Universidad de las Ciencias Informáticas

Facultad 8

Título: Concepción y desarrollo del Módulo Arbitraje del Proyecto Infodrez

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Yendri Manresa Peña

Henry Herminio Carmenate García

Tutor: MBC Humberto González Hernández

Co-tutor: MCs Isbel Herrera del Sol

Ciudad de la Habana, Junio 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yendri Manresa Peña

_____

Firma del Autor

Henry Herminio Carmenate García

_____

Firma del Autor

Humberto González Hernández

_____

Firma del Tutor

FRASE

"Si alguna vez nuestro trabajo nos pareciera bueno, debemos luchar por hacerlo mejor, luchar por hacerlo perfecto, sabiendo de antemano que no hay obra humana totalmente perfecta."

Fidel Castro Ruz.

AGRADECIMIENTOS.

Agradecemos a todas las personas que de una forma u otra han contribuido al desarrollo del presente trabajo de diploma. Especialmente, a nuestros padres y familiares por apoyarnos en todo momento y estar siempre a nuestro lado.

A nuestro tutor Humberto por sus innumerables aportes y por conducirnos de la manera en que lo hizo.

DEDICATORIA

Sería imposible no dedicarle este trabajo a la persona que más me ha querido y que aunque no pudo ver uno de sus principales sueños hecho realidad debido a que la vida no se lo permitió, este trabajo, especialmente, se lo dedico a mi abuelita linda Alicia Guerrero, por ser una de las personas a quien le debo todo lo que soy.
A mis padres y familiares por toda la confianza que siempre han tenido en mí y por ser una de las mejores familias que desearía un ser humano.

Yendri.

Dedico este trabajo a mis padres Enrique y Giselda, a mis hermanos Yosvani y Tinito, a mis tíos y tías, a Ana mi abuela y al viejo Chimbo, a mi primo Yudiel, y a mi madrina Mayra.

Henry.

RESUMEN

El desarrollo alcanzado por el software se ha convertido en el pilar más importante en la realización de productos informáticos, este ha pasado de ser una resolución de problemas y herramienta de análisis de información, a ser una industria por si misma. Su desarrollo en el país ha contribuido a la informatización de los procesos que se realizan, logrando con ello una mayor rapidez en la ejecución y confiabilidad, es decir una mayor calidad en la seguridad y procesamiento de la información de las actividades. En la Universidad de las Ciencias Informáticas, la Cátedra de Ajedrez decidió informatizar en un gran porciento las actividades relacionadas con el ajedrez creando el proyecto Infodrez, entre estas actividades se encuentran todos los procesos que los árbitros realizan en el transcurso de un torneo, normalmente estos procesos se realizan manualmente por lo que decidió conformar el Módulo Arbitraje dentro del proyecto, con el objetivo de informatizar todos estos procesos. Para ello se desarrollará un sistema informático utilizando el lenguaje PHP, el gestor de Bases de Datos MySQL y el uso de una herramienta CASE como Visual Paradigm. Se aplicará RUP como metodología para el desarrollo del software.

PALABRAS CLAVES

Ajedrez, Arbitraje, Torneo, Pareo, Sorteo, Árbitros.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS.

ÍNDICE DE FIGURAS.

INTRODUCCIÓN

La Informatización de la Sociedad es el proceso de utilización ordenada y masiva de las Tecnologías de la Información y las Comunicaciones en la vida cotidiana, para satisfacer las necesidades de todas las esferas de la sociedad, en su esfuerzo por lograr cada vez más eficacia y eficiencia en todos los procesos y por consiguiente mayor generación de riqueza y aumento en la calidad de vida de los ciudadanos.

Una sociedad que aplique la informatización en todas sus esferas y procesos será más eficaz, eficiente y competitiva. Es evidente que para los países subdesarrollados resulta un reto el logro de este propósito, ya que su problemática fundamental está en lograr la supervivencia de sus pueblos.

Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones; y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría a la sociedad acercarse más hacia el objetivo de un desarrollo sostenible.

Dentro de esta tarea de informatización de la sociedad cubana está inmersa la Universidad de las Ciencias Informáticas. Como parte de esto se llevan a cabo proyectos conjuntos con múltiples organismos y organizaciones de la sociedad y el estado, así como con instituciones pertenecientes al centro, tal es el caso del proyecto Infodrez que se desarrolla en conjunto con la Cátedra de Ajedrez de la UCI.

En la misma se encontró la siguiente **situación problemática:**

En la actualidad, la Cátedra de Ajedrez Honorífica “Remberto Fernández“ de la Universidad de las Ciencias Informáticas (UCI) no cuenta con un sistema automatizado que le permita gestionar todas las actividades que en esta se realizan. Como parte de la información que es necesario automatizar dentro de la cátedra de ajedrez, está todo lo referente a los árbitros y algunas de sus variadas funciones.

Este sistema permitirá realizar una serie de funciones que los árbitros realizan manualmente, como es el caso de realizar sorteo, realizar pareo, actualizar resultados de las partidas, entre otras actividades. Para dar solución a esto se pretende crear un módulo dentro del gestor de contenidos de ajedrez que permita manejar todos los contenidos relacionados al arbitraje.

Este trabajo surge como necesidad de dar solución a las situaciones antes expuestas; por lo que el **problema** a solucionar en él, consiste en: ¿Cómo dotar la Cátedra de Ajedrez de la UCI, Remberto Fernández, de una herramienta, que permita la gestión de contenidos de arbitraje?

Por tanto el **objeto de estudio** de este trabajo es el proceso de desarrollo de un software que gestione los contenidos relacionados con el proceso arbitral. Estos estarán a disposición de todos los árbitros, ya que serán los principales usuarios de la aplicación.

De lo anteriormente planteado se deriva que el **campo de acción** que abarca este trabajo, es la automatización de la gestión de los diferentes contenidos relacionados con los árbitros, y permitirles que organicen estos contenidos dentro de la aplicación Web.

Como **idea a defender** tenemos que si se desarrolla una herramienta para la gestión de contenidos, es posible lograr la gestión de una aplicación Web que permita la automatización de todo el proceso arbitral.

El **objetivo de la investigación** de este trabajo será desarrollar una herramienta que permita gestionar los contenidos relacionados con el arbitraje en la UCI.

Para cumplir con este objetivo y resolver la situación problemática planteada, se proponen las siguientes **tareas**:

- Investigar con los Árbitros de la Cátedra de Ajedrez de la UCI “Remberto Fernández” sobre los procesos arbitrales.
- Realizar un estudio detallado de las tendencias y tecnologías existentes para la gestión de contenidos arbitrales.
- Estudiar las metodologías existentes para el análisis y diseño de aplicaciones con tecnologías Web.
- Comparar estas metodologías teniendo en cuenta sus características, ventajas y desventajas.
- Seleccionar la metodología que se adapte a las características de la herramienta a implementar.
- Estudiar las herramientas existentes para la implementación de aplicaciones con tecnologías Web.

- Comparar estas herramientas teniendo en cuenta sus características, ventajas y desventajas.
- Seleccionar la herramienta que se adapte a las características del sistema a implementar.
- Desarrollar análisis, diseño e implementación del sistema con la metodología y la herramienta seleccionadas.

Este trabajo está estructurado en 4 capítulos, en los cuales se tratan las siguientes cuestiones:

Capítulo 1. Fundamentación Teórica: Este capítulo trata acerca de algunos elementos teóricos que sirven de soporte para la realización de todo el trabajo en general, tales como: algunos sistemas similares existentes vinculados al campo de acción, además de una descripción y selección de las herramientas a utilizar.

Capítulo 2. Características del Sistema: En dicho capítulo se realiza una descripción del objeto de estudio, además de que se describen todos los procesos, actores, casos de usos y trabajadores del negocio actual. También se muestra el diagrama de casos de usos del negocio y el modelo de objetos del negocio. Por otro lado, se definen cuáles son los requerimientos funcionales y no funcionales, a la vez que se presentan los actores del sistema y diagrama de casos de uso del mismo, acompañado de la descripción textual de cada uno de estos.

Capítulo 3. Análisis y Diseño del Sistema: En este capítulo se presentan los diagramas de clases del análisis, lo cual influye notablemente a la hora de concebir el diseño del sistema, además de los diagramas de clases Web, que reflejan de una forma más clara cómo va a funcionar dicho sistema y qué clases están presentes en el mismo, acompañado también de la descripción de cada una de estas.

Capítulo 4. Implementación: En este capítulo se describe como está implementado el sistema, a través de los diagramas de componentes y el diagrama de despliegue.

Al final se podrán observar las conclusiones generales, las recomendaciones, citas bibliográficas hechas a lo largo del trabajo, así como, un glosario de términos y siglas utilizadas. Por último, se podrán observar los anexos, los cuales contienen información de apoyo a algunos aspectos tratados durante el trabajo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

El siguiente capítulo aborda la fundamentación teórica acerca del módulo de Arbitraje perteneciente al Sistema Gestor de Contenidos de Ajedrez, en este se realiza un estudio de los sistemas similares existentes, además se presentan las tendencias y tecnologías actuales y se realiza una selección de aquellas que serán utilizadas durante el desarrollo del proyecto.

1.2 Estado del Arte.

El ajedrez es un juego de mesa para dos personas. Más precisamente, es un juego de guerra. Es uno de los juegos más populares del mundo. Se considera no sólo un juego, sino un arte, una ciencia y un deporte mental. Esto último es muy apropiado, dado que se juega a menudo de forma competitiva.

Arbitraje:

En Derecho: Las partes, de mutuo acuerdo, deciden nombrar a un tercero independiente, denominado árbitro, y que será el encargado de resolver el conflicto. El árbitro, a su vez, se verá limitado por lo pactado entre las partes para dictar el laudo arbitral. Deberá hacerlo conforme a la legislación que hayan elegido las partes, o incluso basándose en la simple equidad, si así se ha pactado.

Cuando un arbitraje se ajusta a la legalidad, sustituye completamente a la jurisdicción ordinaria, que deberá abstenerse de conocer el litigio. Sin embargo, sí que será necesario acudir a la misma (a través de la acción ejecutiva) cuando sea necesaria la intervención de las autoridades para hacer cumplir el laudo arbitral.

Árbitro de Ajedrez:

Es aquella persona encargada de velar por que se haga cumplir lo establecido por el reglamento del arbitraje dentro de los torneos ajedrecísticos, y para dar cumplimiento a esto, el arbitro debe realizar un numero de procesos que son el objeto de estudio de este trabajo.

1.2.1 Sistemas que gestionan el Arbitraje (Internacional)

En la actualidad en el ámbito internacional existen muchos sitios que publican contenidos relacionados con los procesos arbitrales, dichos sitios son de gran valor dado que publican la información necesaria para que los árbitros se mantengan actualizados; algunos ejemplos de estos sitios son el sitio oficial de FIDE (www.FIDE.com) y el de la federación española de Ajedrez (www.FEDA.org). Ambos sitios y otros menos relevantes a pesar de tratar bastante a fondos todos los temas relacionados con el ajedrez, aún tienen algunas carencias en los temas relacionados con el arbitraje, aunque cuentan con sesiones específicas que tratan estos temas. El portal que lleva la delantera en los temas específicos del arbitraje de ajedrez es (www.arbitrosdeajedrez.com) que a pesar de estar ahora en una etapa de mantenimiento se centra en los temas de arbitraje únicamente.

A pesar de existir estos sitios antes mencionados, en ellos, no se realiza ninguno de los procesos arbitrales; sino que se encargan de dar información acerca de los mismos; existe una herramienta que es la encargada de realizar la mayoría de estos procesos que es el Swiss-Manager, pero que posee la gran limitación de ser software propietario y una herramienta desktop.

1.2.2 Sistemas que gestionan el Arbitraje (Nacional)

A nivel nacional se cuenta actualmente con el sitio de ajedrez en Cuba (www.cuba.cu/ajedrez), en el que al igual que los sitios de las federaciones internacionales, se trata el tema del arbitraje muy superficialmente y los árbitros en la mayoría de los casos tienen que recurrir a los sitios externos o al Swiss-manager.

1.2.3 Sistemas que gestionan el Arbitraje (UCI)

En nuestra universidad contamos con el portal Infodrez en el cual se tratan bastante a fondos temas relacionados con el ajedrez pero aún persiste la carencia de una sesión especializada en los temas del arbitraje porque aunque el portal de Infodrez cuenta con una sesión dedicada al mismo, esta solo se encarga de publicar noticias relacionadas con el tema.

¿Por qué esta propuesta?

La razón fundamental de esta propuesta radica en dotar a los árbitros de una herramienta que les permita automatizar todos lo procesos posibles, siendo esta herramienta software libre, y soportada sobre una plataforma Web integrada (Infodrez) que contará con otros temas relacionados con el ajedrez que le resultarán muy útiles a los árbitros.

1.3 Solución a través de un sistema de gestión Web.

Una aplicación Web es aquella que los usuarios usan accediendo a un servidor Web a través de la red (Internet o intranet). Las aplicaciones Web son populares debido a la practicidad del navegador como cliente ligero. La habilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad.

Las interfaces de las páginas Web poseen ciertas limitantes en la funcionalidad del cliente. Métodos comunes en las aplicaciones de escritorio como dibujar en la pantalla, arrastrar o soltar no están soportados por las tecnologías Web estándar. Los desarrolladores utilizan generalmente lenguajes interpretados que se ejecutan en el cliente Web para obtener una mayor funcionalidad, así como tecnologías que se ejecutan en el servidor para no tener que recargar la página en su totalidad, algo que molesta mucho a los usuarios. Recientemente se han desarrollado técnicas para coordinar estos lenguajes con tecnologías del lado del servidor, como por ejemplo PHP y AJAX que es una técnica de desarrollo Web que usa una combinación de varias tecnologías como se verá más adelante.

1.3.1 Arquitectura Cliente-Servidor.

La arquitectura cliente-servidor es una relación entre procesos corriendo en máquinas separadas. El servidor es un proveedor de servicios. El cliente es un consumidor de servicios. Cliente y servidor interactúan por un mecanismo de pasaje de mensajes, el cliente realiza un “Pedido de servicio” y a su vez el servidor da una “Respuesta”.

Esta arquitectura es una forma de dividir las responsabilidades de un Sistema Informático separando la interfaz de usuario (Nivel de presentación) de la gestión de la información (Nivel de gestión de datos). (DESARROLLOWEB, 2005)

1.4 Técnicas y tecnologías del lado del cliente.

Las técnicas y tecnologías del lado del cliente son aquellas que se utilizan para desarrollar los programas que se ejecutaran en los mismos, entiéndase por cliente a los navegadores como: Firefox, Internet Explorer, Safari, Konqueror, Opera, Netscape Navigator etc. Estos clientes son los encargados de visualizar la información que han solicitado las personas, dicha información el navegador es capaz de leerla debido a que este interpreta código HTML, y algunos lenguajes script como son VBScript, Java Script. (DESARROLLOWEB, 2005)

1.4.1 Lenguajes del lado del cliente.

El Cliente es un ordenador que accede a recursos y servicios brindados por otro llamado Servidor, generalmente en forma remota. El cliente recibe los servicios que ofrece un servidor. El término se usó inicialmente para dispositivos que no eran capaces de ejecutar programas por sí mismos, pero podían interactuar con ordenadores remotos por red. Estos terminales eran clientes de los ordenadores centrales de tiempo compartido.

JavaScript

Es el lenguaje que permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas Web un gran dinamismo. Se trata de un lenguaje de tipo script compacto, basado en objetos y guiado por eventos diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.

Los programas JavaScript van dentro de los documentos HMTL, y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos.

La característica principal de JavaScript, de hecho, es la de ser un lenguaje de scripting, pero, sobre todo, la de ser el lenguaje de scripting por excelencia y, sin lugar a dudas, el más usado. Esta particularidad conlleva una notable serie de ventajas y desventajas según el uso que se le deba dar y teniendo en cuenta la relación que se establece entre el mecanismo cliente-servidor. (EGUILUZ PEREZ, 2004)

¿Cuáles son las ventajas y cuáles las desventajas respectivas de los lenguajes de scripting y los lenguajes compilados? Algunos ejemplos.

- Un problema importante es que el código es visible y puede ser leído por cualquier persona, incluso si está protegido con las leyes del copyright.
- El código del script debe descargarse completamente antes de poderse ejecutar. Si los datos que un script utiliza son muchos, el tiempo que tardará en descargarse será muy largo, mientras que la interrogación de la misma base de datos en el servidor sería más rápida.
- Al ejecutar un código JavaScript del lado del cliente se está liberando de carga al servidor lo que proporciona que las aplicaciones sean más rápidas.

¿Por qué JavaScript?

Debido a su gran compatibilidad con todos los navegadores modernos es sin dudas el script más utilizado del lado del cliente. Con este lenguaje a parte de definir interactividades con el usuario se pueden crear efectos especiales en las páginas los cuales son interpretados y ejecutados por el navegador de modo que el único recurso que se hace indispensable para ejecutar un código JavaScript es el navegador Web.

Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del Modelo de Objetos de Documento, frecuentemente abreviado DOM, es una interfaz de programación de aplicaciones (en lo adelante API) para documentos HTML (páginas Web) y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.

En la especificación del DOM, el término "documento" se utiliza en un sentido amplio ya que es el documento el contenedor que soporta los demás elementos. A través del DOM los programadores pueden construir documentos, navegar por su estructura, añadir, modificar o eliminar elementos y contenido. Se puede acceder a cualquier elemento que se encuentre en un documento HTML o XML, y se puede modificar, eliminar o añadir usando DOM, salvo algunas excepciones.

HTML (HyperText Markup Language)

Es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces (hipervínculos) que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia (gráficos, sonido, entre otros). Este lenguaje es el que se utiliza para presentar información en la World Wide Web. La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.), así como los diferentes efectos que se quieren dar (cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por un programa especializado. (MENDEZ, 1996)

XML (Extensible Markup Language).

XML es un lenguaje de marco que ofrece un formato para la descripción de datos estructurados, esto facilita una organización más precisa de los contenidos y unos resultados de búsquedas más significativos en varias plataformas. Además, XML habilita una nueva generación de aplicaciones para ver y manipular datos basadas en la Web. Se trata de una tecnología sencilla que tiene a su alrededor

otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. (HIPERTEXT, 2007)

Hojas de Estilo en Cascada (Cascading Style Sheets).

Son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). La virtud del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style". (CONCLASE, 1999)

Las ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio Web completo con lo que se agiliza de forma considerable la actualización de la misma.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio Web remoto. Así por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

1.5 Tecnologías del lado del servidor.

Las tecnologías del lado del servidor son aquellos programas o servicios que corren en un servidor remoto y que brindan funcionalidades al sistema.

PHP

Es un lenguaje de programación usado generalmente para la creación de contenido para sitios web. PHP es un acrónimo recurrente que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web. (PHP, 1999)

Python

Es un lenguaje interpretado, lo que ahorra un tiempo considerable en el desarrollo de programas, pues no es necesario compilar ni enlazar con librerías que usen la memoria de la computadora. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa. (HETLAND, 1999)

Perl

Es un lenguaje de propósito general, inicialmente fue desarrollado para la manipulación de texto, pero actualmente es utilizado para el desarrollo de varias tareas, entre las que se encuentran administración de sistemas, desarrollo Web, programación en red y desarrollo de GUI. (CASTELLANO, 2002)

Es fácil de usar, eficiente y completo. Soporta tanto la programación estructurada como la programación orientada a objetos y la programación funcional, tiene incorporado un poderoso sistema de procesamiento de texto y una enorme colección de módulos disponibles.

¿Por qué PHP?

