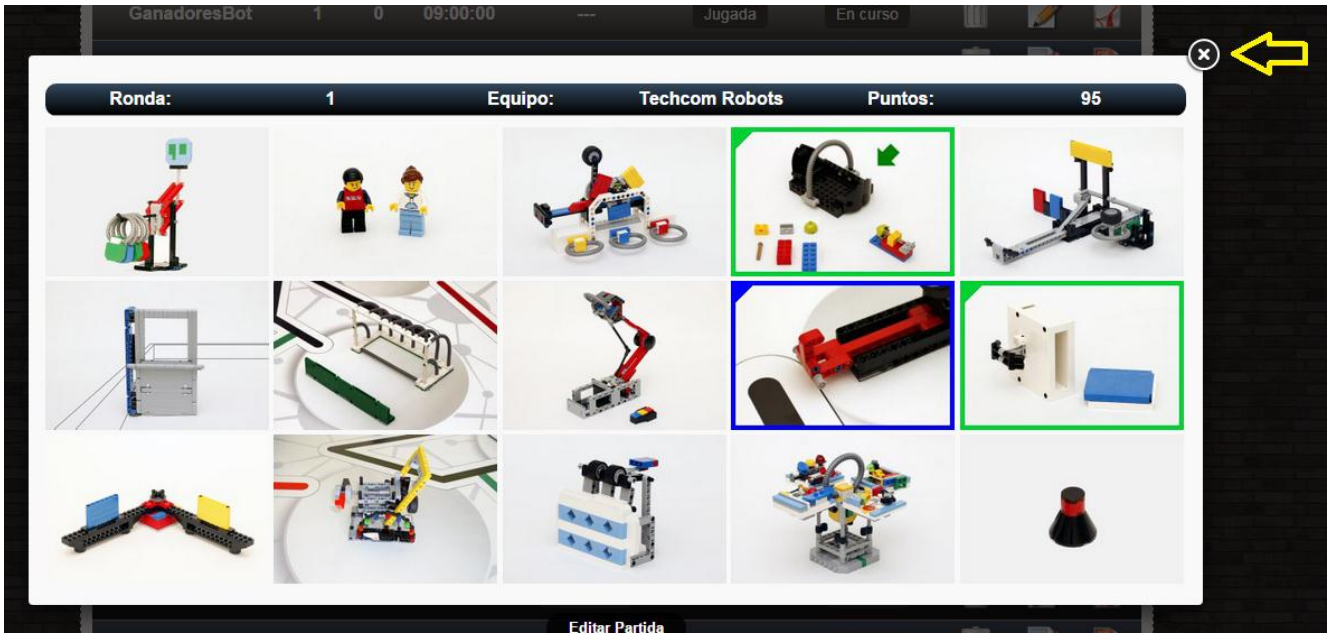


Figura 21.30 Cerrar menú edición partida

21.14 Modificar el estado de una partida

Con el transcurso del torneo las partidas van cambiando de estado. Estos cambios de estados siguen el siguiente patrón.

-Al ser planificada una partida, esta se inicializa como "No jugada".

-Al llegar la hora en la que la partida debe ser jugada, el jefe de árbitros la marca como "En curso".

-Una vez haya concluido la partida, el jefe de árbitros marcará la partida como "Jugada"

Para modificar el estado de las partidas debemos buscar la partida en cuestión y pulsar sobre los botones "No jugada", "En curso" y "Jugada" según corresponda. Ver figura 21.31.

Figura 21.31 Controlar los estados de la partida

Cada uno de los cambios de estados de la partida repercutirá en tiempo real en otras partes de la aplicación web sin necesidad de que otros tengan que recargar la página, en concreto para cada estado se producirán los siguientes cambios en otros elementos del sistema FLL+:

-Cambiar estado a "No jugada". La partida se mostrará en la clasificación del marcador con tres guiones consecutivos (---). Si el paso anterior era "En curso" la partida desaparecerá del apartado configuración de arbitraje para que ningún árbitro pueda entrar a arbitrarla y en el caso de que hubiese un árbitro arbitrándola, todas las acciones que haga el árbitro no tendrán efecto informándole al mismo de ello.