Para el desarrollo de la aplicación Web se debe tener en cuenta cuestiones esenciales como: garantizar que el sistema sea multiplataforma y que pueda ser desarrollado en software libre por las características del país, ya que este está bloqueado y de las restricciones de las principales compañías de desarrollo informático. Analizando esto el más adecuado es el PHP por ser libre, multiplataforma, su flexibilidad de comunicación con los principales gestores de bases de datos y por su potencialidad en funcionalidades y rapidez.

Ventajas de su uso:

- Es un lenguaje multiplataforma y permite utilizar programación Orientada a Objetos.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad como son: InterBase, mSQL, MySQL, Oracle, Informix, PosgreSQL.
- Permite leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Es un leguaje libre de licencias de uso, por lo que se presenta como una alternativa de fácil acceso para todos los desarrolladores.

- Permite crear los formularios para la web.
- Biblioteca nativa de funciones sumamente amplia que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en .pdf hasta analizar código XML.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

1.6 Combinación de varias tecnologías.

JavaScript y XML Asíncronos (Asynchronous JavaScript And XML, en lo adelante AJAX) es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tres tecnologías ya existentes:

- XHTML (o HTML) y CSS para el diseño que acompaña a la información.
- DOM accedido con un lenguaje de scripting por parte del usuario, como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano.

Propiedades.

- Peticiones y respuestas HTTP.
- Código de lado del cliente usando JavaScript.
- Programación de lado del servidor en PHP.
- Transferir y procesar datos mediante el uso de XML.

1.7 Gestor de Base de Datos.

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes y acrónimos respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, por sus siglas en inglés.

PostgreSQL

Servidor de base de datos relacional libre, liberado bajo la licencia BSD. Es una alternativa a otros sistemas de bases de datos de código abierto (como MySQL, Firebird y MaxDB), así como sistemas propietarios como Oracle o DB2. (SORIANO, 1998)

MySQL

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario desarrollado como software libre, es el sistema de base de datos open source más popular (MySQL, 2008). Dentro de las características más importantes de este software de base de datos están:

- Es desarrollado en C y C++.
- Funciona en diferentes plataformas.
- Es compatible con lenguajes de programación como C, C++, Java, Perl, PHP, Python.
- Proporciona sistemas de almacenamientos transaccionales y no transaccionales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y debe ser tan rápida como sea posible.

¿Por qué MySQL?

A partir de las características que ofrecen los distintos SGBD se decidió; escoger para la realización de la BD de la aplicación a MySQL como gestor, ya que es un fuerte sistema gestor de BD de código abierto. A través de este es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla. Permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix. Es multihilo, con lo que puede beneficiarse de sistemas multiprocesador. Permite manejar multitud de tipos para columnas, así; como registros de longitud fija o variable.

1.8 Servidor de aplicaciones Web Apache.

El servidor HTTP Apache es un software gratuito de código fuente abierto que funciona sobre una multitud de sistemas operativos como: Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementan el protocolo HTTP/1.1. Es un servidor altamente configurable de diseño modular, es muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos para Apache que son

adaptables a este. Apache trabaja con PHP y otros lenguajes de script. También trabaja con Java. Teniendo todo el soporte que se necesita para tener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. (WIKIPEDIA, 2008)

1.9 Metodologías de desarrollo de software.

Todo desarrollo de software se torna riesgoso y difícil de controlar, pero si no se utiliza una metodología, existe una mayor exposición a obtener clientes y desarrolladores insatisfechos con el resultado. Muchas veces no se toma en cuenta la utilización de una metodología adecuada y por esta causa se desarrolla el software solo tomando en cuenta los requerimientos del cliente de manera que en la etapa final si este solicita algún cambio en los requerimientos se hace muy difícil de realizar y es justo éste, uno de los factores que ocasiona un atraso en el proyecto y por tanto la incomodidad del desarrollador por no cumplir con el cambio solicitado y el malestar por parte del cliente por no tomar en cuenta su pedido.

Obviamente para evitar estos incidentes se debe llegar a un acuerdo formal con el cliente, al inicio del proyecto, de tal manera que cada cambio o modificación no perjudique el desarrollo del mismo, así como escoger una metodología adecuada para el desarrollo del software que sirva como guía para realizar de forma disciplinada y eficiente el producto deseado. (PENIN)

1.9.1 Programación Extrema (XP).

Es un enfoque de la ingeniería de software que pone más énfasis en la adaptabilidad que en la previsibilidad. Ésta metodología promueve la idea de que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Las características fundamentales de esta metodología son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas.
- Programación en parejas.
- Frecuente integración del equipo de programación con el cliente.
- Corrección de todos los errores.
- Refactorización del código.
- Propiedad del código compartida.

- Simplicidad.

1.9.2 El Proceso Unificado de Modelado (RUP).

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- Inicio: El Objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- Transición: El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. (WIKIPEDIA, 2008)

El ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

Disciplina de Desarrollo:

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

Disciplina de Soporte:

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- Actividades: Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: Vienen hacer las personas o gentes involucradas en cada proceso.
- Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Características del Proceso Unificado.

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Representan requerimientos funcionales. Los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. La arquitectura se refleja en los casos de uso pues cada producto tiene tanto una función como una forma, ninguna es suficiente por si sola.

Iterativo e Incremental: RUP propone dividir el trabajo en partes más pequeñas o mini proyectos, donde cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son miniproyectos.

Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación las cuales están estrechamente relacionadas, cuyo resultado es una versión del software.

Beneficios de la iteración:

- Reduce el coste del riesgo al coste de un solo incremento.
- Menos riesgo de no sacar el producto al mercado en fecha.
- Acelera el ritmo de desarrollo.
- Las necesidades del usuario y correspondientes requisitos no pueden definirse completamente

al principio. Se requieren iteraciones sucesivas.

1.10 Herramientas CASE.

La tecnología CASE (WIKIPEDIA, 2008) supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información y se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento de los programas.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

Automatizar:

- El desarrollo del software.
- La documentación.
- La generación del código.
- El chequeo de errores.
- La gestión del proyecto.

Permitir:

- La reutilización del software.
- La portabilidad del software.
- La estandarización de la documentación.

Visual Paradigm

Es una herramienta CASE que ofrece un entorno de creación de diagramas para UML, diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad; uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación; capacidades de

ingeniería directa (versión profesional) e inversa; modelo y código que permanece sincronizado en todo el ciclo de desarrollo; disponibilidad de múltiples versiones, para cada necesidad; disponibilidad de integrarse en los principales IDEs; disponibilidad en múltiples plataformas, y muy útil para la generación de código fuente en PHP, también con el Paradigm se generan script de las tablas de salidas para las clases persistentes. (Hernandis, 2005)

Beneficios:

- Navegación intuitiva entre el modelo visual y el código.
- Poderosa herramienta de generación de PDF/HTML a partir de diagramas UML.
- Sincronización entre el código fuente y el modelo en tiempo real o bajo demanda.
- Entorno visual de modelado superior.
- Soporte para toda la notación UML.
- Sofisticados y automáticos diagramas de capas.
- Análisis de textos.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la
- comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

1.11 Herramientas de desarrollo.

1.11.1 Zend Development Environment

Este IDE, acrónimo de Integrated Development Environment es una de las herramientas más potente de programación para el lenguaje PHP. Cuenta con analizadores de código, permite completamiento de código entre otras ventajas. El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. (WIKIPEDIA, 2007)

Lo más destacable es que contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento ofreciendo nombres de las funciones y parámetros que deben

recibir. El hecho de que esté desarrollado sobre Java da la ventaja de que puede ser utilizado en cualquier sistema operativo.

1.11.2 NuSphere PHPEd.

Es un potente editor de PHP tanto para personas experimentadas como para principiantes, con soporte, con soporte para múltiples formatos. Resalta los distintos tags con diferentes colores para hacer mucho más fácil la programación. Esta herramienta incluye un cliente de FTP (File Trasfer Protocol) y un servidor Web integrados, que se configuran según las necesidades del trabajo que se vaya a realizar.

1.12 Arquitectura.

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.
La Arquitectura de Software establece los fundamentos para que analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

Esta se selecciona y diseña con base en unos objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura software de tres capas para implementar sistemas en tiempo real.

La arquitectura software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura software debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

Generalmente, no es necesario inventar una nueva arquitectura software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

Monolítica: Donde el software se estructura en grupos funcionales muy acoplados.

Cliente-servidor: Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.

Arquitectura de tres niveles: Generalización de la arquitectura cliente-servidor donde la carga se divide en tres partes con un reparto claro de funciones: una capa para la presentación, otra para el cálculo y otra para el almacenamiento. Una capa solamente tiene relación con la siguiente.

Modelo Vista Controlador (MVC): es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

- **Modelo**: Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos; por ejemplo, calculando si hoy es el cumpleaños del usuario o los totales impuestos en un carrito de compra.

- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.

- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Muchas aplicaciones utilizan un mecanismo de almacenamiento persistente (como puede ser una base de datos) para almacenar los datos. MVC no menciona específicamente esta capa de acceso a datos.

Es común pensar que una aplicación tiene tres capas principales: presentación (IU), dominio, y acceso a datos. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre vista y controlador es menos clara.

Arquitectura n-Capas: La que más comúnmente se usa es la de cuatro capas, la capa que se agrega es la que surge de separar definitivamente las reglas de negocio de la de Acceso a Datos. Esta arquitectura trae consigo la ventaja de aislar definitivamente la lógica de negocios de todo lo que tenga que ver con el origen de datos, ya que desde el manejo de la conexión, hasta la ejecución de una consulta, la manejará la capa de Acceso a Datos. De este modo, ante cualquier eventual cambio, solo

se deberá tocar un módulo específico, así como al momento de plantear la escalabilidad de este sistema, si se han respetado las reglas básicas de diseño, entonces no se deberán afrontar grandes modificaciones.

- Abstracción total acerca del origen de datos**.** Las distintas capas se especializan absolutamente en la funcionalidad que deben brindar (procesamiento en las reglas de negocios o presentación de datos en la capa cliente) sin importar cual es el origen de los datos procesados.

- Bajo costo de desarrollo y mantenimiento de las aplicaciones. Si bien al momento del diseño podemos observar una mayor carga de complejidad, la utilización de esta arquitectura brinda un control más cercano de cada componente, así como también la posibilidad de una verdadera reutilización del código.
- Estandarización de las reglas de negocio. Las reglas de negocio se encuentran encapsuladas en un set de rutinas comunes y pueden ser llamadas desde diversas aplicaciones sin necesidad de saber cómo esta funciona o ha sido diseñada.

- Mejor calidad en las aplicaciones. Como las aplicaciones son construidas en unidades separadas, estas pueden ser testeadas independientemente y con mucho más detalle, esto conduce a obtener un producto mucho más sólido.

- Reutilización de código. La concepción natural de un sistema desarrollado con esta arquitectura, promueve la reutilización de sus componentes en varias partes del propio desarrollo y de futuros sistemas.

Conclusiones.

En este capítulo se detallaron las condiciones y problemas que rodean el objeto de estudio; y a través de los conceptos y definiciones planteadas, se determinaron las condiciones específicas que envuelven al mismo. Se analizaron las características de diferentes herramientas, para la creación de un software o aplicación así como algunas metodologías.

Después de este análisis y la fundamentación realizada, el lenguaje de programación que se utilizará es PHP y JavaScript para el control del las diferentes funciones a realizar en el cliente, como gestor de base de datos se estableció el MySQL.

Se empleará la metodología RUP por ser un proceso iterativo e incremental, que permite dirigir el proceso por casos de uso y tener un mejor rectoreo de la calidad en cada etapa del proceso, como

lenguaje de modelado UML y la herramienta que se usará como entorno de creación de diagramas para UML el Visual Paradigm.

Se decidió el uso de la arquitectura n-Capas, debido a que permite trabajar de una forma más organizada, lo cual trae grandes beneficios a la hora de hacer cambios y corregir errores. Una vez conocidas las herramientas óptimas y los conceptos a utilizar se puede empezar a desarrollar la propuesta de sistema.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción.

En el campo del software resulta de mucha utilidad la creación de modelos que organicen y presenten los detalles importantes de problemas reales que se vinculan con el sistema informático a construir.

Como se ha expresado anteriormente el objetivo del Proceso Unificado, es guiar a los desarrolladores de cualquier sistema software en la implementación y distribución eficiente de sistemas que se ajusten a las necesidades de los clientes.

En el presente capítulo, se dará una visión de lo que funcionalmente debe realizar el sistema con el objetivo de erradicar los problemas anteriormente mencionados. Se transitará a lo largo de la fase de inicio por cada uno de los flujos de trabajo desde el negocio hasta el sistema. Todo esto se desarrollará por medio de un conjunto de artefactos y modelos resultantes de la aplicación de la metodología de desarrollo de software.

Estos modelos deben cumplir una serie de propiedades entre las que se encuentran la de ser coherentes y que tengan relación con el resultado final del producto. Uno de los modelos útiles previo al desarrollo de un software es el modelo del negocio.

2.2 Propuesta del sistema.

Para informatizar el trabajo del árbitro el sistema debe comenzar autenticando a estos como administradores del sistema, si éste se encuentra arbitrando algún torneo el sistema debe permitirle trabajar con ese torneo de lo contrario debe permitirle crear un nuevo torneo. Una vez creado, el sistema debe posibilitar la creación de las bases técnicas.

Cuando se inicia un torneo del tipo Todos contra Todos se hace un sorteo donde se le asigna a cada jugador participante un número aleatorio por lo que el sistema debe permitirle al árbitro realizar esta acción. Acto seguido se realiza el pareo que es donde se le dice al jugador sus oponentes, en que ronda se enfrentarán y con que piezas juegan.

El sistema debe permitir mostrar el cuadro sinóptico del torneo y para esto le brindará la opción de que actualice cada una de las partidas de forma independientes. Una vez terminada al menos una ronda del torneo este debe brindar la posibilidad de conocer las posiciones de los jugadores dentro del torneo.

2.3 Modelado del Negocio.

El modelado del negocio es una técnica para comprender y definir los procesos del negocio, roles y responsabilidades de la organización en los modelos de casos de uso del negocio y de objetos.

Los propósitos que se persiguen al modelar el negocio son:

- Comprender la dinámica de la organización en la cual se va a implementar el sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los usuarios y desarrolladores tengan un entendimiento común de la organización.

Por tanto la finalidad del modelado del negocio es describir cada proceso del negocio especificando sus datos, actividades, roles y reglas del negocio.

Este proceso está soportado por dos tipos de modelos de UML: modelos de casos de uso y modelo de objetos.

Un modelo de casos de uso del negocio describe los procesos de una empresa en términos de casos de uso y actores del negocio.

Un modelo de objetos del negocio es un modelo que describe cómo colaboran los trabajadores y las entidades del negocio dentro del flujo (realización).

Para lograr un mejor entendimiento del modelado del negocio, se presenta una descripción de cuales serán los actores y trabajadores del mismo.

2.4 Actor y Trabajadores del Negocio.

Un actor puede representar a una persona física, otro sistema, un dispositivo, siendo siempre un tercero fuera del sistema que colabora con él. La definición de actores, sirve para definir el contexto externo del sistema, esto es, delimitar los elementos que se encuentran fuera y dentro del mismo, y un actor juega un rol para cada caso de uso en el que colabora.

Tabla 1: Descripción de los actores del negocio.

Actores del negocio	Justificación
Director del torneo	Es quien exige al árbitro que estén todas las condiciones creadas para la realización del torneo.
Federación	Es la organización rectora de todos los eventos relacionados con el ajedrez a la cual se le entregan todos los reportes oficiales de los diferentes torneos.
Organizador	Es la persona encarga de organizar todos los torneos ajedrecísticos.
Jugador	Son los protagonistas en una partida de ajedrez.

El trabajador, por su parte, representa una posición que se puede asignar a una persona o equipo en una organización de desarrollo de software, y para cada trabajador se especifican las responsabilidades y habilidades requeridas, ya que este define el comportamiento y las responsabilidades de un individuo.

Tabla 2: Descripción de los trabajadores del negocio.

Trabajadores del negocio	Justificación
Árbitro	Es la persona encargada de velar por el cumplimiento de las leyes ajedrecísticas dentro del desarrollo de los torneos de este deporte.

2.5 Diagrama de Casos de Uso del Negocio.

Con el objetivo de obtener una visión general de los diferentes procesos del negocio se realiza el diagrama de casos de uso del negocio, donde cada proceso se representa como un caso de uso y solo aparecerán los actores del negocio correspondientes a los roles externos al sistema este diagrama permite mostrar los límites y el entorno de la organización (Ver figura 1).

Diagrama de Casos de Uso del Negocio

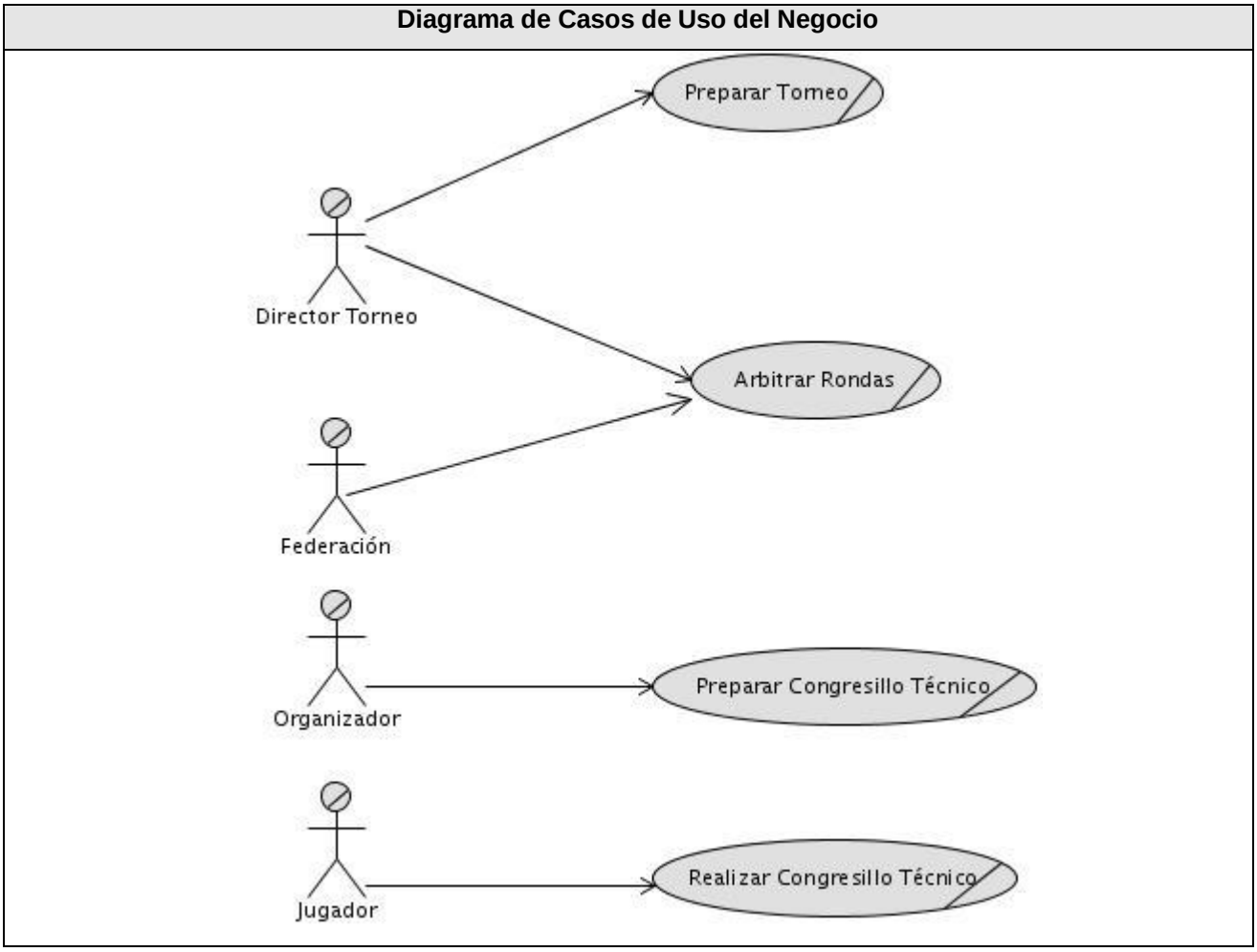

Figura 1: Diagrama de Casos de Uso del Negocio.

2.6 Descripciones Textuales.

La descripción textual de los casos de uso del negocio se formaliza en un documento generalmente llamado “Especificación del caso de uso de negocio”. Para realizar la misma, inicialmente se rellena una plantilla de descripción, y después, a partir de la información reflejada en dicha plantilla, se construye un conjunto de diagramas (diagramas de actividades) que describen completamente el caso de uso del negocio.

2.6.1 Descripción Textual del CUN Preparar Congresillo Técnico.

Tabla 3: Descripción textual del CUN Preparar Congresillo Técnico.

Caso de Uso	**Preparar Congresillo Técnico**
Actores	Organizador (inicia).
Propósito	Garantizar las condiciones necesarias para el buen desarrollo del torneo, tanto desde el punto de vista técnico como organizativo.
Resumen	El CUN se inicia cuando el Organizador le entrega la convocatoria del torneo y el listado de jugadores al Árbitro Principal. Este recibe lo entregado por el Organizador y actualiza los datos de los jugadores, ordenándolos por ELO. Una vez terminada esta acción elabora las Bases Técnicas del Torneo y de esta forma garantiza las condiciones para realizar el sorteo de los jugadores quedando terminado el CUN.
Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. El Organizador entrega la convocatoria del torneo y el listado de jugadores.	2. Recibe la convocatoria del torneo y el listado de jugadores.
	3. Actualiza los datos de los jugadores por ELO.
	4. Elabora Bases Técnicas del Torneo.
	5. Garantiza las condiciones necesarias para realizar el sorteo de los jugadores y termina el CUN.

2.6.2 Descripción Textual del CUN Realizar Congresillo Técnico.

Tabla 4: Descripción textual del CUN Realizar Congresillo Técnico.

Caso de Uso	**Realizar Congresillo Técnico**
Actores	Jugador (inicia).
Propósito	Arbitrar las rondas de un torneo.
Resumen	El CUN se inicia cuando el Jugador asiste al Congresillo Técnico, acto seguido el Árbitro Principal comenta las bases técnicas del torneo, conforma el comité de apelaciones y pasa a realizar el sorteo y de ser necesario el pareo forzado una vez culminadas estas acciones informa el número de sorteo de cada jugador y termina el CUN.
Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. Asiste al Congresillo Técnico.	2. Comenta las Bases Técnicas del Torneo.
3. Acepta las Bases Técnicas del Torneo.	4. Conforma el Comité de Apelaciones.
	5. Pide al jugador que escoja el número de sorteo.
6. Escoge el número de sorteo.	7. Registra el orden final de los jugadores en las tablas de pareo.
	8. Informa los números de cada jugador y finaliza el CUN.
Flujo Alternativo 1	
3. No acepta las Bases Técnicas del Torneo.	4. Pregunta por nuevos puntos a incluir o cambiar en las Bases Técnicas del Torneo.
5. Propone los puntos a incluir o cambiar en las Bases Técnicas del Torneo.	6. Acepta los nuevos puntos o explica por qué estos no son aceptados.
7. Se procede al flujo normal de eventos desde la actividad número 4.	

Flujo Alternativo 2	
4. Problemas para jugar una determinada ronda.	5. Pide al jugador que escoja el primer número del sorteo.
6. Se procede al flujo normal de eventos desde la actividad 6.	

2.6.3 Descripción Textual del CUN Preparar Torneo.

Tabla 5: Descripción textual del CUN Preparar Torneo.

Caso de Uso	**Preparar Torneo**
Actores	Director del Torneo (inicia).
Propósito	Preparar todas las condiciones físicas necesarias para el desarrollo del torneo.
Resumen	El CUN se inicia cuando el Árbitro Principal chequea todas las condiciones del local, dígase ambientación, implementos de juegos, condiciones de los baños, chequea que las listas de los jugadores estén con el ELO verdadero, además de que estén todas las condiciones para llevar los datos del torneo y así queda terminado el CUN.
Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. El Director del Torneo exige que estén todas las condiciones creadas.	2. El Árbitro Principal chequea condiciones del local de juego.
.	3. El Árbitro Principal chequea que estén las mesas necesarias.
	4. El Árbitro Principal chequea implementos de juego y su correspondencia con los participantes.
	5. El Árbitro Principal provee un salón de análisis.
	6. El Árbitro Principal acondiciona la sala de juegos

	7. El Árbitro Principal cuenta con una computadora garantiza el programa de arbitraje y termina el CUN.
Flujo Alternativo	
	7. El Árbitro Principal no cuenta con una computadora garantiza condiciones para llevar los datos del torneo y termina el CUN.

2.6.4 Descripción Textual del CUN Arbitrar Ronda.

Tabla 6: Descripción textual del CUN Arbitrar Rondas.

Caso de Uso	**Arbitrar Rondas**
Actores	Director del Torneo (inicia).
Propósito	Arbitrar las rondas de un torneo.
Resumen	El CUN se inicia cuando Director del Torneo ordena comenzar la ronda, acto seguido el Árbitro Principal asegura las condiciones para el comienzo de la ronda, y describe las acciones que realiza el mismo para que el desarrollo de la ronda se realice con la calidad requerida para esto es necesario actualizar el cuadro sinóptico, chequear si algún jugador perdió por no llegar a tiempo, chequear los apuros de tiempo, registrar los resultados de cada partida, ordenar los lugares del torneo utilizando sistemas de desempates de ser necesario, después de esto procede a los cálculos de Rating Performance y Variación de ELO, elaborar un reporte del torneo y así queda terminado el CUN.
Curso normal de los eventos	
Acción del actor	**Respuesta del Negocio**
1. El Árbitro Principal asegura condiciones para el inicio de la ronda.	
2. El Árbitro Principal chequea si algún jugador perdió por no llegar a tiempo.	

3. El Árbitro Principal da tareas específicas a los Árbitros Adjuntos.	4. El Árbitro Adjunto Cumple con las tareas orientadas por el principal.
5. El Árbitro Principal hace cumplir el reglamento.	6. El Árbitro Adjunto actualiza el Cuadro Sinóptico
7. El Árbitro Principal chequea los apuros de tiempo.	
8. El Árbitro Principal registra los resultados de cada partida.	
9. El Árbitro Principal registra resultados del torneo utilizando, de ser necesario, sistemas de desempates.	
10. El Árbitro Principal realiza cálculos de Rating Performance y Variación de ELO.	
11. El Árbitro Principal entrega resultados finales al director del torneo	12. El Director del Torneo recibe resultados finales del mismo.
13. El Árbitro Principal conforma reporte del torneo y las normas conseguidas.	
14. El Árbitro Principal entrega reporte del torneo.	15. La Federación recibe el reporte del torneo.
16. El Árbitro Principal conforma reporte de cumplimiento de una norma de jugadores y árbitros.	
17. El Árbitro Principal informa evaluación a los árbitros adjuntos.	18. Árbitro Adjunto recibe evaluación.
19. El Árbitro Principal da por terminado el torneo y termina el CUN.	

2.7 Diagramas de Actividades.

Con el objetivo de mostrar el flujo de trabajo que realiza cada proceso del negocio se utilizarán los diagramas de actividades que no son más que diagramas de flujos (paralelos o secuenciales) que muestran el flujo de control de una actividad a otra, donde los datos aparecen como objetos que fluyen entre las actividades como se muestra en las siguientes figuras.

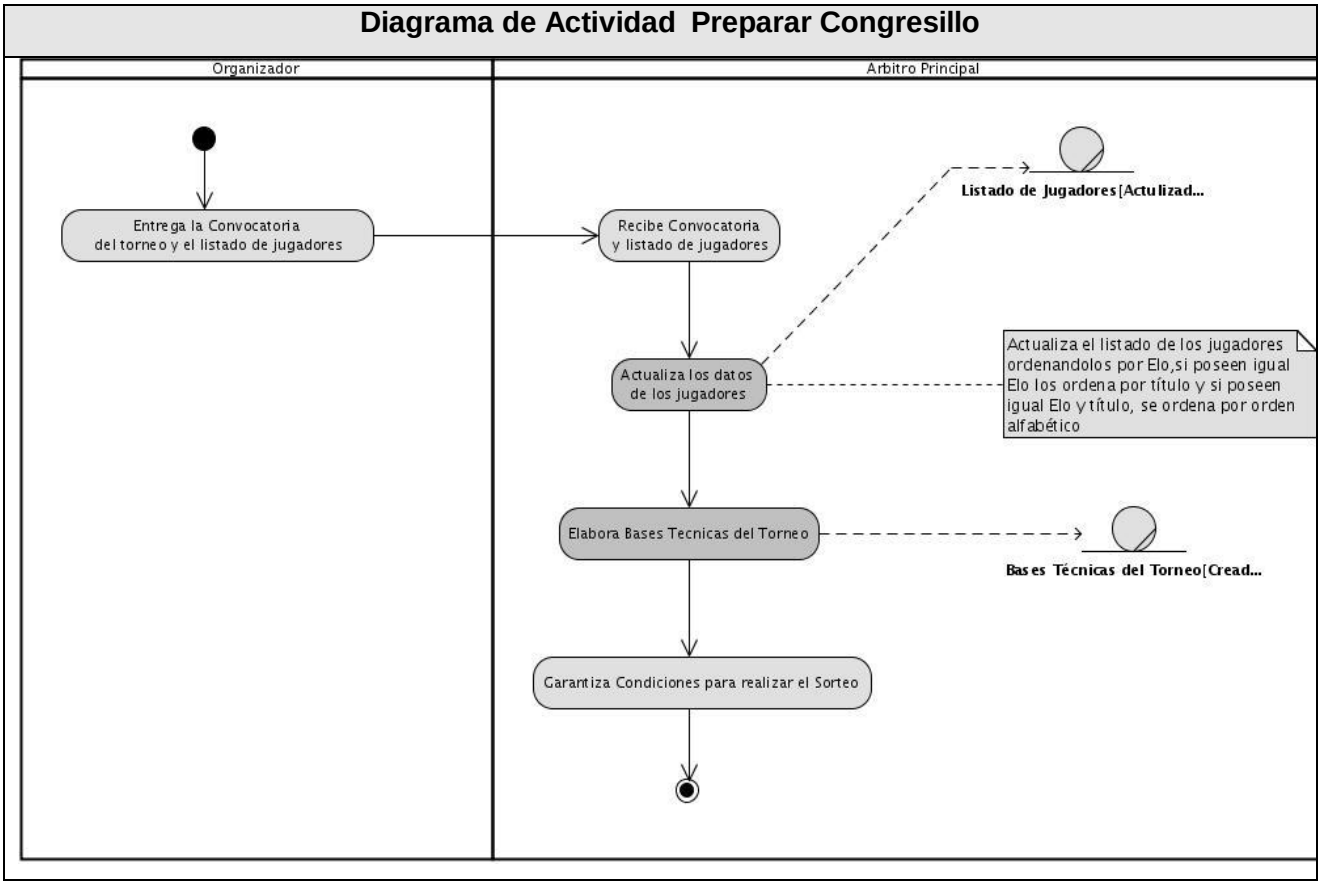

Figura 2: Diagrama de actividad Preparar Congresillo.

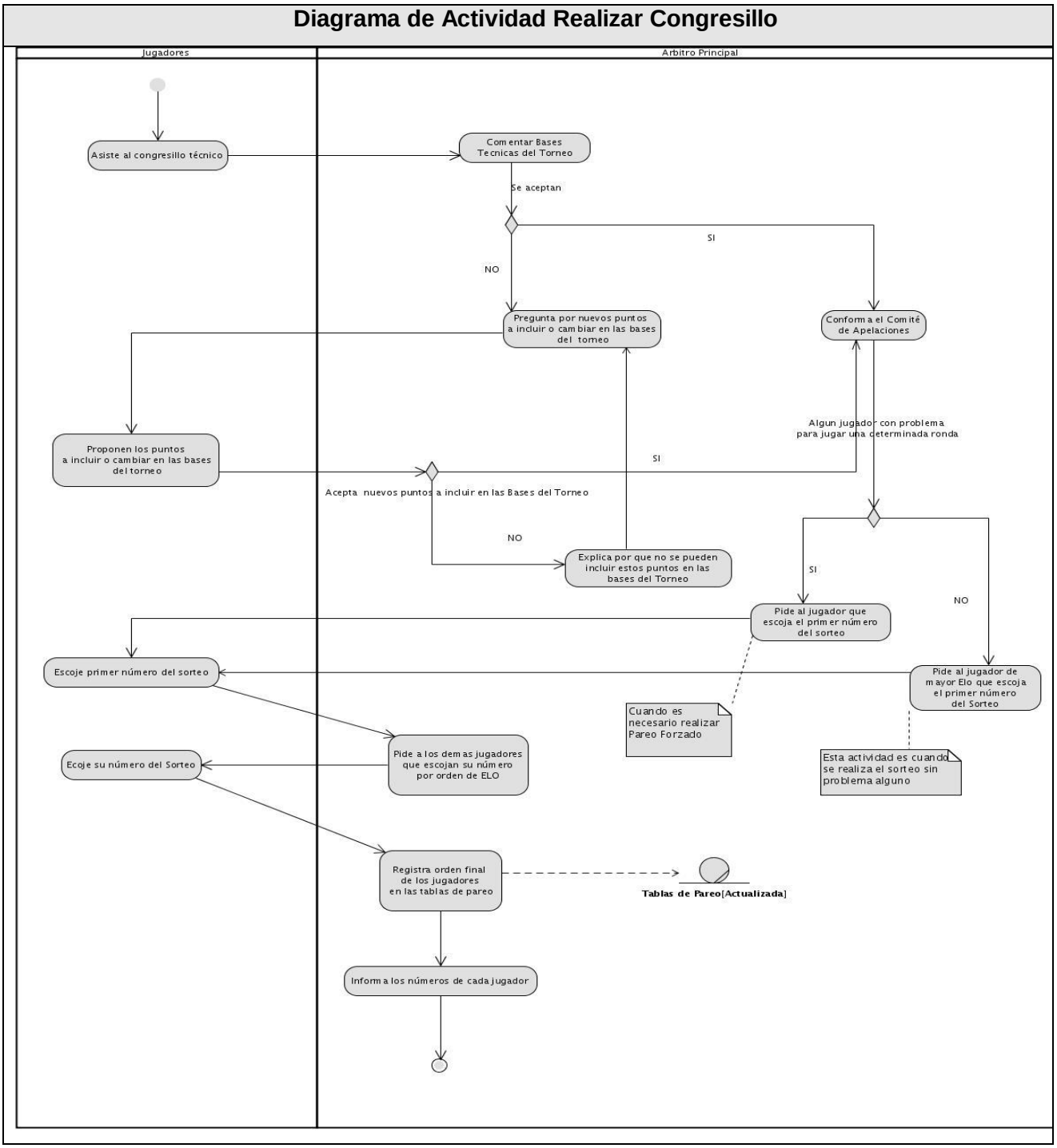

Figura 3: Diagrama de actividad Realizar Congresillo.

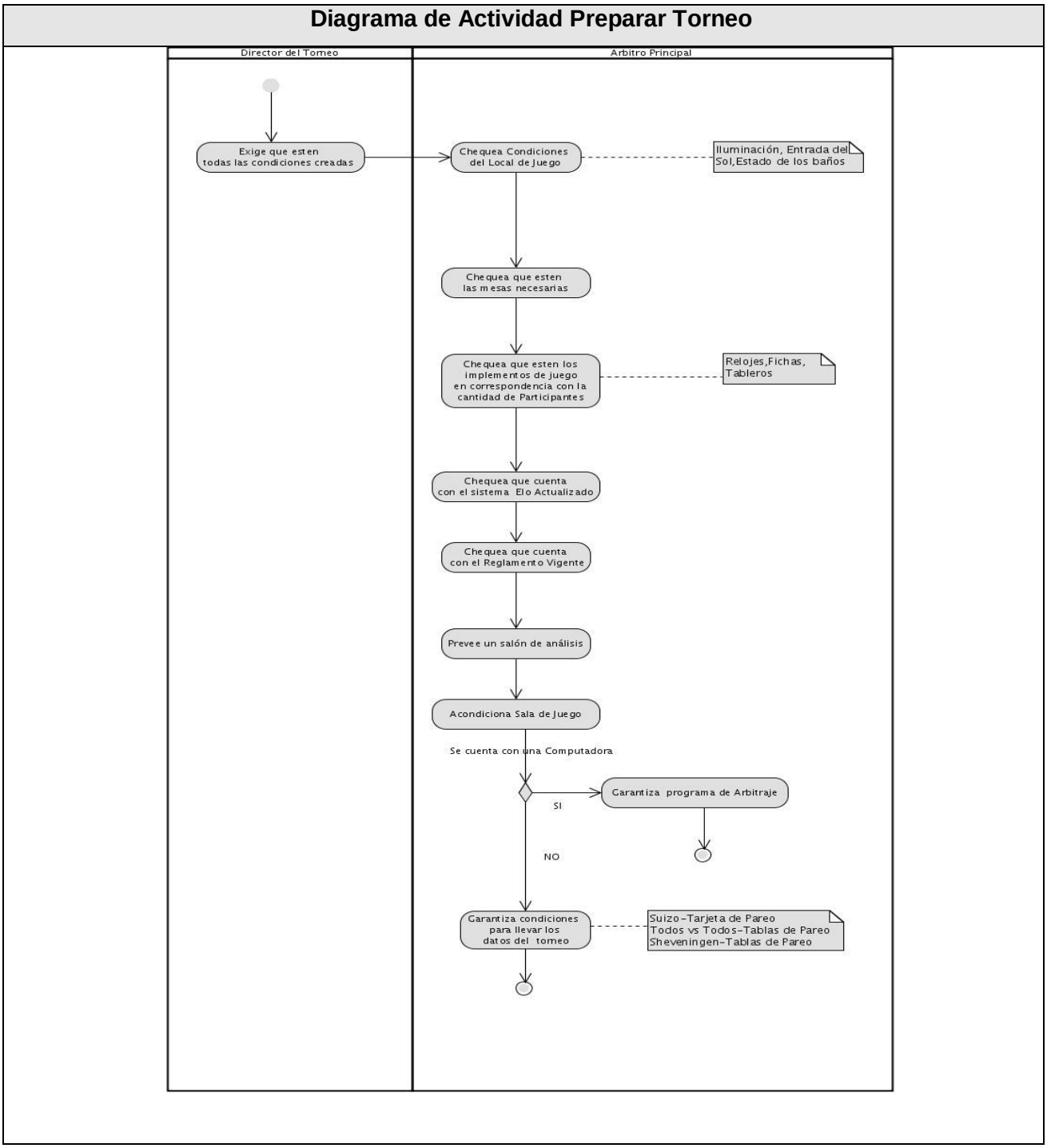

Figura 4: Diagrama de actividad Preparar Torneo.