-Cambiar estado a “Jugada”. La partida se mostrará en la clasificación del marcador con la puntuación obtenida en la misma. Como en el caso anterior, si el estado anterior de la partida era “En curso” la partida desaparecerá del apartado configuración de arbitraje para que ningún árbitro pueda entrar a arbitrarla y en el caso de que hubiese un árbitro arbitrándola, todas las acciones que haga el árbitro no tendrán efecto informándole al mismo de ello.

-Cambiar estado a “En Curso”. La partida se mostrará como “En curso” dentro del marcador, además, aparecerá de forma automática en los resultados de las partidas actuales dentro del marcador. Por último, la partida será accesible desde el menú de configuración de arbitraje para que los árbitros puedan comenzar su arbitraje.

Nota: Observa que las partidas marcadas como “Jugadas” aparecen con un fondo negro degradado a azul y letras blancas. Las partidas marcadas como “En curso” aparecen con un fondo negro degradado a azul con letras amarillas y las partidas marcadas como “No jugadas” aparecen con un fondo con degradado de grises. De esta forma, puedes ver rápidamente el estado de cada una de las partidas.

21.15 Exportar una partida a PDF

Una vez que cierta partida ha sido marcada como “Jugada” el sistema FLL+ te permite generar un documento en formato PDF con el resumen de la partida. Lo habitual es imprimir dos copias, una para la organización de la FLL y otra para el equipo que participo en la partida, para que, tanto el árbitro como el equipo, puedan firmar esa hoja con el resumen de las puntuaciones obtenidas en cada una de las pruebas, mostrando así su conformidad con el arbitraje llevado a cabo por la organización del evento FLL.

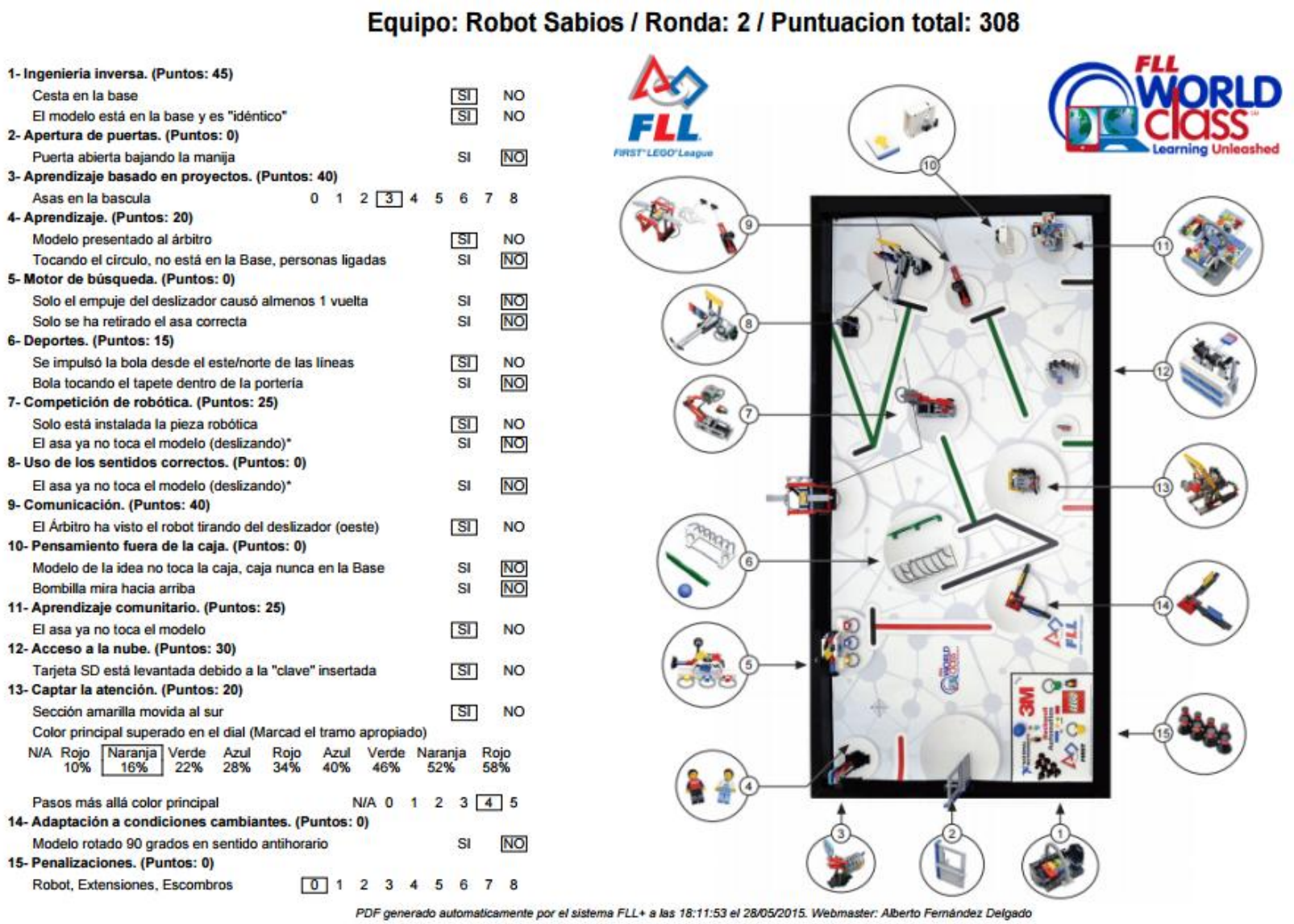
A continuación pasamos a detallar los pasos necesarios para exportar una partida a formato PDF.