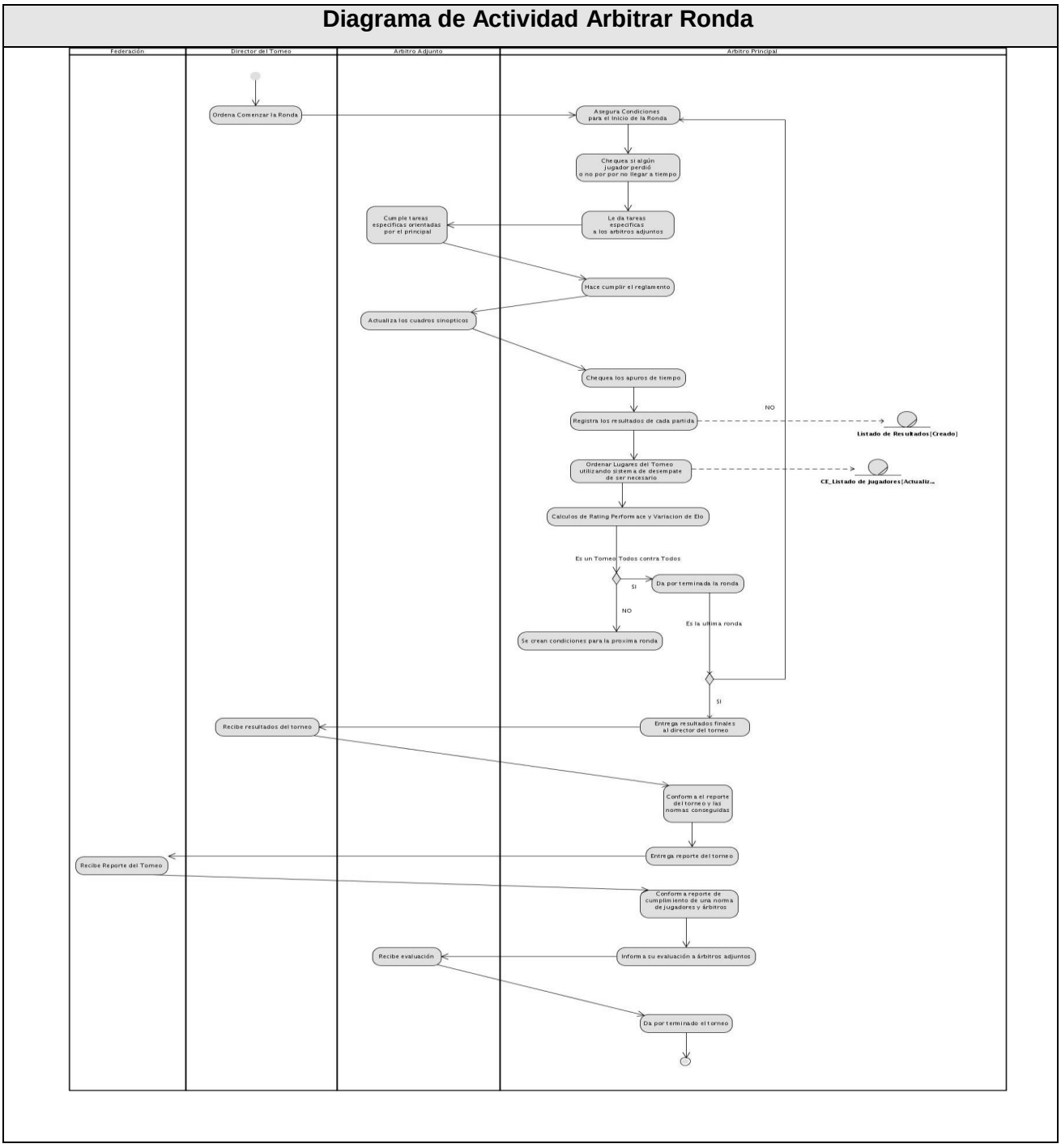

Figura 5: Diagrama de Actividad Arbitrar Ronda.

2.8 Diagramas de Clases del Modelo de Objetos.

Este diagrama muestra la relación existente entre los trabajadores y las entidades del negocio y proporciona un acercamiento a la identificación de los futuros actores y entidades del sistema.

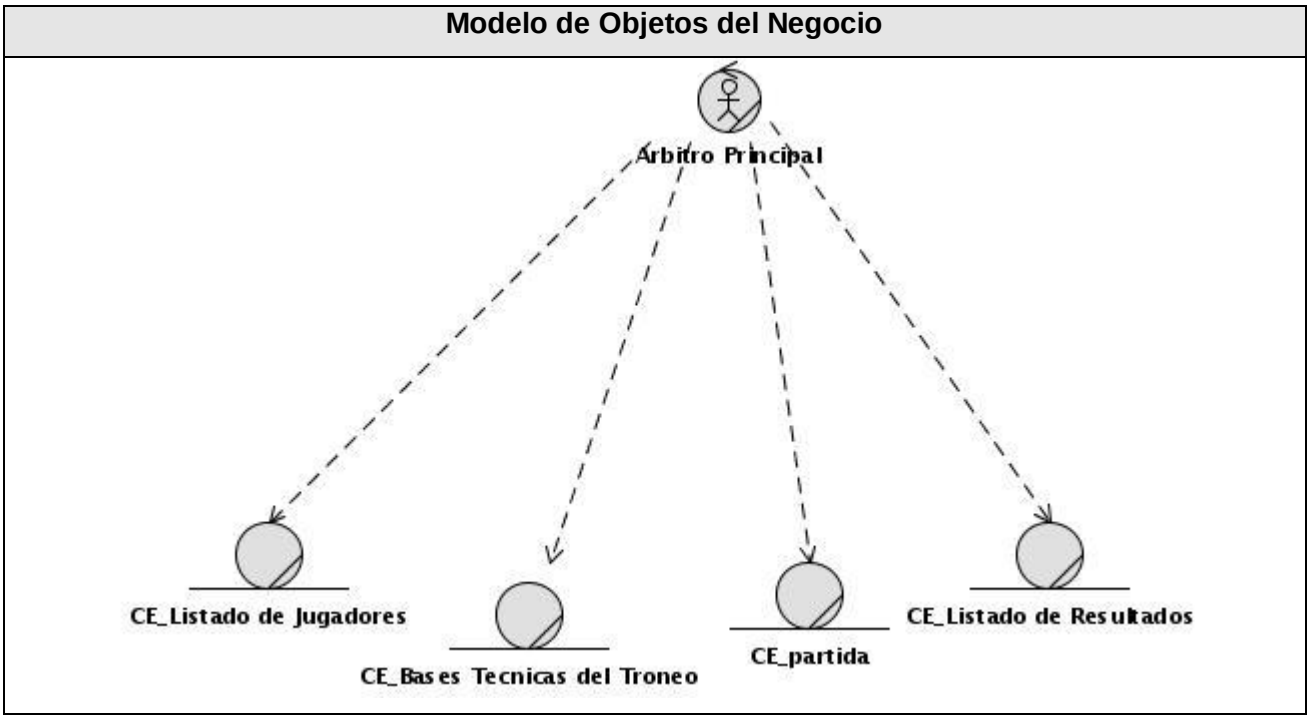

Figura 6: Modelo de Objetos del Negocio.

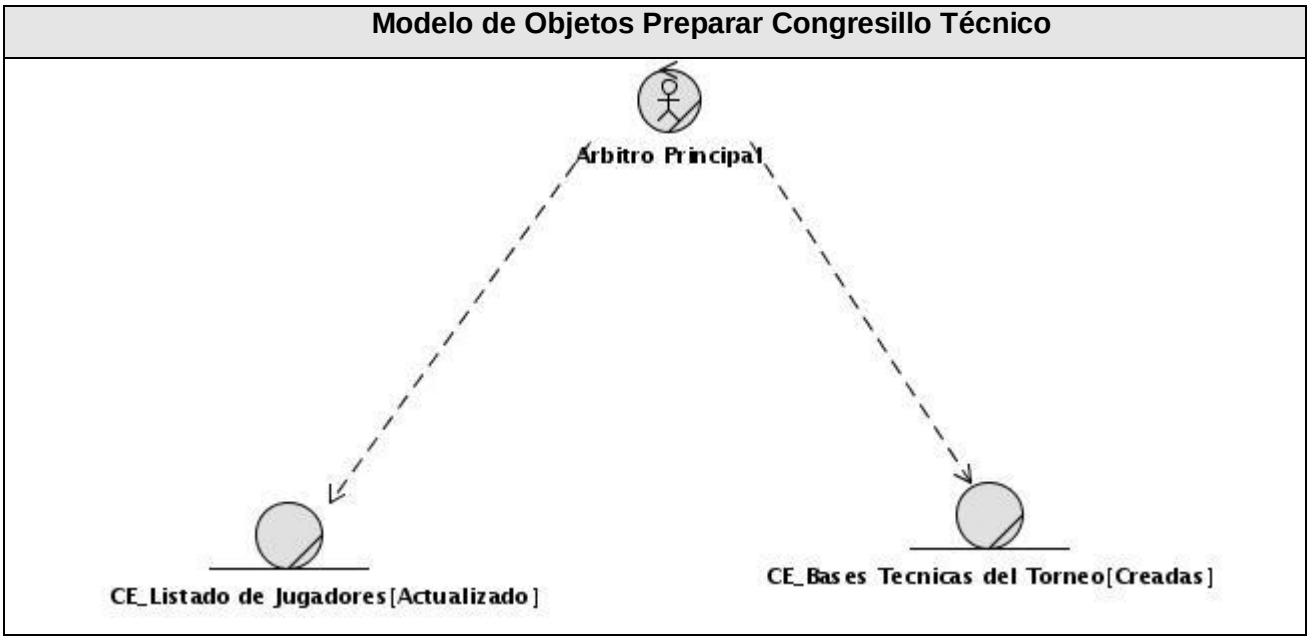

Figura 7: Modelo de Objetos Preparar Congresillo Técnico.

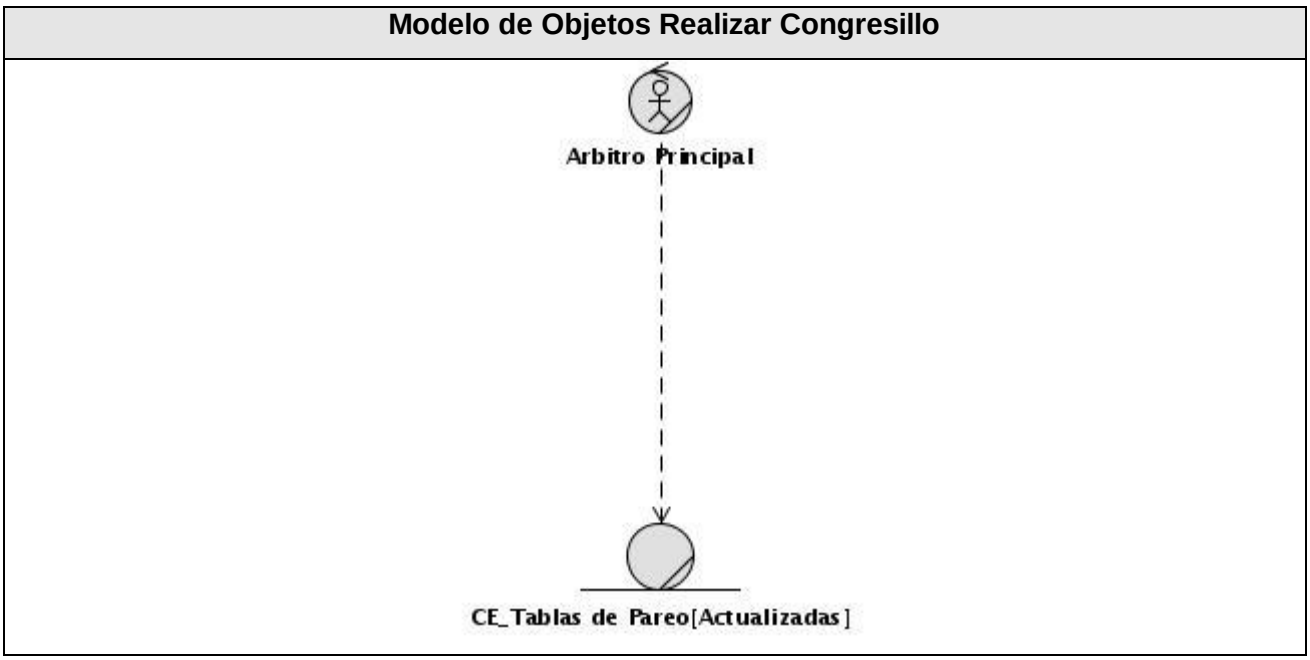

Figura 8: Modelo de Objetos Realizar Congresillo.

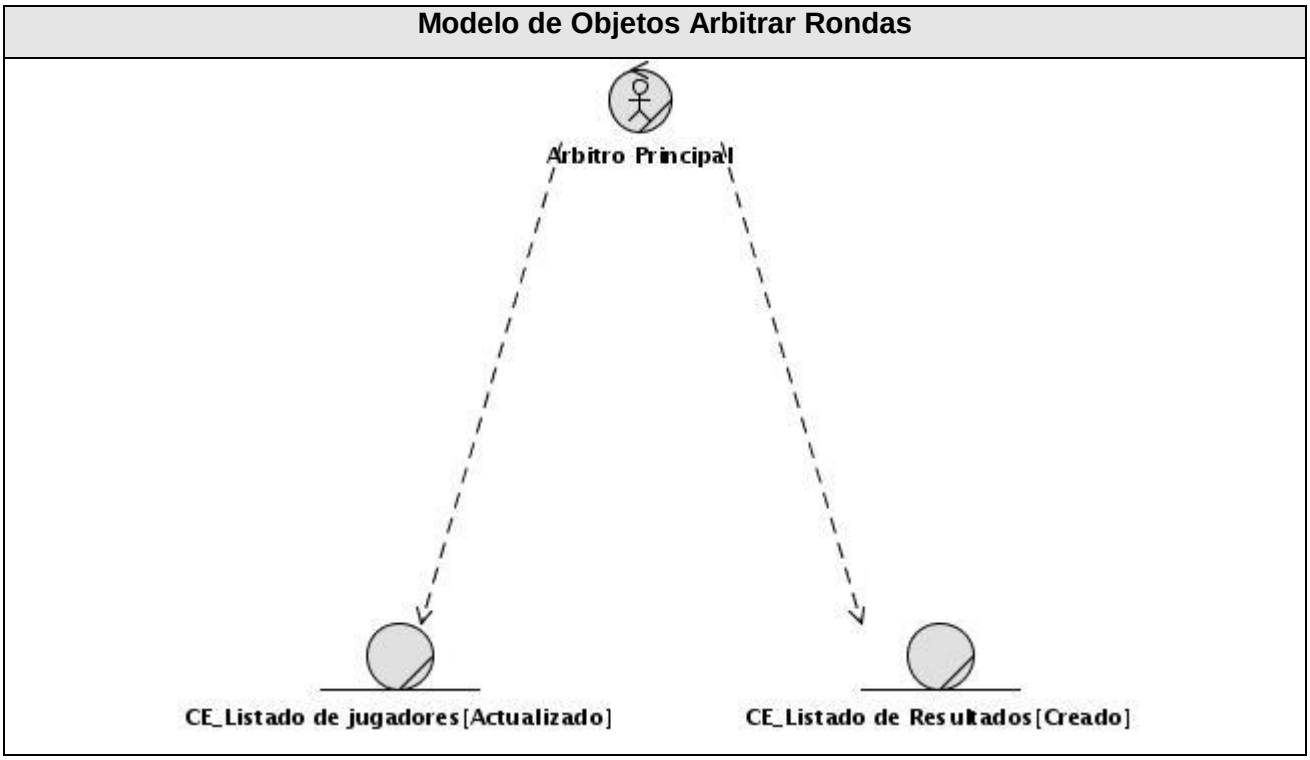

Figura 9: Modelo de Objeto Arbitrar Rondas.

2.9 Requerimientos.

Una de las principales tareas en el ciclo de desarrollo de un sistema es la de determinar los requerimientos del sistema de información (es decir, las condiciones o capacidades que el sistema debe cumplir). El propósito principal del flujo de trabajo de RUP captura de requisitos es guiar el desarrollo hacia el sistema correcto donde se describan con claridad y sin ambigüedades el comportamiento del mismo. Así como lograr una comunicación efectiva entre el cliente (incluyendo a los usuarios) y a los desarrolladores sobre qué debe y qué no debe hacer el sistema.

Los requisitos se pueden clasificar en: funcionales y no funcionales.

2.9.1 Requisitos Funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Su meta principal es identificar y documentar las acciones que en realidad debe ejecutar el sistema para que cumpla con los objetivos planteados al inicio de este trabajo.

Todas estas acciones se convierten en requisitos funcionales y de acuerdo a los objetivos propuestos el sistema debe ser capaz de:

RF 1 Autenticarse.

RF 1.1 Autenticar usuarios.

RF 1.2 Cerrar Sesión.

RF 2 Gestionar Jugador

RF 2.1 Insertar Jugador.

RF 2.2 Actualizar Jugador.

RF 2.3 Mostrar información de jugador.

RF 2.4 Eliminar Jugador

RF 2.5 Buscar Jugador por Nombre.

RF 3 Gestionar Torneo.

RF 3.1 Buscar Torneo.

RF 3.2 Crear Torneo.

RF 3.3 Eliminar Torneo

RF 4 Gestionar Árbitro.

RF 4.1 Insertar Árbitro.

RF 4.2 Eliminar Árbitro.

RF 4.3 Actualizar Árbitro.

RF 4.4 Buscar Árbitro.

RF 5 Gestionar Ciudad.

RF 5.1 Insertar Ciudad.

RF 5.2 Eliminar Ciudad.

RF 5.3 Actualizar Ciudad.

RF 5.4 Buscar Ciudad.

RF 6 Gestionar Federación.

RF 6.1 Insertar Federación.

RF 6.2 Eliminar Federación.

RF 6.3 Actualizar Federación.

RF 6.4 Buscar Federación.

RF 7 Gestionar Ritmo de Juego.

RF 7.1 Insertar Ritmo de Juego.

RF 7.2 Eliminar Ritmo de Juego.

RF 7.3 Actualizar Ritmo de Juego.

RF 7.4 Buscar Ritmo de Juego.

RF 8 Elaborar Bases Técnicas del Torneo.

RF 8.1 Conformar Reporte de Bases Técnicas.

RF 9 Realizar Sorteo.

RF 10 Realizar Pareo.

RF 10.1 Mostrar Partidas.

RF 10.2 Mostrar Partidas por Rondas.

RF 11 Gestionar Partidas.

RF 11.1 Insertar Resultado de Partida.

RF 11.2 Actualizar Resultado de Partida.

RF 12 Mostrar Cuadro Sinóptico.

RF 13 Mostrar Lugares del Torneo.

RF 14 Mostrar Variación Rating ELO.

2.9.2 Requisitos no Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto y de esta forma poder decir si el producto tiene la calidad requerida.

Software:

Para la instalación del servidor:

- Servidor Apache versión 2 o superior.
- Servidor de bases de datos MySQL versión 4 o superior.
- Sistemas Operativos GNU/Linux (recomendado) o Microsoft Windows.

Para la interpretación por clientes (Navegadores Web):

- Internet Explorer v6, Internet Explorer v7 (recomendado) o superior.
- Mozilla Firefox 2.* (recomendado) o superior (recomendado).
- Cualquier navegador libre (excepto Opera).

Hardware:

- Mínimo Requerido: RAM 128 Mb, Disco duro 512 Mb.
- Mínimo Recomendado: RAM 256 Mb, Disco duro 1Gb.
- Conexión de red por MODEM o fibra óptica.

Apariencia o Interfaz Externa

- El diseño de la interfaz debe ser lo más sencillo y claro posible y contar con aspectos que definan claramente cada una de sus funcionalidades.
- La navegabilidad debe ser fácil y sencilla.

Usabilidad:

- El sistema podrá ser utilizado por cualquier usuario que acceda al mismo.
- Proporcionará rápido acceso de búsqueda de la información, en tiempos cortos.
- Contará con zonas de acceso limitado, a la cual solo accederán los usuarios autorizados con roles determinados.
- Fácil gestión de la información

Rendimiento

- El sistema deberá ser capaz de gestionar toda la información y dar respuestas rápidas, casi inmediatas, a las solicitudes realizadas.
- Debe ser eficiente en la gestión de las solicitudes que se realicen, para que el usuario obtenga, sin mucha navegación, el resultado que desea.
- Debe estar disponible las 24 horas del día.

Diseño e implementación:

- El sistema es recomendado que funcione sobre una plataforma de sistema operativo GNU/Linux a modo texto, basado en el modelo cliente/servidor. Los lenguajes necesarios para trabajar son HTML, JavaScript y PHP4 o superior. Como gestor de Base Datos se utilizará MySQL, se desarrollará sobre un sistema GNU/Linux, utilizándose herramientas como el ZendStudio, phpmyadmin, Visual Paradigm.

Portabilidad:

- Es un sistema multiplataforma. Podrá ser montado sobre los sistemas operativos: Unix, Linux, Windows.

Seguridad:

- Verificar la autenticidad y niveles de privilegio del usuario antes de hacer cualquier operación dentro del sistema. Verificar que las funcionalidades sean mostradas en correspondencia con los niveles de privilegios. Protección contra operaciones no autorizadas que en algún momento puedan afectar la integridad de los datos.

Confidencialidad:

- Solo se permitirá conexiones mediante usuarios del directorio activo.

Soporte:

- El sistema debe ser de fácil instalación y configuración, con vista a poder darle un mantenimiento relativamente fácil y sencillo en caso de fallos.

Legales:

- La plataforma está basada en la licencia GNU/GPL.
- El sistema es reconocido y autorizado por entidades superiores tales como la directiva de la UCI.

Ayuda y documentación en línea

- Ofrecerá una documentación de ayuda para uso del sistema por los usuarios, la cual estará disponible desde cualquier parte del mismo para aclarar cualquier duda que el usuario presente a la hora de manejar y hacer uso de la aplicación web.

2.10 Modelación del Sistema.

Los actores del sistema definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que interactúan con el mismo intercambiando información con este.

A continuación se detallan los actores del sistema:

2.11 Descripción de los Actores del sistema.

Tabla 7: Descripción de los Actores del Sistema.

Actores del sistema	Justificación
Árbitro Principal	Es el encargado de realizar todo el proceso de arbitraje en un torneo o evento de ajedrez.
Administrador	Es el encargado de gestionar todo lo referente a los árbitros que interactuarán con el sistema.
Usuario	Es un actor genérico del cual son heredan Árbitro Principal y Administrador.

2.12 Diagrama de casos de uso del sistema.

Los Casos de Uso del sistema (CUS) son un conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor. Con otras palabras, son “fragmentos” de funcionalidad que el sistema ofrece a los actores que interactúan con el mismo, reportándoles algunos que otro beneficio.

A continuación se muestra la figura 2.10 correspondiente al diagrama de casos de uso del sistema:

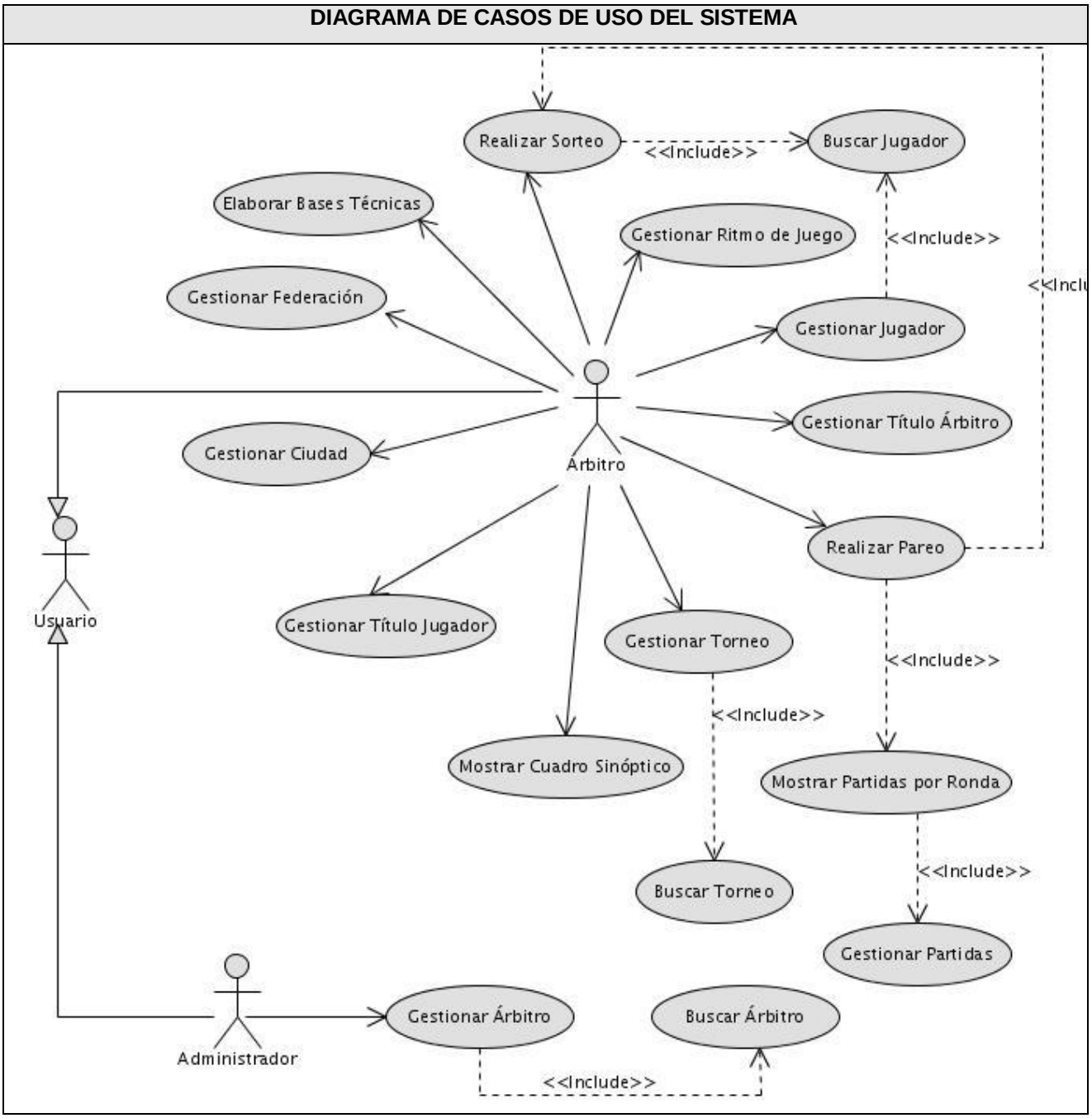

Figura 10: Diagrama de Casos de Uso del Sistema.

2.13 Descripción de los Casos de Uso del Sistema.

La descripción de los casos de uso del sistema, detallan las acciones que tienen lugar durante la interacción actor-sistema, es decir, describe el flujo de actividades que realiza el actor al hacer uso del sistema y las correspondientes respuestas del mismo. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre que es lo que el sistema debe hacer (requisitos).

2.13.1 Descripción del CUS "Realizar Sorteo".

Tabla 8: Descripción del CUS Realizar Sorteo.

Caso de Uso	**Realizar Sorteo**
Actores	Árbitro principal
Propósito	Realizar sorteo conociendo inicialmente la lista de jugadores que participaran en el torneo.
Resumen	El caso de uso comienza cuando el Árbitro Principal añade los jugadores a un torneo, acto seguido escoge la opción sorteo del menú Torneo y para cada jugador realiza la opción de sortear.
Referencias	RF1, RF2
Precondiciones	El árbitro principal debe estar logueado en el sistema y debe poseer con antelación la lista de jugadores participantes.
Poscondiciones	Después de realizado el sorteo ya se cuenta con que cada jugador tiene un número aleatorio asignado por lo que ya nos encontramos en condiciones de realizar el pareo.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Árbitro principal: Busca los jugadores que participarán en el torneo y los adiciona a este.	2- El sistema muestra un mensaje informando que los jugadores se han adicionado correctamente al torneo.
3- Árbitro principal: Selecciona la opción sorteo del menú Arbitraje.	4- El sistema muestra todos los jugadores participantes en el torneo ordenados por ELO, brindando la posibilidad de que el árbitro pueda sortear los jugadores.

5- Árbitro principal: Sortea los jugadores por orden de ELO.	6- El sistema asigna y muestra el número de sorteo para cada jugador.
	7- El caso de uso termina.
Curso alterno de los eventos	
	2- Si la operación no tuvo éxito emite un mensaje informado que el jugador no se pudo adicionar con éxito al torneo. 3- El sistema brinda la posibilidad de que el árbitro vuelva a introducir los datos del jugador y se proceda al flujo 1 del curso normal de eventos.
Prioridad	Crítico

2.13.2 Caso de uso: Gestionar Torneo.

Tabla 9: Descripción del CUS Gestionar Torneo.

Caso de Uso	**Gestionar Torneo**
Actores	Árbitro principal
Propósito	Permite a los árbitros gestionar toda la información referente a un torneo (Crear un nuevo torneo, modificar un torneo existente, eliminar un torneo, ver todos los torneos en los que esta trabajando).
Resumen	El CUS se inicia cuando el árbitro selecciona la opción Torneo del menú Arbitraje luego selecciona la el tipo de gestión a realizar sobre el torneo introduce los datos necesarios, el sistema realiza la acción seleccionada por el árbitro y termina el CUS.
Referencias	RF 4
Precondiciones	- El Árbitro Principal debe estar logueado en el sistema.
Poscondiciones	-Torneo adicionado a la Base de Datos. -Torneo eliminado de la Base de Datos.
Curso normal de los eventos	

Acción del actor	**Respuesta del sistema**
1-El Árbitro selecciona la opción Torneos del menú Arbitraje.	2-El sistema muestra las opciones Iniciar Torneo. Ver Torneo. Eliminar Torneo.
Escenario 1 :Iniciar Torneo	
1- Árbitro Principal: Selecciona la opción de iniciar un nuevo torneo.	2-El sistema muestra el formulario de creación de torneo con la opción de entrada de los datos necesarios para esta acción.
3- Árbitro Principal: Introduce los datos necesarios para la creación del nuevo torneo y pulsa el botón crear.	4-El sistema verifica que los datos son correctos y en caso afirmativo inserta los datos en la base de datos concluyendo con esto el CUS.
Curso alterno de los eventos	
	Si los datos introducidos por el árbitro son incorrectos el sistema muestra un mensaje indicando en que elemento del formulario fue donde estuvo el error. Se procede al flujo normal de eventos desde la acción 3.
Escenario 2 :Eliminar Torneo	
1- Árbitro Principal: Selecciona la opción eliminar torneo del menú Torneo.	2-El sistema brinda la posibilidad de que el árbitro busque el torneo que desea eliminar devolviendo como resultado todos los torneos que cumplen con el parámetro de búsqueda.
3- Árbitro Principal: Selecciona el torneo que desea eliminar dando sobre el vínculo eliminar	4-El sistema verifica que el árbitro tiene los permisos de administración sobre el torneo a eliminar y caso correcto elimina a este del sistema.
	5-El sistema muestra un mensaje informando

	que el torneo ha sido eliminado del sistema concluyendo con esto el CUS.
Curso alterno de los eventos	
	4-Si el árbitro no posee los permisos necesarios sobre el torneo el sistema muestra un mensaje de error cuando el árbitro trata de eliminar dicho torneo. Se procede al flujo normal de eventos desde la acción 2.
Escenario 2 :Ver Torneos	
1- Árbitro Principal: Selecciona la opción ver torneos del menú Torneos	2-El sistema muestra todos los torneos en los que se encuentra dicho árbitro dándole a este la posibilidad que trabaje con uno de estos torneos concluyendo con eso el CUS.
Curso alterno de los eventos	
	2-Si el árbitro no se encuentra administrando ningún torneo en el momento en que el árbitro pulsa sobre la opción ver torneos el sistema muestra un mensaje informado al árbitro.
Prioridad	Crítico

2.13.3 Caso de uso: Gestionar Árbitro.

Tabla 10: Descripción del CUS Gestionar Árbitro.

Caso de Uso	**Gestionar Árbitro**
Actores	Administrador
Propósito	Permite a los administradores gestionar toda la información referente a un los árbitros Insertar un nuevo árbitro, modificar los datos de un árbitro, eliminar un árbitro, Buscar Árbitro).
Resumen	El CUS se inicia cuando el administrador se auténtica en el sistema y el mismo lo envía al módulo de administración donde se muestra un menú con las opciones de gestionar la información de los árbitros.

Referencias	RF 10
Precondiciones	- El Administrador debe estar logueado en el sistema.
Poscondiciones	-Árbitro adicionado a la Base de Datos. - Árbitro eliminado de la Base de Datos. -Datos del Árbitro actualizados.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1-El Administrador se loguea en el sistema entrando a la sesión de administración del mismo.	2-El sistema muestra las opciones Nuevo Árbitro Editar Árbitro. Eliminar Árbitro.
Escenario 1 : Nuevo Árbitro	
1- Administrador: Selecciona la opción de Nuevo Árbitro	2-El sistema muestra el formulario con todos las opciones para que el administrador entre todos los datos que se necesitan para insertar un árbitro al sistema.
3- Administrador: Introduce los datos necesarios para adicionar un nuevo árbitro y pulsa el botón Entrar.	4-El sistema verifica que los datos son correctos y en caso afirmativo inserta los datos en la base de datos concluyendo con esto el CUS.
Curso alterno de los eventos	
	Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje indicando en que elemento del formulario fue donde estuvo el error. Se procede al flujo normal de eventos desde la acción 3.
Escenario 2: Editar Árbitro.	
1- Administrador: Selecciona la opción editar árbitro del menú de Administración.	2-El sistema brinda la posibilidad de que el administrador busque el árbitro que desea editar devolviendo como resultado todos los

	árbitros que cumplen con el parámetro de búsqueda.
3- Administrador: Selecciona el árbitro al cual desea modificar.	4-El sistema muestra un formulario con todos los datos del árbitro
5- Administrador: Modifica los datos deseados y pulsa el botón Entrar.	5-El sistema verifica que los datos entrados son correctos y en caso afirmativo registra los mismos en la Base de Datos concluyendo con esto el CUS.
Curso alterno de los eventos	
	4-Si los datos a modificar no fueron introducidos correctamente en el sistema el mismo muestra un mensaje indicando en que parte del formulario fue donde estuvo el error, se procede al flujo normal de eventos desde la acción 5.
Escenario 3: Eliminar Árbitro.	
1- Administrador: Selecciona la opción eliminar árbitro del menú de árbitro.	2-El sistema brinda la posibilidad de que el administrador busque el árbitro que desea eliminar.
3- Administrador: Entra el parámetro de búsqueda y pulsa el botón buscar.	4-El sistema muestra todos los árbitros que responden al parámetro de búsqueda y unidos a estos muestra también la opción de eliminar.
5- Administrador: Pulsa sobre el botón eliminar del árbitro seleccionado.	6-El sistema elimina al árbitro de la base de Datos concluyendo con esto el CUS.
Prioridad	Crítico

2.13.4 Caso de uso: Elaborar Bases Técnicas del Torneo.

Tabla 11: Descripción del CUS Elaborar Bases Técnicas del Torneo.

Caso de Uso	**Elaborar Bases Técnicas**
Actores	Árbitro principal
Propósito	Brindar la posibilidad al árbitro de elaborar un reporte con las Bases Técnicas de un torneo en un formato pdf.
Resumen	El caso de uso comienza cuando el Árbitro Principal oprime el botón Bases Técnicas y a partir de ahí se genera un documento con las mismas.
Referencias	RF 5
Precondiciones	El torneo no debe haberse creado.
Poscondiciones	Se conforma el documento que contendrá las bases técnicas del torneo.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Árbitro Principal: pulsa el botón elaborar bases técnicas del torneo.	2 – El sistema muestra el formulario de entrada de las bases técnicas.
3- Árbitro Principal: Introduce los datos necesarios para que se guarden las bases técnicas del mismo.	4- El sistema guarda las bases técnicas en la base de datos y brida la posibilidad al árbitro que guarde la misma en formato pdf.
	5- El sistema genera un documento pdf con las bases técnicas del torneo.
	6- El CUS termina.
Prioridad	Crítico

2.13.5 Caso de uso: Mostrar Cuadro Sinóptico.

Tabla 12: Descripción del CUS Mostrar Cuadro Sinóptico.

Caso de Uso	**Mostrar Cuadro Sinóptico**
Actores	Árbitro principal
Propósito	Brindar la posibilidad al árbitro de que pueda ver como marchan los resultados particulares de cada una de las partidas en un torneo dado.
Resumen	El CUS inicia cuando el árbitro selecciona la opción Cuadro Sinóptico del menú Torneo mostrándose este con toda la información hasta el momento.
Referencias	RF 6
Precondiciones	Debe haber terminado alguna de las partidas de la ronda en cuestión. El torneo debe estar creado.
Poscondiciones	Se mantiene el resultado de las partidas actualizadas.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Árbitro Principal: Pulsa la opción cuadro sinóptico del menú de arbitraje.	2- El sistema busca los resultados de cada una de las partidas del torneo en cuestión.
	3- El sistema muestra todos los resultados particulares de cada uno de los jugadores con el formato oficial exigido por el cliente.
	4- Finaliza CUS.
Curso alterno de los eventos	
	3- El sistema si todavía no se ha realizado el

	sorteo del torneo en cuestión, muestra un mensaje, de que para ver el Cuadro Sinóptico primero se debe realizar el Sorteo debido a que este es quien define la posición de estos en el cuadro Sinóptico.
	4- Finaliza el CUS.
Prioridad	Crítico

2.13.6 Caso de uso: Mostrar Pareo.

Tabla 13: Descripción del CUS Realizar Pareo.

Caso de Uso	**Mostrar Pareo**
Actores	Árbitro principal
Propósito	Mostrar el pareo de los torneos que se juegan con el sistema todos contra todos.
Resumen	El CUS inicia cuando el árbitro selecciona la opción Pareo del menú Torneo, acto seguido se muestra el pareo de todas las rondas del torneo.
Referencias	RF 8
Precondiciones	Ya debe existir un torneo creado y todos los jugadores a participar en el mismo deben estar ya en el sistema.
Poscondiciones	Se muestra la tabla de pareo completas del torneo en cuestión.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Árbitro Principal: Pulsa en el Botón pareo del menú de Arbitraje.	2- El sistema ejecuta el algoritmo encargado de realizar el pareo verificando que primero haya concluido el sorteo.

	3- El sistema guiado por el número de sorteo de cada jugador para el torneo en cuestión busca el nombre de cada jugador y construye un arreglo con todas las partidas organizadas por rondas.
	4- El sistema busca si ya existe resultados para alguna de estas partidas y en caso afirmativo los muestra junto con el las partidas organizadas por rondas.
	5- Finaliza el CUS
Curso alterno de los eventos	
	2- Si existe algún participante en torneo que no cuenta aún con un número de sorteo el sistema muestra un mensaje informando que se debe realizar primero el sorteo para luego realizar el pareo.
	3- Finaliza el CUS.
Prioridad	Crítico

Conclusiones.