-Primero busque la partida que desea exportar desde la pestaña planificador.

-Una vez tengas claro que partida desea exportar, haz clic en el icono “PDF” tal y como se muestra en la figura 21.32.

Figura 21.32 Como exportar a PDF una partida

-FLL+ generara un documento con un resumen de las puntuaciones obtenidas en cada una de las diferentes pruebas que forman la partida con todo tipo de detalle. Un ejemplo de partida exportada a PDF lo podemos ver en la figura 21.33.



Equipo: Robot Sabios / Ronda: 2 / Puntuacion total: 308

1- Ingenieria inversa. (Puntos: 45)
Cesta en la base [SI] NO
El modelo está en la base y es "idéntico" [SI] NO
2- Apertura de puertas. (Puntos: 0)
Puerta abierta bajando la manija SI [NO]
3- Aprendizaje basado en proyectos. (Puntos: 40)
Asas en la bascula 0 1 2 [3] 4 5 6 7 8
4- Aprendizaje. (Puntos: 20)
Modelo presentado al árbitro [SI] NO
Tocando el circulo, no está en la Base, personas ligadas SI [NO]
5- Motor de búsqueda. (Puntos: 0)
Solo el empuje del deslizador causó almenos 1 vuelta SI [NO]
Solo se ha retirado el asa correcta SI [NO]
6- Deportes. (Puntos: 15)
Se impulsó la bola desde el este/norte de las líneas [SI] NO
Bola tocando el tapete dentro de la portería SI [NO]
7- Competición de robótica. (Puntos: 25)
Solo está instalada la pieza robótica [SI] NO
El asa ya no toca el modelo (deslizando)* SI [NO]
8- Uso de los sentidos correctos. (Puntos: 0)
El asa ya no toca el modelo (deslizando)* SI [NO]
9- Comunicación. (Puntos: 40)
El Árbitro ha visto el robot tirando del deslizador (oeste) [SI] NO
10- Pensamiento fuera de la caja. (Puntos: 0)
Modelo de la idea no toca la caja, caja nunca en la Base SI [NO]
Bombilla mira hacia arriba SI [NO]
11- Aprendizaje comunitario. (Puntos: 25)
El asa ya no toca el modelo [SI] NO
12- Acceso a la nube. (Puntos: 30)
Tarjeta SD está levantada debido a la "clave" insertada [SI] NO
13- Captar la atención. (Puntos: 20)
Sección amarilla movida al sur [SI] NO
Color principal superado en el dial (Marcad el tramo apropiado)

N/A	Rojo 10%	[Naranja 16%]	Verde 22%	Azul 28%	Rojo 34%	Azul 40%	Verde 46%	Naranja 52%	Rojo 58%

Pasos más allá color principal N/A 0 1 2 3 [4] 5
14- Adaptación a condiciones cambiantes. (Puntos: 0)
Modelo rotado 90 grados en sentido antihorario SI [NO]
15- Penalizaciones. (Puntos: 0)
Robot, Extensiones, Escombros [0] 1 2 3 4 5 6 7 8

PDF generado automaticamente por el sistema FLL+ a las 18:11:53 el 28/05/2015. Webmaster: Alberto Fernández Delgado

Figura 21.33 Ejemplo informe PDF de una partida

21.16 Gestionar control de árbitros

En este apartado vamos a explicar cómo podemos gestionar que árbitro está haciendo que y en qué momento desde el panel de administración.