Durante este capítulo fueron expuestas las características que contendrá el sistema, apoyándose para ello en el análisis de los actuales procesos de negocio, siendo identificado, quiénes son los actores y trabajadores que intervienen en el mismo y con cuáles actividades y entidades interactúan.

Se identificaron además, los requisitos funcionales y no funcionales que debe cumplir el sistema en cuestión, y con ello fueron expuestos los casos de uso a tratar durante el desarrollo del mismo, con la correspondiente descripción textual de cada uno, lo cual provee de una visión general de qué es lo que el sistema debe hacer, por lo que se está en condiciones de pasar a ver cómo es que el mismo va a realizar las operaciones antes descritas y con ello, darle solución a los problemas planteados.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.

3.1 Introducción.

En este capítulo se procede a la confección de la propuesta de solución, basándose en el análisis hecho anteriormente. Para el desarrollo de este capítulo se tuvieron en cuenta los requisitos funcionales, como no funcionales obtenidos durante el flujo de la captura de requisitos. El mismo constará con varios artefactos que tienen como objetivos mostrar el diseño y la implementación del sistema propuestos como solución.

3.2 Diagrama de Clases del Análisis

En el análisis se presentan los siguientes estereotipos de clases:

Clase de frontera o interfaz: Modela la interfaz del sistema, y manejan la comunicación entre el entorno y el interior del mismo. Durante el diseño, estas clases son refinadas para tomar en consideración los mecanismos de interfaz seleccionados o implementados, además de facilitar la comunicación con otros sistemas, etc.

Clases de entidad o sistema: Representan la información manejada en el caso de uso, además de que modelan información y comportamiento asociado que generalmente es de larga duración. Reflejan entidades del mundo real, que resultan necesarias para realizar tareas internas del sistema.

Clases de control o software: Coordinan los eventos necesarios para la realización o especificación del caso de uso, con otras palabras, son las que ejecutan el caso de uso. Usualmente son dependientes de la aplicación, además de tener un control sobre todas las acciones a realizar.

3.2.1 Diagrama de Clases del Análisis del CUS Realizar Sorteo.

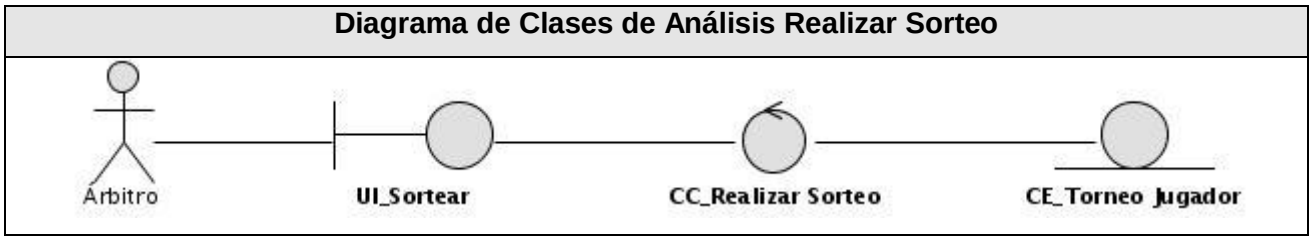

Figura 11: Diagrama de clases de análisis del CUS Realizar Sorteo.

3.2.2 Diagrama de Clases de Análisis del CUS Gestionar Torneo.

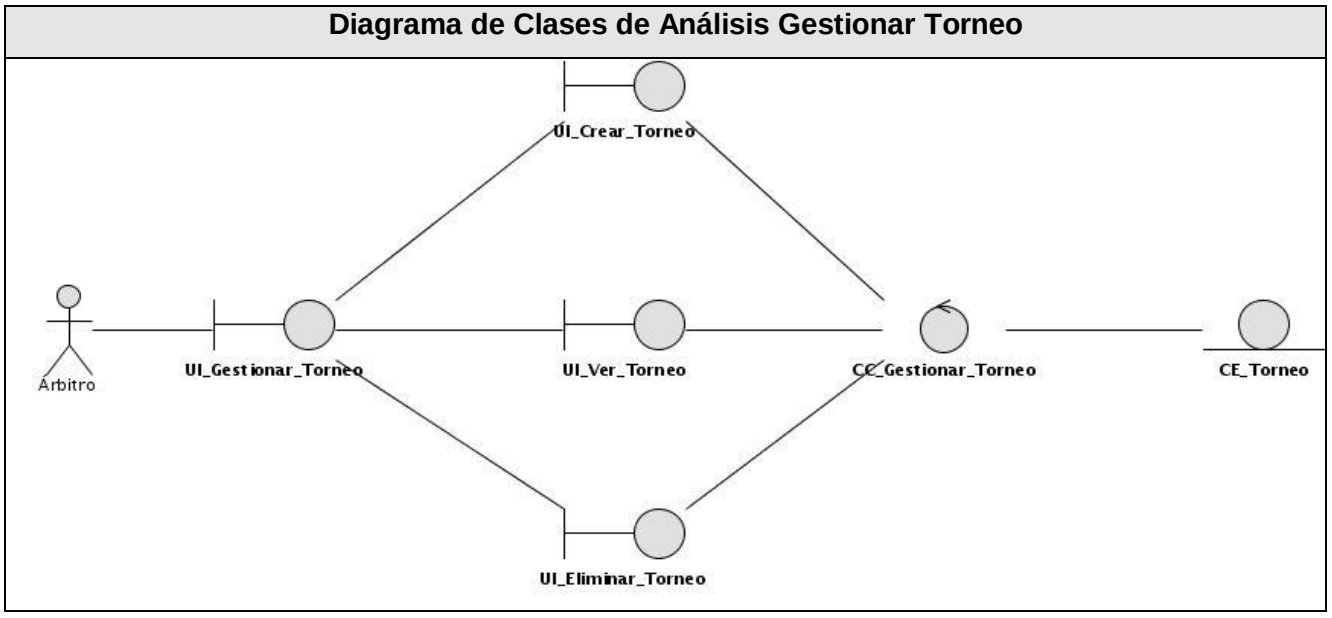

Figura 12: Diagrama de Clases de Análisis del CUS Gestionar Torneo.

3.2.3 Diagrama de Clases del Análisis del CUS Gestionar Árbitro.

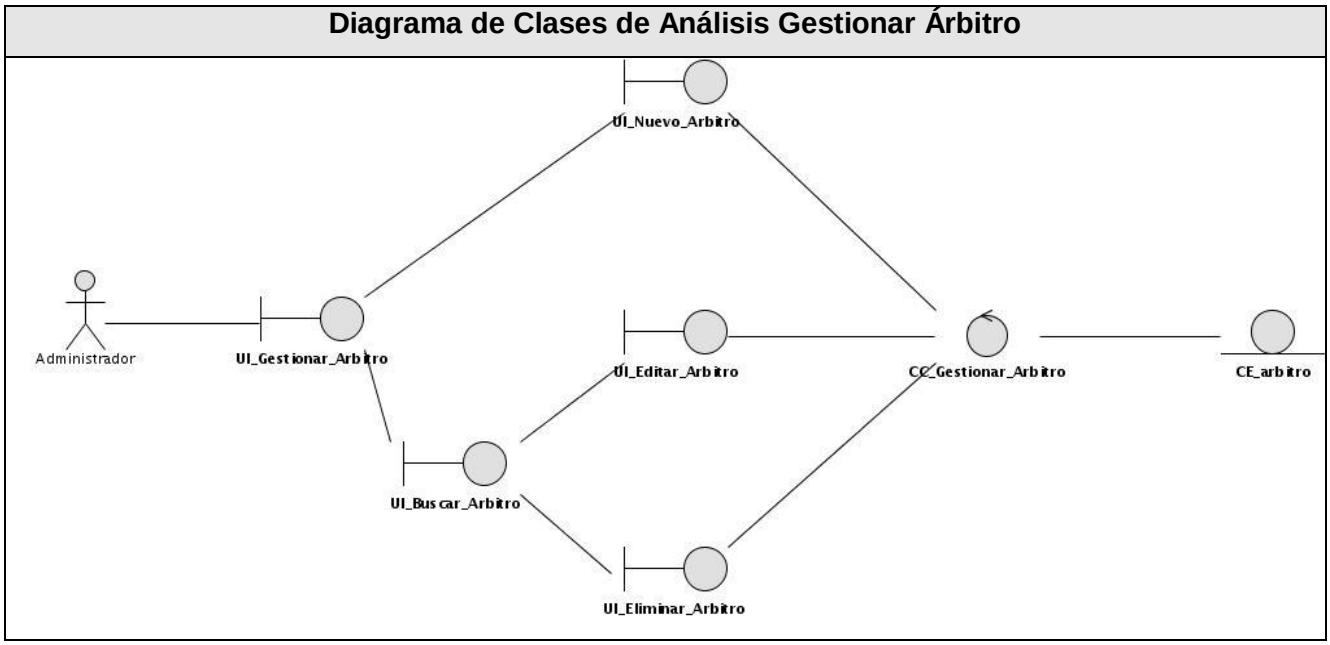

Figura 13: Diagrama de Clases del Análisis del CUS Gestionar Árbitro.

3.2.4 Diagrama de Clases del Análisis del CUS Elaborar Bases Técnicas del Torneo.

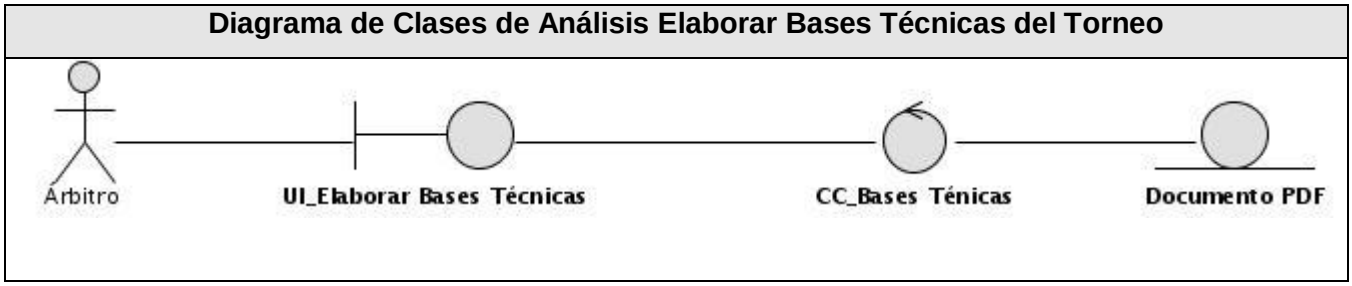

Figura 14: Diagrama de Clases del Análisis del caso de uso Elaborar Bases Técnicas del Torneo.

3.2.5 Diagrama de Clases del Análisis del CUS Mostrar Cuadro Sinóptico.

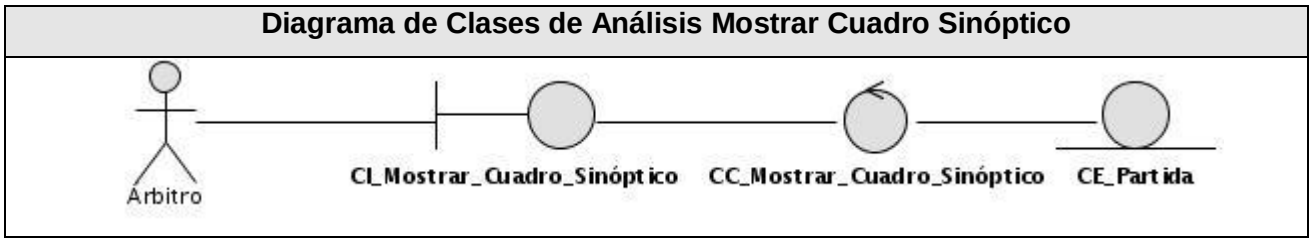

Figura 15: Diagrama de Clases de Análisis del CUS Mostrar Cuadro Sinóptico.

3.2.6 Diagrama de Clases del Análisis del CUS Realizar Pareo.

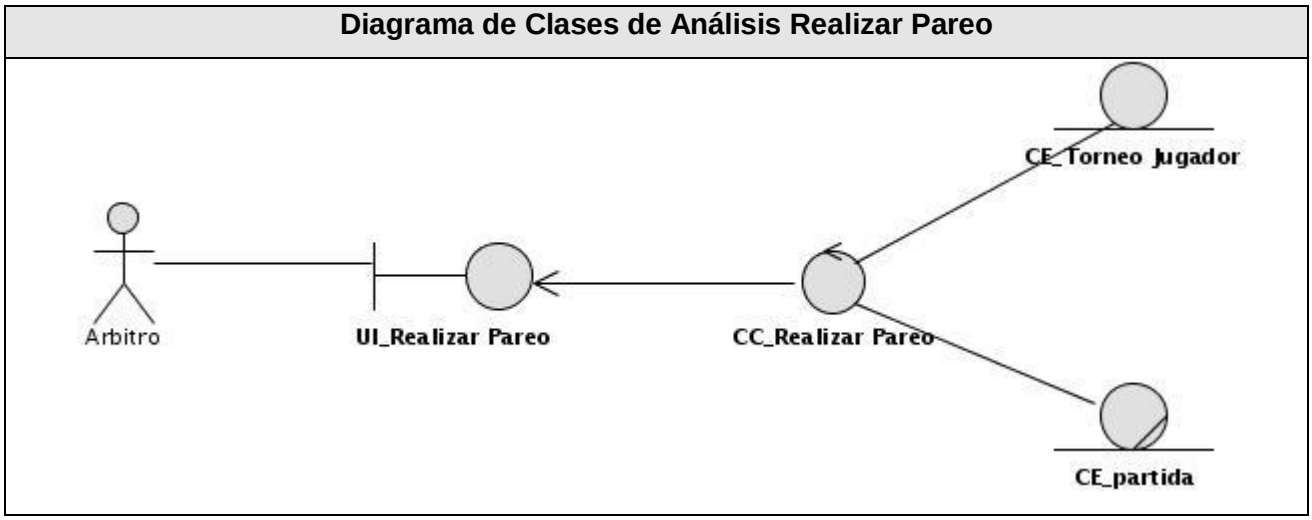

Figura 16: Diagrama de Clases del Análisis del CUS Realizar Pareo.

3.3 Diagrama de Clases del Diseño.

Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. En el caso de las aplicaciones Web, el diagrama de clases representa las colaboraciones que ocurren entre las páginas, donde cada página lógica puede ser representada como una clase.

3.3.1 Diagrama de Clases Web para el CUS Realizar Sorteo.

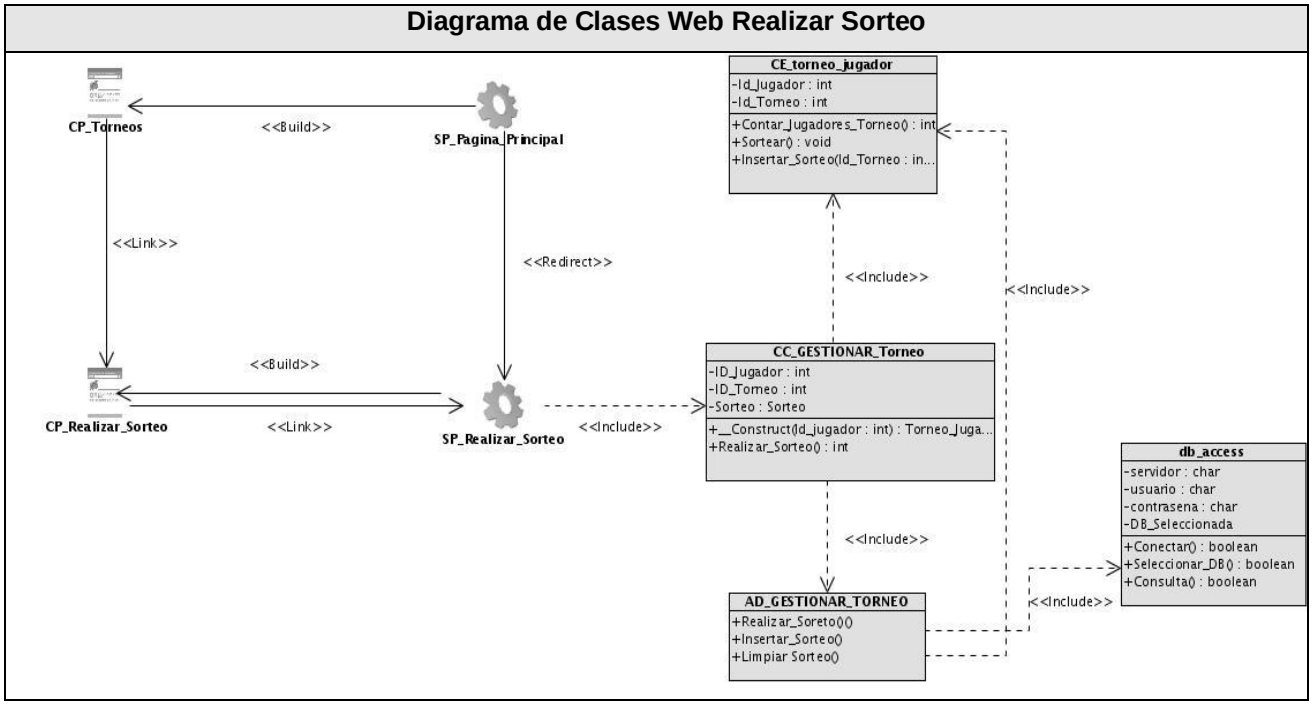

Figura 17: Diagrama de Clases Web del CUS Realizar Sorteo.

3.3.2 Diagrama de Clases Web para el CUS Gestionar Torneo.

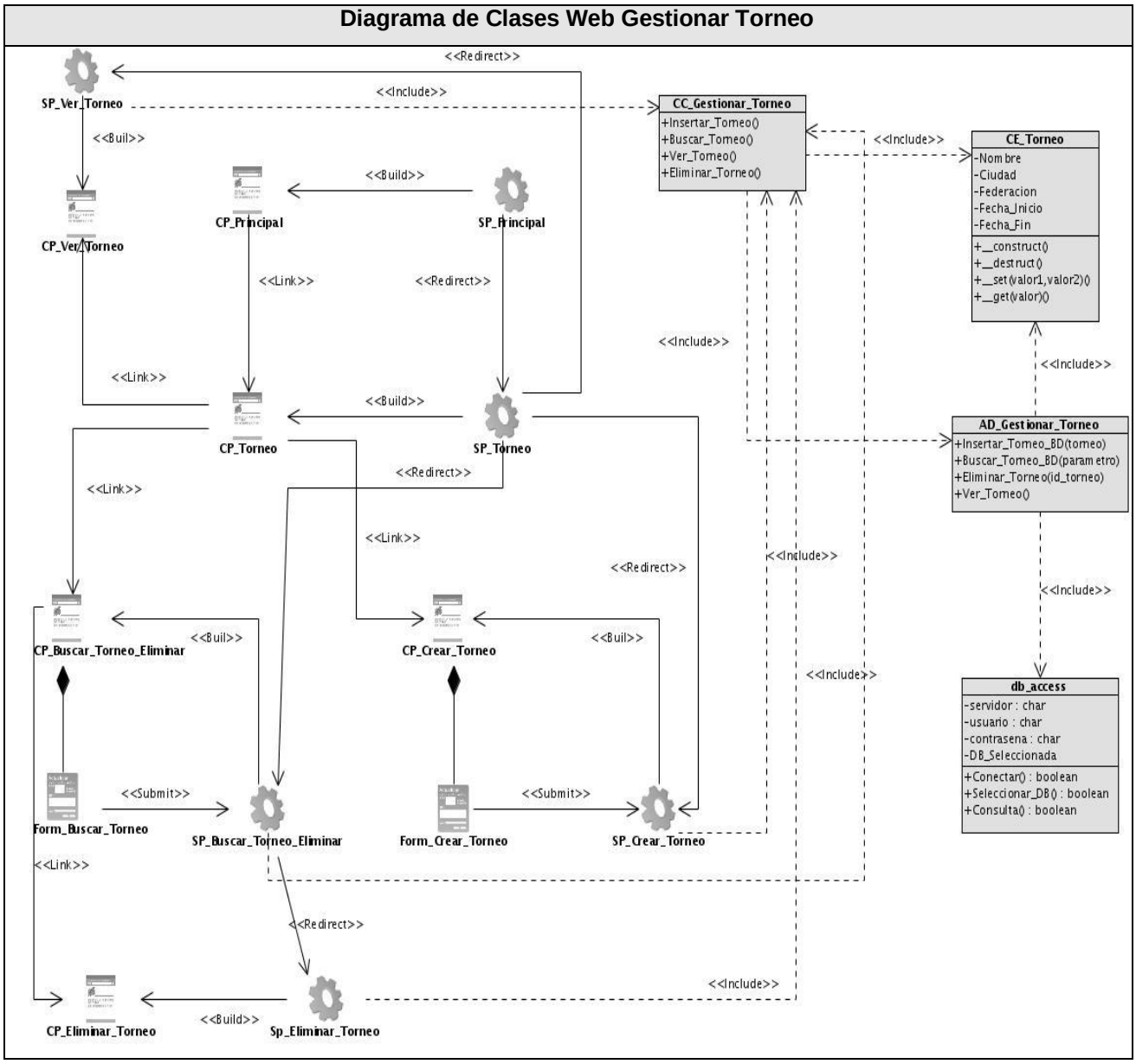

Figura 18: Diagrama de Clases Web del CUS Gestionar Torneo.