Para explicarlo vamos a suponer que el jefe de árbitros ha marcado dos partidas en el estado “En curso” pero aún no tienen un árbitro asociado. Pues bien, imaginemos que el jefe de árbitros le dice a cierto árbitro que es el turno de que arbitre la ronda 2 del equipo “Locas Tuercas”. El árbitro accederá al panel de configuración de arbitraje y podrá ver algo como lo indicado en la figura 21.34.

Figura 21.34 Comenzar a arbitrar una partida

En el momento en que el árbitro pulse en el botón arbitrar el sistema le asignará un token único que le dará permiso para arbitrar esa partida, a la vez que, desde la pestaña planificador del apartado de administración aparecerá una línea indicando que ese árbitro ha comenzado a arbitrar la ronda 2 del equipo “Locas tuercas”.

Para acceder a la sección de control de árbitros tan solo debemos pulsar el botón “Control arbitraje” dentro del planificador, tal y como se muestra en la figura 21.35.

Figura 21.35 Acceder al menú control arbitraje

Al hacer clic sobre ese botón nos aparecerá un menú como el de la figura 21.36.

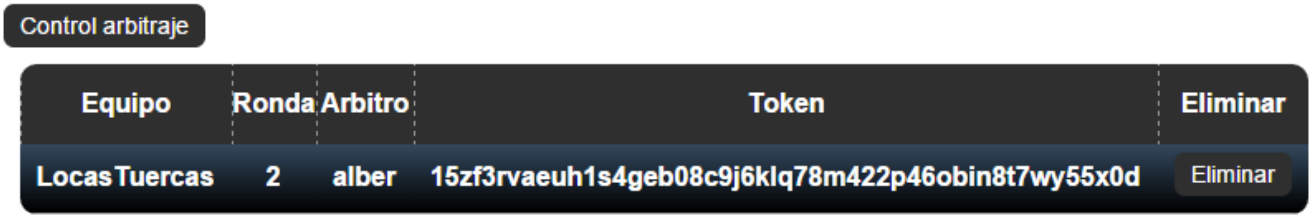


Figura 21.36 Menú Control Arbitraje

Como vemos, para cada partida que este siendo arbitrada, este panel nos mostrará, el equipo que está arbitrando, la ronda, que árbitro ha comenzado a arbitrar y el token que permite a ese árbitro llevar a cabo su tarea.

Por último y para terminar este apartado, podemos ver que a la izquierda de la figura 21.36 tenemos un botón en el que pone "Eliminar". Este botón sirve para poder sacar a un árbitro de una partida concreta, en el caso de que por error algún árbitro haya accedido a una partida que no debía ser arbitrada por él. En el momento en que eliminemos ese token de acceso y el árbitro pulse sobre alguna de las pruebas se le informara con un mensaje como el mostrado en la figura 21.37.

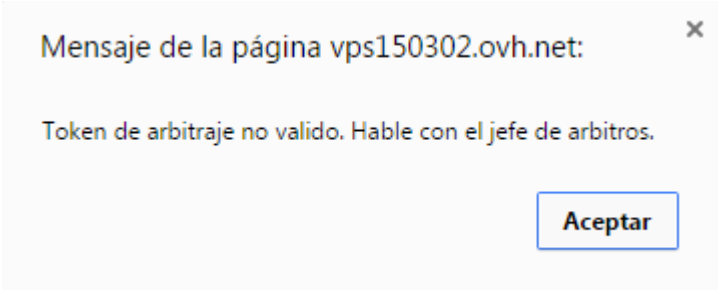


Figura 21.37 Mensaje token no válido

21.17 Iniciar la cuenta atrás

Una vez los árbitros han entrado a las partidas que deben arbitrar, el jefe de árbitros deberá iniciar la cuenta atrás.