3.3.3 Diagrama de Clases Web para el CUS Gestionar Árbitro.

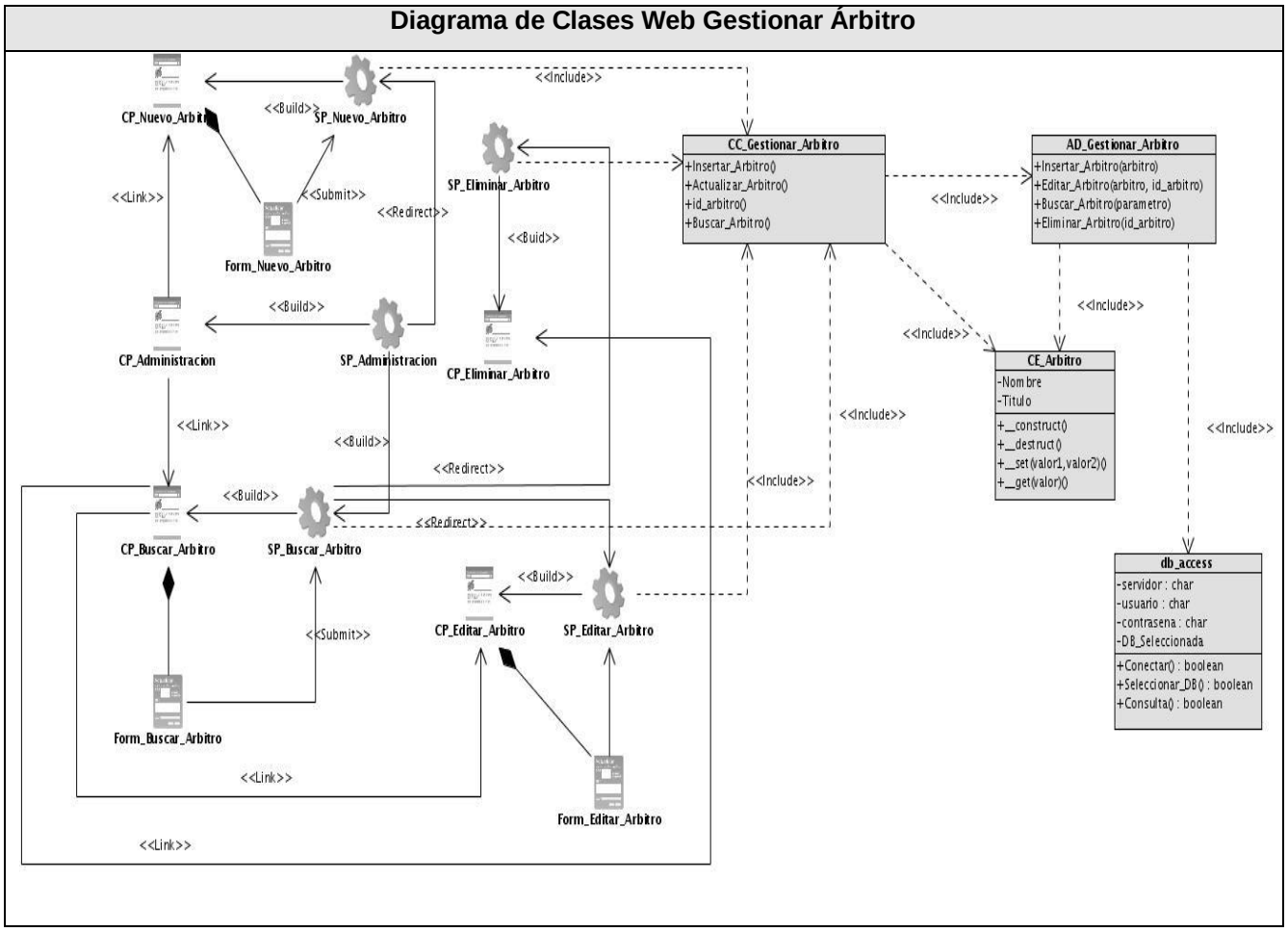

Figura 19: Diagrama de Clases Web del CUS Gestionar Árbitro.

3.3.4 Diagrama de Clases Web para el CUS Elaborar Bases Técnicas del Torneo.

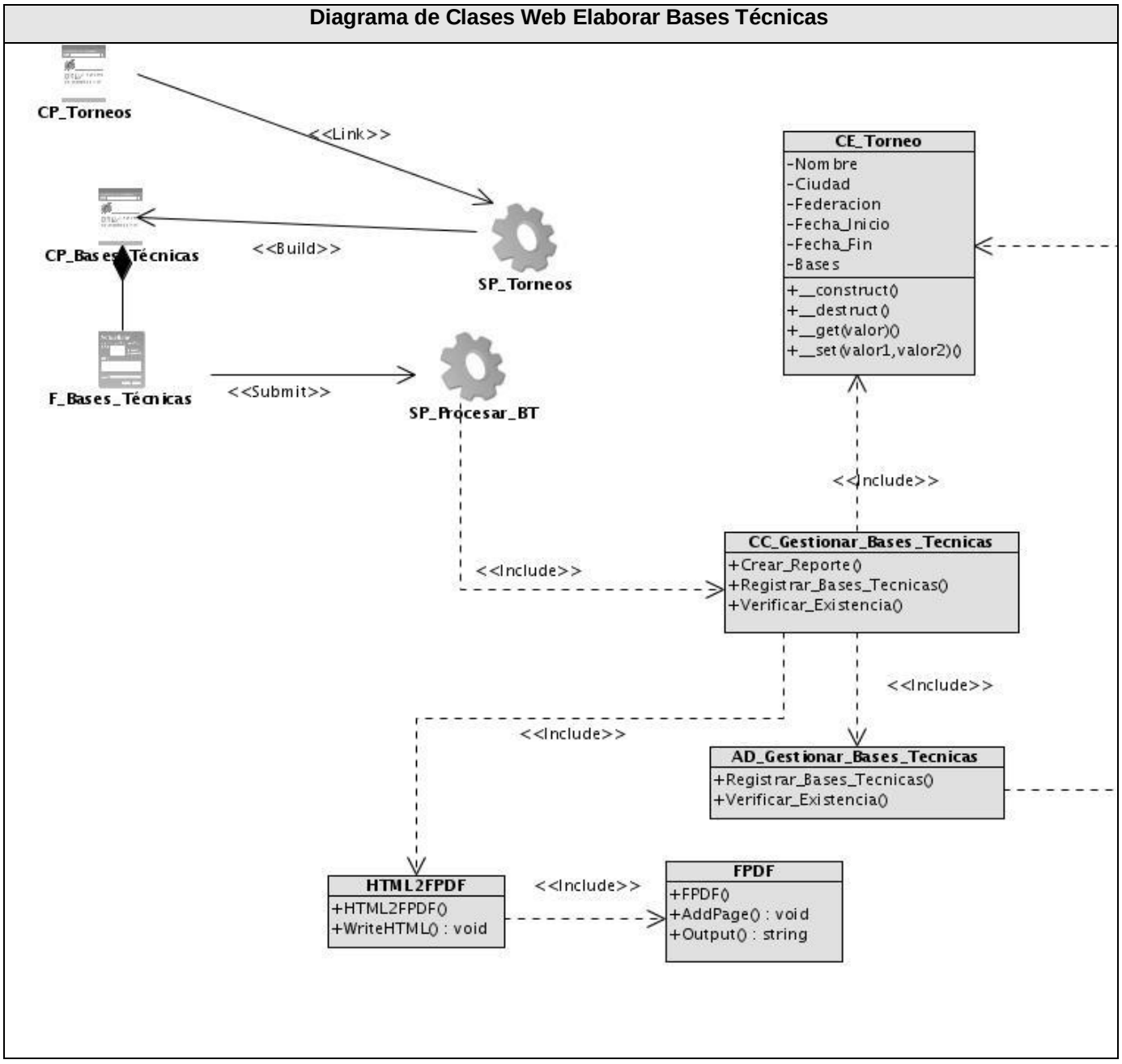

Figura 20: Diagrama de Clases Web del CUS Elaborar Bases Técnicas del Torneo.

3.3.5 Diagrama de Clases Web para el CUS Mostrar Cuadro Sinóptico.

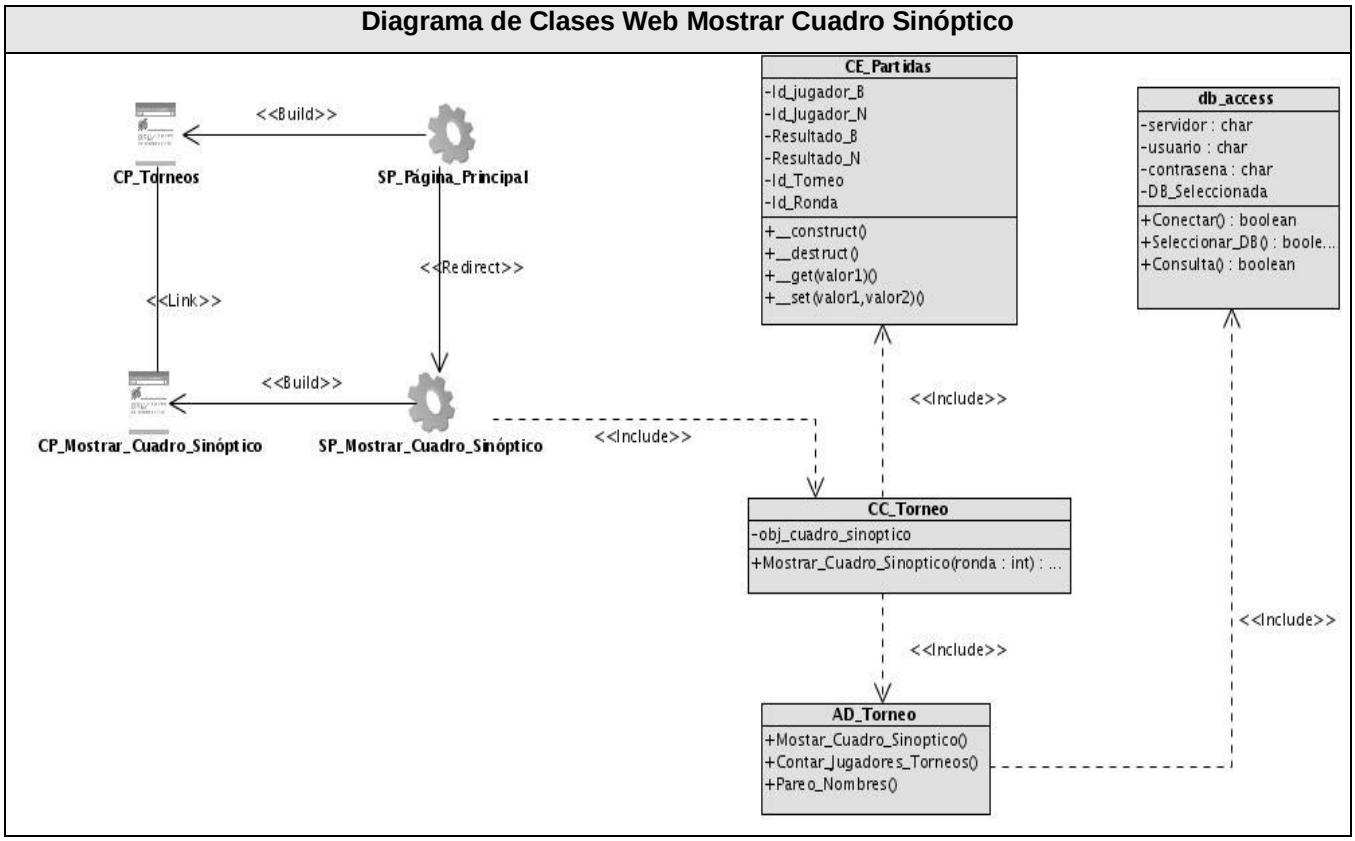

Figura 21: Diagrama de Clases Web del CUS Mostrar Cuadro Sinóptico.

3.3.6 Diagrama de clases Web del CUS Mostrar Pareo.

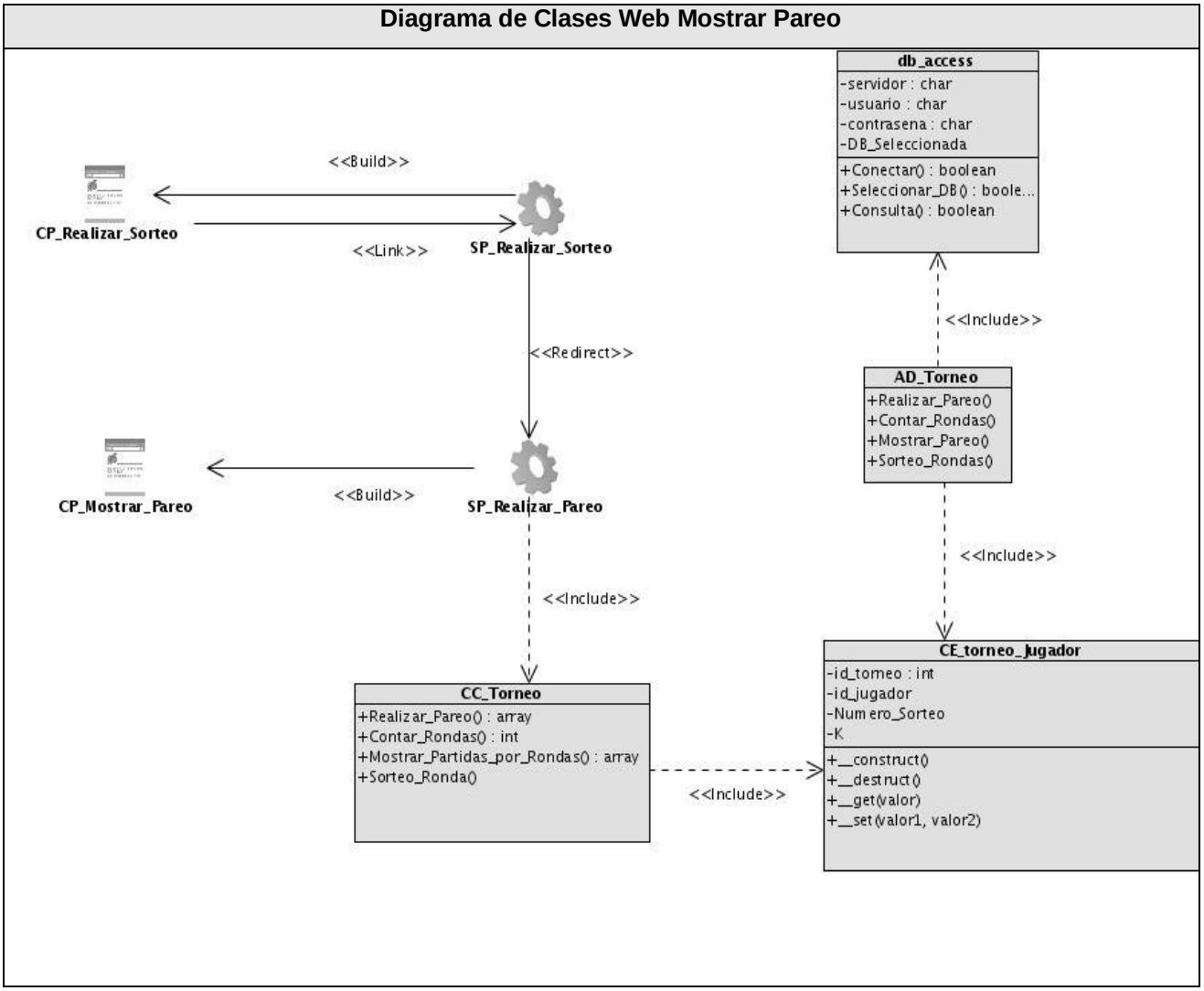

Figura 22: Diagrama de Clases Web del CUS Mostrar Pareo.

3.4 Diagramas de Interacción.

El modelo de diseño descrito para el desarrollo del sistema describe los diagramas de secuencia cada una de las realizaciones de los casos de uso, en esta sesión solo se representan cuatro diagramas, los cuales están entre los más importantes del sistema.