La forma de hacerlo es muy sencilla. Tan solo debemos hacer clic sobre el botón "Cuenta atrás" dentro del planificador y FLL+ mostrará un menú como el que aparece en la figura 21.38.

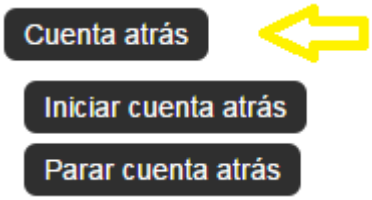


Figura 21.38 Iniciar cuenta atrás

Después pulsamos en el botón "Iniciar cuenta atrás" y desde el marcador podremos ver como se inicia una cuenta atrás desde 10 hasta 0. Una vez haya llegado a 0 comenzarán los dos minutos y medio que es el tiempo que dura una partida según indica el reglamento FLL. Ver figura 21.39.

Figura 21.39 Detalle cuenta atrás iniciada

21.18 Detener la cuenta atrás

En cualquier momento el jefe de árbitros puede detener la cuenta atrás ya sea porque alguno de los equipos ha tenido un imprevisto o causas similares.

Para hacerlo pulsamos en el botón "Parar cuenta atrás" dentro del menú "Cuenta atrás". Ver figura 21.40.

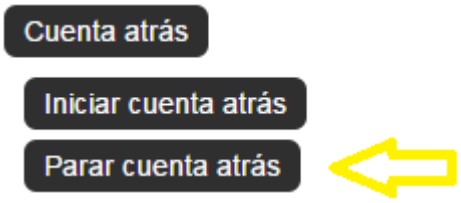


Figura 21.40 Parar cuentaatrás

En el momento en que pulsemos ese botón, la cuenta atrás se detendrá automáticamente.

21.19 Acceder a la documentación oficial de la FLL

Durante el transcurso de un evento FLL, es frecuente que surjan situaciones de excepción, dudas relativas al arbitraje etc.

FLL+ dispone de un apartado dentro del apartado de administración con toda la documentación oficial FLL (reglamento, pruebas, hojas de puntuación…) que ayudará al jefe de árbitros a tener muy accesible esta información para poder manejar de forma correcta este tipo de situaciones de excepción, dudas…

Para acceder a esta documentación tan solo debemos pulsar en la pestaña "DOCUMENTACION" dentro del panel de administración, tal y como se muestra en la figura 21.41.

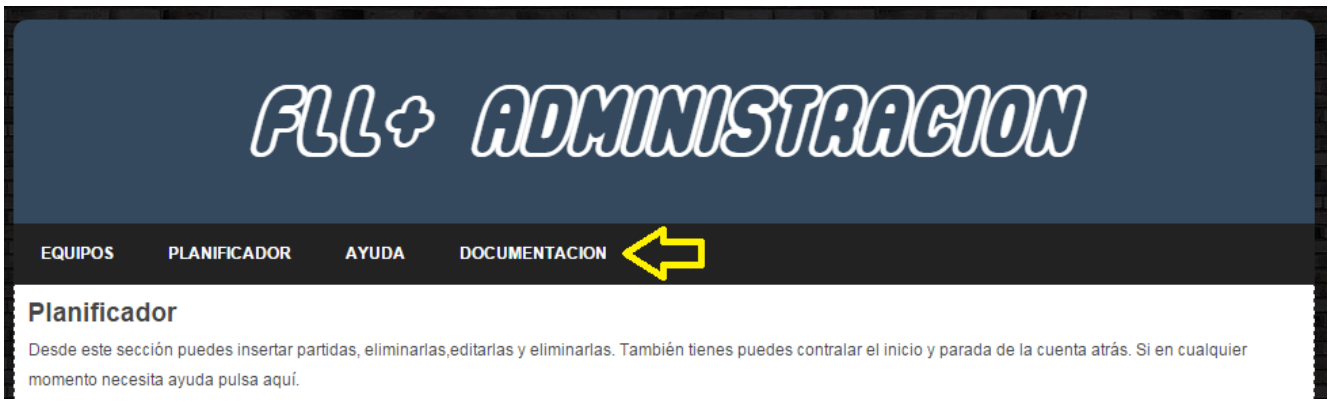


Figura 21.41 Acceder a la documentación

Nota: También es posible acceder a este manual de usuario desde la pestaña ayuda, para facilitar a la organización del evento ayuda en caso de que no se consiga o no se sepa utilizar algunas de las funcionalidades que ofrece FLL+. Para acceder a esta ayuda tan solo debemos pulsar en la pestaña "Ayuda". Ver figura 21.42.

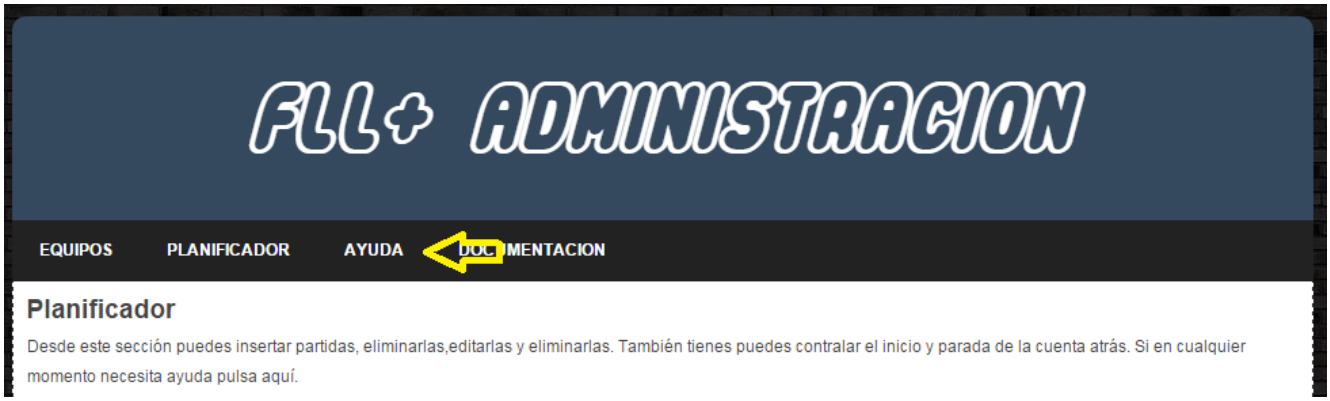


Figura 21.42 Acceder a la ayuda

Bloque VII. Conclusiones y anexos

Capítulo 22. Conclusiones

22.1 Objetivos alcanzados

En este apartado vamos a hacer una reflexión respecto a la idea de si hemos alcanzado los objetivos que nos habíamos marcado en las primeras fases de análisis o no.

Dadas las pruebas realizadas, hemos sido capaces de construir un sistema capaz de soportar cientos de usuarios conectados al mismo tiempo, visualizando en tiempo real los resultados de un torneo FLL. A su vez, el sistema permite a los árbitros arbitrar las partidas de una forma cómoda mientras envían las puntuaciones de forma transparente al servidor.

Por otro lado, la aplicación permite al jefe de árbitros llevar un control absoluto sobre los diferentes elementos, tales como, los equipos, las partidas, los árbitros… Por tanto podemos afirmar que se han alcanzado los objetivos que nos propusimos en fases anteriores y que el proyecto ha finalizado con éxito.

22.2 Dificultades

A pesar de las dificultades propias del desarrollo de software, como pueden ser la búsqueda de soluciones en la fase de diseño para alcanzar los objetivos marcados o la dificultad de realizar una herramienta que sea factible implementar en el tiempo del que se dispone, he encontrado algunas dificultades a lo largo del desarrollo del proyecto que merecen ser mencionadas en este apartado.

Estas dificultades han sido las siguientes:

- Proporcionar una interfaz web que muestre datos en tiempo real sin necesidad de que el usuario tenga que llevar a cabo ninguna acción.

- Implementar el patrón datamapper cuando los objetos de la lógica de dominio están relacionados unos con otros.

- Modificar el Document Object Model (DOM) con ayuda del framework jQuery una vez se ha obtenido la respuesta del web service en formato JSON.

- Tomar las decisiones acerca de que motor de bases de datos utilizar para hacer la aplicación web más rápida sin perder funcionalidad.

-Aprender la sintaxis y semántica del framework jQuery, que aunque al principio hizo retrasar la fase de implementación, finalmente se ganó tiempo y se alcanzó un nivel de calidad, que sin él, probablemente no se hubiese alcanzado.

22.3 Líneas de trabajo futuras

Ahora llega la hora de la autocrítica. En este punto vamos a explicar que partes creemos que se pueden mejorar o que funcionalidades adicionales podría tener la aplicación, para que, si en un futuro se retoma el proyecto, podamos tener unas líneas de trabajo por las que continuar el proyecto para hacerlo más funcional.

A continuación pasamos a enumerar esas líneas de trabajo:

-Añadir una opción que permita visualizar el marcador con diferentes estilos. Incluso podrían ser estilos configurables por el usuario.

-A pesar de que las pruebas de la FLL son relativamente estáticas y con "estáticas" me refiero a que cambian muy poco o no lo hacen de unos años a otros y que ahora mismo introducir nuevas pruebas o modificar las existentes no llevaría mucho trabajo, se podría realizar un módulo en la parte de administración que permitiese modificar o insertar nuevas pruebas a través de un interfaz web.

-Añadir un módulo en la parte de administración que permita crear, eliminar y modificar usuarios, para no tener que crearlos directamente desde la base de datos.

-Traducir la aplicación a otros idiomas, de forma que desde el fichero de configuración "config.php" se pueda seleccionar un lenguaje u otro, para que al utilizar la herramienta en torneos FLL celebrados en otros países, la organización pueda trabajar con la herramienta es su lengua materna.

-Realizar una versión móvil de la aplicación. Esto no sería muy complejo, ya que la mayor parte del sistema es responsive, así que introduciendo un layout que muestre la página web, se podría realizar fácil y rápidamente una aplicación para dispositivos móviles.

Apéndices

Contenido CD

A continuación, describimos los contenidos del CD-ROM que acompaña esta documentación:

- Directorio **codigoFuente**
  - Contiene todos los ficheros con extensión PHP, JS y CSS que forman la aplicación.
- Directorio **instalación**
  - Contiene todo el código fuente comprimido en formato “rar” y el fichero .sql que forma la estructura de tablas de la base de datos.
- Directorio **manual**
  - Contiene el manual de instalación y de usuario de la aplicación
- Fichero **memoria.pdf**

Referencias web

- http://www.php.net/

  Organización PHP

  Descripción Página oficial del lenguaje PHP

  Aplicación En esta web se ha consultado toda la documentación relativa al lenguaje PHP.

- https://jquery.com/

  Organización jQuery

  Descripción Página oficial de jQuery

  Aplicación Se ha utilizado para consultar la documentación de este framework

- http://www.w3schools.com/

  Organización w3schools

  Descripción Página de tutoriales sobre aplicaciones web

  Aplicación Se ha utilizado para consultar cuestiones relativas a css, Ajax y javascript

- http:// http://fancybox.net/

  Organización Fancybox

  Descripción Página oficial del plugin FancyBox para jQuery.

  Aplicación Se ha utilizado para consultar la documentación oficial de este plugin

- http://hilios.github.io/jQuery.countdown/

  Organización github

  Descripción Página oficial del plugin countDown para jQuery.

  Aplicación Se ha utilizado para consultar la documentación oficial de este plugin

- https://librosweb.es

  Organización Libros web

  Descripción Página con libros sobre desarrollo web

  Aplicación Se ha utilizado para consultar documentación sobre la tecnología ajax

Bibliografía

[1] G. Booch, J. Rumbaugh, I. Jacobson. *"El lenguaje unificado de desarrollo de software"*, Ed. Addison Wesley.

[2] C. Larman. *"UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado",* Ed. Pearson Prentice Hall.

[3] I. Sommerville. *"Ingeniería de Software"*, Ed. Addison-Wesley.

[4] J. Chessman, J. Daniels. *"UML Components. A Simple Process for Specifying Component-Based Software",* Ed. Addison-Wesley.

[5] R. S. Presman. *"Ingeniería del Software. Un enfoque práctico"*, Ed. McGraw Hill

[6] C. Pérez . *"MySQL para Windows y Linux "*, Ed. Ra-Ma.

[7] L. Ullman. *"MySQL : guía de aprendizaje ",* Ed. Pearson Educación.

[8] Vaswani, Vikram. *"Fundamentos de PHP*", Ed. McGraw-Hill