3.4.1 Diagrama de Secuencia del CUS Realizar Sorteo.

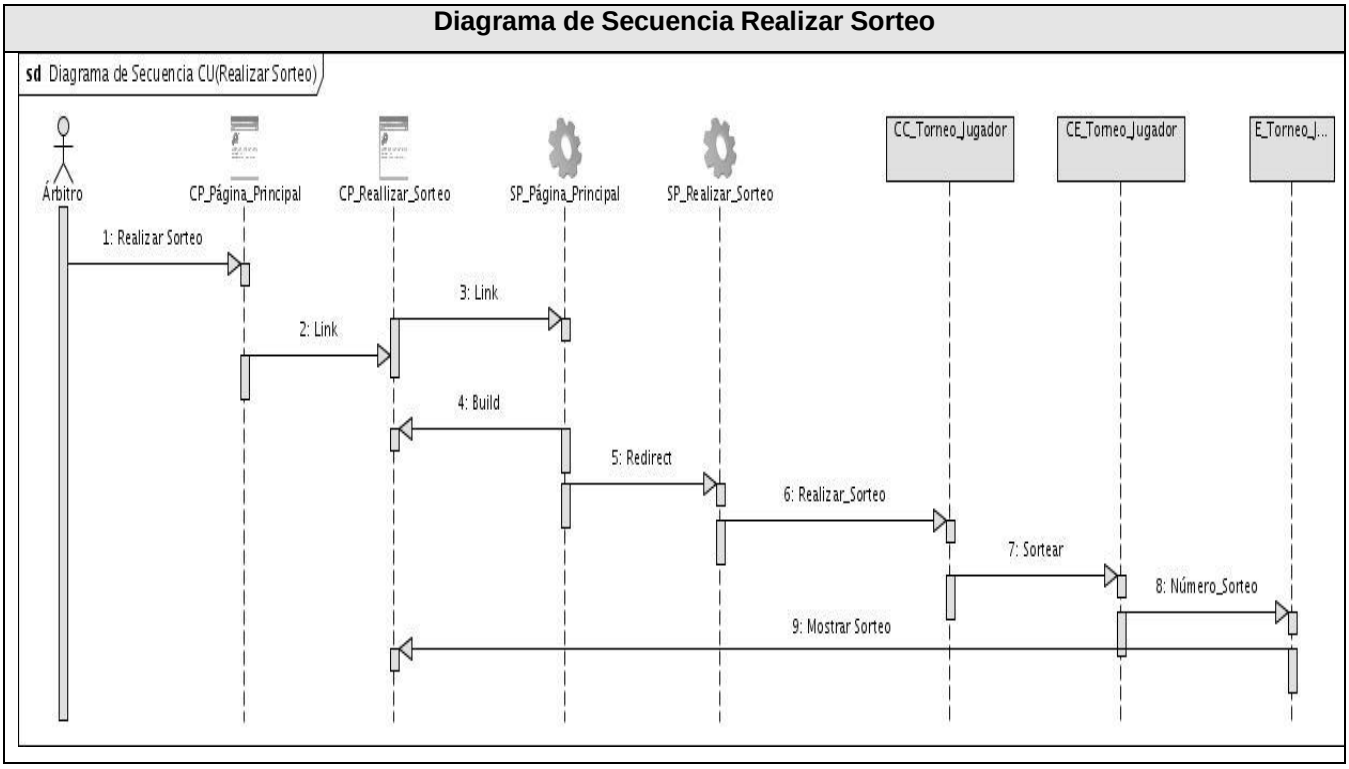

Figura 23: Diagrama de Secuencia del CUS Realizar Sorteo.

3.4.2 Diagrama de Secuencia del CUS Elaborar Bases Técnicas.

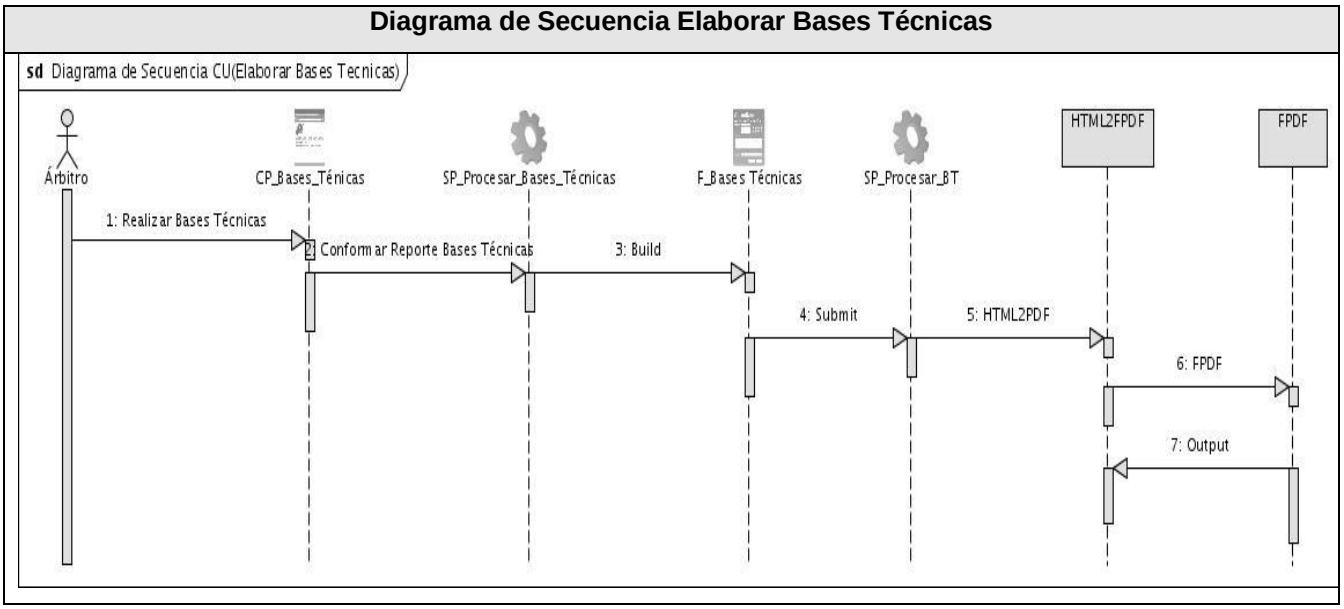

Figura 24: Diagrama de Secuencia del CUS Elaborar Bases Técnicas.

3.4.3 Diagrama de Secuencia del CUS Mostrar Cuadro Sinóptico.

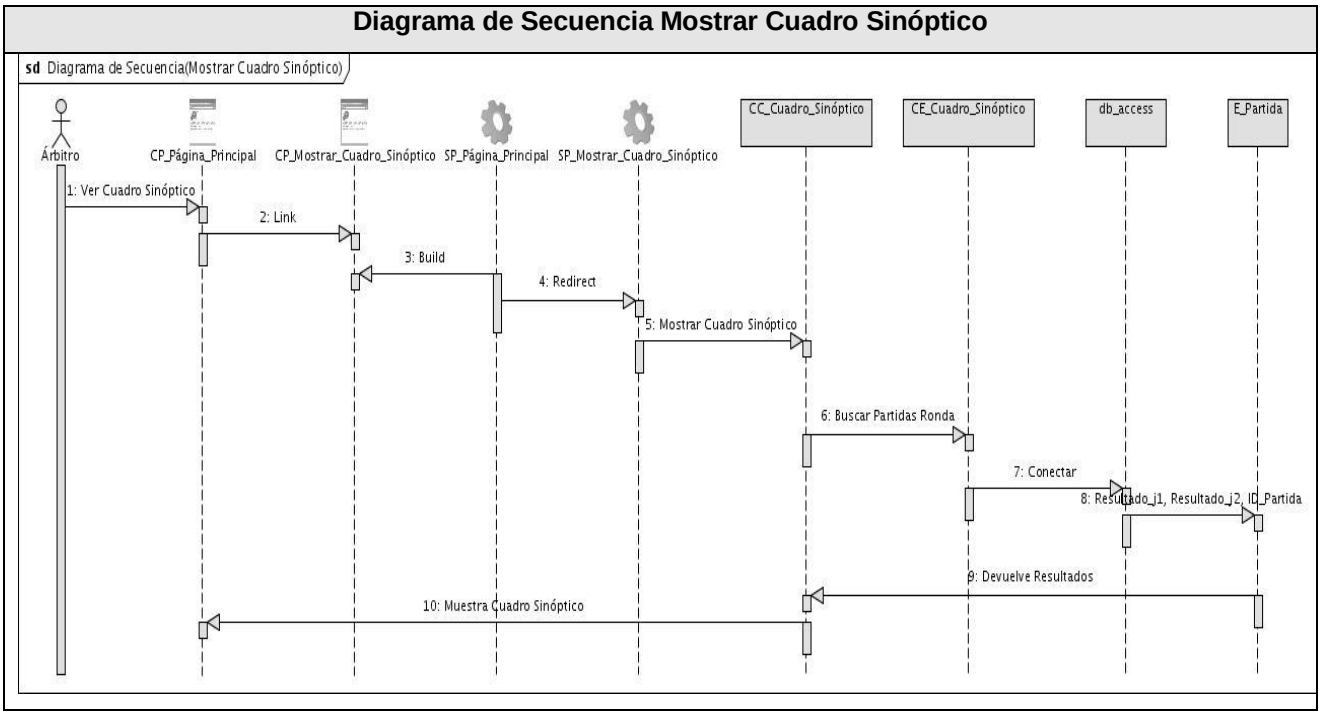

Figura 25: Diagrama de Secuencia del CUS Mostrar Cuadro Sinóptico.

3.4.4 Diagrama de Secuencia del CUS Mostrar Pareo.

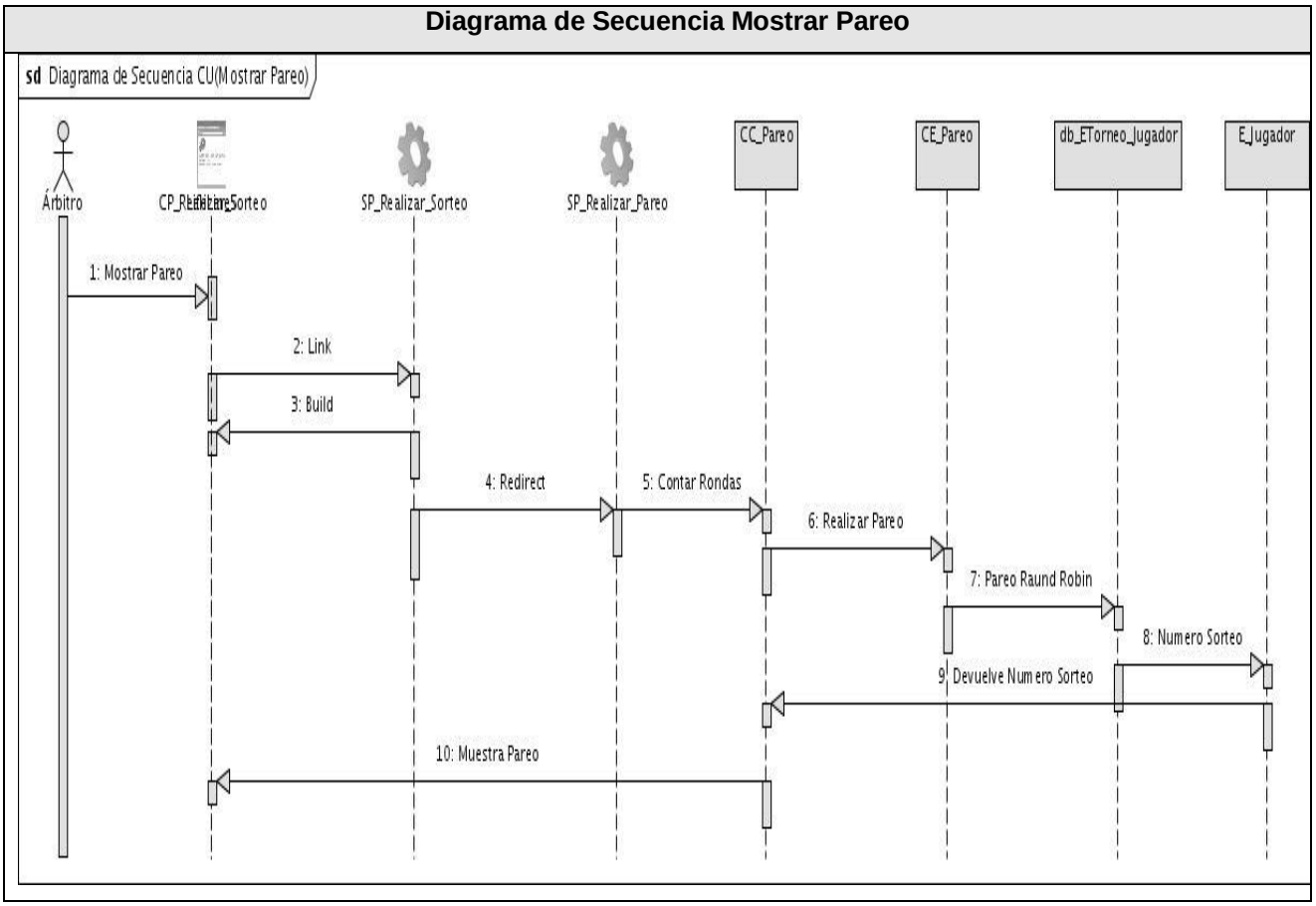

Figura 26: Diagrama de Secuencia del CUS Mostrar Pareo.

3.5 Descripción Textual de las clases Web.

La descripción textual de las clases del diseño permiten comprender el alcance y la responsabilidad de cada una de estas dentro del sistema, con dicha descripción se puede conocer qué funcionalidad específica realiza cada clase, además de la información que cada una de esta maneja.

3.5.1 Clase Controladora CC_Gestionar_Arbitro.

Tabla 14: Descripción de la Clase Controladora CC_Gestionar_Arbitro.

Nombre	CC_Gestionar_Arbitro	
Tipo de Clase	Controladora	
Atributo		**Tipo**
Responsabilidad		
	Insertar _ Arbitro ()	
	Actualizar _ Arbitro()	
	Seleccionar_Todo_Arbitro()	
	Buscar_ Arbitro()	
	Eliminar_ Arbitro()	
	Seleccionar_Todo_Arbitro()	
Descripción	Es la clase que se encarga de gestionar todo lo referente a los árbitros, que serán los usuarios del sistema.	

3.5.2 Clase Controladora CC_Gestionar_Jugador.

Tabla 15: Descripción de la Clase Controladora CC_Gestionar_Jugador.

Nombre	CC_Gestionar_Jugador	
Tipo de Clase	Controladora	
Atributo		**Tipo**
Responsabilidad		
	Insertar _ Jugador()	
	Seleccionar_ Jugador()	

	Seleccionar_Todo_Jugador ()
	Actualizar_ Jugador ()
	Eliminar_ Jugador ()
	Buscar_ Jugador ()
Descripción	Es la clase que se encarga de gestionar todo lo referente a los jugadores del sistema.

3.5.3 Clase Controladora CC_Gestionar_Torneo.

Tabla 16: Descripción de la Clase Controladora CC_Gestionar_Torneo.

Nombre	CC_Gestionar_Torneo
Tipo de Clase	Controladora
Atributo	**Tipo**
Responsabilidad	
	Ver_Todos_Torneos()
	Insertar_Torneo()
	Buscar_Torneo()
	Eliminar_Torneo()
	Adicionar_Jugadores_Torneo()
	Insertar_Jugadores_Torneo(id_jugador)
	Buscar_Jugadores_Torneo()
	Eliminar_Jugador_Torneo()
	Sortear()
	Buscar_Jugadores_Torneo_Ordenados()
	Limpiar_Sorteo()
	Mostrar_Datos_del_Torneo()
	Realizar_Pareo()
	Pareo_Numero_Sorteo()
	Contar_Jugadores_Torneo()
	Insertar_Ronda()

	Insertar_Resultado_Partida()
	Buscar_Partidas_Ronda()
	Actualizar_Partidas()
	Mostrar_Cuadro_Sinoptico()
	Sorteo_Nombre()
Descripción	Es la clase controladora más importante del sistema debido a que se encarga de manejar todas las operaciones que se realizan en un torneo.

3.5.4 Clase de acceso a dato Acceso_Datos.

Tabla 17: Descripción de la Clase de acceso a datos Acceso_Datos.

Nombre	Acceso_Datos	
Tipo de Clase	DAO	
Atributo		**Tipo**
Host		String
Usser		String
Password		String
Base_Datos		String
Conexion		String
Responsabilidad		
	Ejecutar_Consulta()	
	Cerrar_Conexion ()	
	Fetch_Array()	
Descripción	Clase encargada de realizar la conexión a la base de datos y ejecutar las consultas.	

3.5.5 Clase de acceso a datos AD_Gestionar_Arbitro.

Tabla 18: Descripción de la Clase de acceso a datos AD_Gestionar_Arbitro.

Nombre	Gestionar_Arbitro	
Tipo de Clase	DAO	
Atributo		**Tipo**
Responsabilidad		
	Insertar_Arbitro_BD(arbitro)	
	Actualizar_Arbitro_BD(arbitro, id_arbitro)	
	Eliminar_Arbitro(id_arbitro)	
	Seleccionar_Todos_Arbitros()	
	Buscar_Arbitro()	
Descripción	Clase encargada de controlar el acceso a los datos del caso de uso Gestionar Árbitro.	

3.5.6 Clase de acceso a datos AD_Gestionar_Jugador.

Tabla 19: Descripción de la Clase de acceso a datos AD_Gestionar_Jugador.

Nombre	AD_Gestionar_Jugador	
Tipo de Clase	DAO	
Atributo		**Tipo**
Responsabilidad		
	Insertar_Jugador_BD(jugador)	
	Eliminar_Jugador(id_ Jugador)	
	Buscar_Jugador()	
	Seleccionar_Todos_Jugador ()	
Descripción	Clase encargada de controlar el acceso a los datos del caso de uso Gestionar Jugador.	

3.5.7 Clase de acceso a datos AD_Gestionar_Torneo.

Tabla 20: Descripción de la Clase de acceso a datos AD_Gestionar_Torneo.

Nombre	AD_Gestionar_Torneo
Tipo de Clase	DAO
Atributo	**Tipo**
Responsabilidad	
	Ver_Torneos()
	Calcular_Estado_Torneo(fecha_i, fecha_f, id_torneo)
	Insertar_Torneo(torneo)
	Buscar_Torneo(parametro)
	Eliminar_Torneo(id_torneo)
	Buscar_Jugadores_Torneo(id_torneo)
	Eliminar_Jugador_Torneo(id_torneo, id_jugador)
	Realizar_Sorteo(id_torneo, id_jugador)
	Insertar_Sorteo(id_torneo, id_jugador, numero)
	Pareo_Round_Robin()
	Buscar_Ronda(id_ronda, id_torneo)
	Actualizar_Ronda(ronda)
	Insertar_Partida(partida)
	Buscar_Nombre_Jugador_ID(id_jugador)
	Actualizar_Partida_BD(id_partida, result_b, result_n)
	Cuadro_Sinoptico(id_torneo)
	Ver_Resultado_Partida(ronda, id_jugador1, id_jugador2)
	Ver_Rondas_Actuales()
Descripción	Es la clase encarga del acceso a los datos de los casos de usos fundamentales del sistema.

3.5.8 Clase entidad CE_Arbitro.

Tabla 21: Descripción de la clase entidad CE_Arbitro.

Nombre	**CE_Arbitro**	
Tipo de Clase	Entidad	
Atributo		**Tipo**
usuario		string
password		string
nombre		string
titulo		string
rol		int
Responsabilidad		
Nombre	__construct(usuario, password, nombre, titulo, rol)	
	__destruct()	
	__set($valor1,$valor2)	
	__get($valor)	
Descripción	Clase entidad que representa la tabla árbitro en la base de datos	

3.5.9 Clase entidad CE_Jugador.

Tabla 22: Descripción de la clase entidad CE_Jugador.

Nombre	CE_Jugador	
Tipo de Clase	Entidad	
Atributo		**Tipo**
nombre		Int
sexo		char
id_titulo		string
codigoFide		string
id_federacion		int
fecha_nacimiento		Date
rating_elo		int
Responsabilidad		

	__get($valor)
	__set($valor1,$valor2)
Descripción	Clase entidad que representa la tabla jugador en la base de datos

3.5.10 Clase entidad CE_Torneo.

Tabla 23: Descripción de la clase entidad CE_Torneo.

Nombre	CE_Torneo	
Tipo de Clase	Entidad	
Atributo		**Tipo**
nombre		string
ciudad		string
federacion		string
fecha_inicio		Date
fecha_fin		Date
arbitro_principal		int
Responsabilidad		
Nombre	__construct(nombre, ciudad, federacion, fecha_inicio, fecha_fin, arbitro_principal)	
	__destruct()	
	__get($valor)	
	__set($valor1,$valor2)	
Descripción	Clase entidad que representa la tabla torneo en la base de datos	

3.6 Diagrama de clases persistentes.

Las clases persistentes son capaces de guardar su estado en un medio permanente; lo cual está dado por el almacenamiento físico de la información de la clase, para la copia de seguridad en caso del fracaso del sistema, o para el intercambio de información.

A continuación se muestra el diagrama de clases persistentes del sistema:

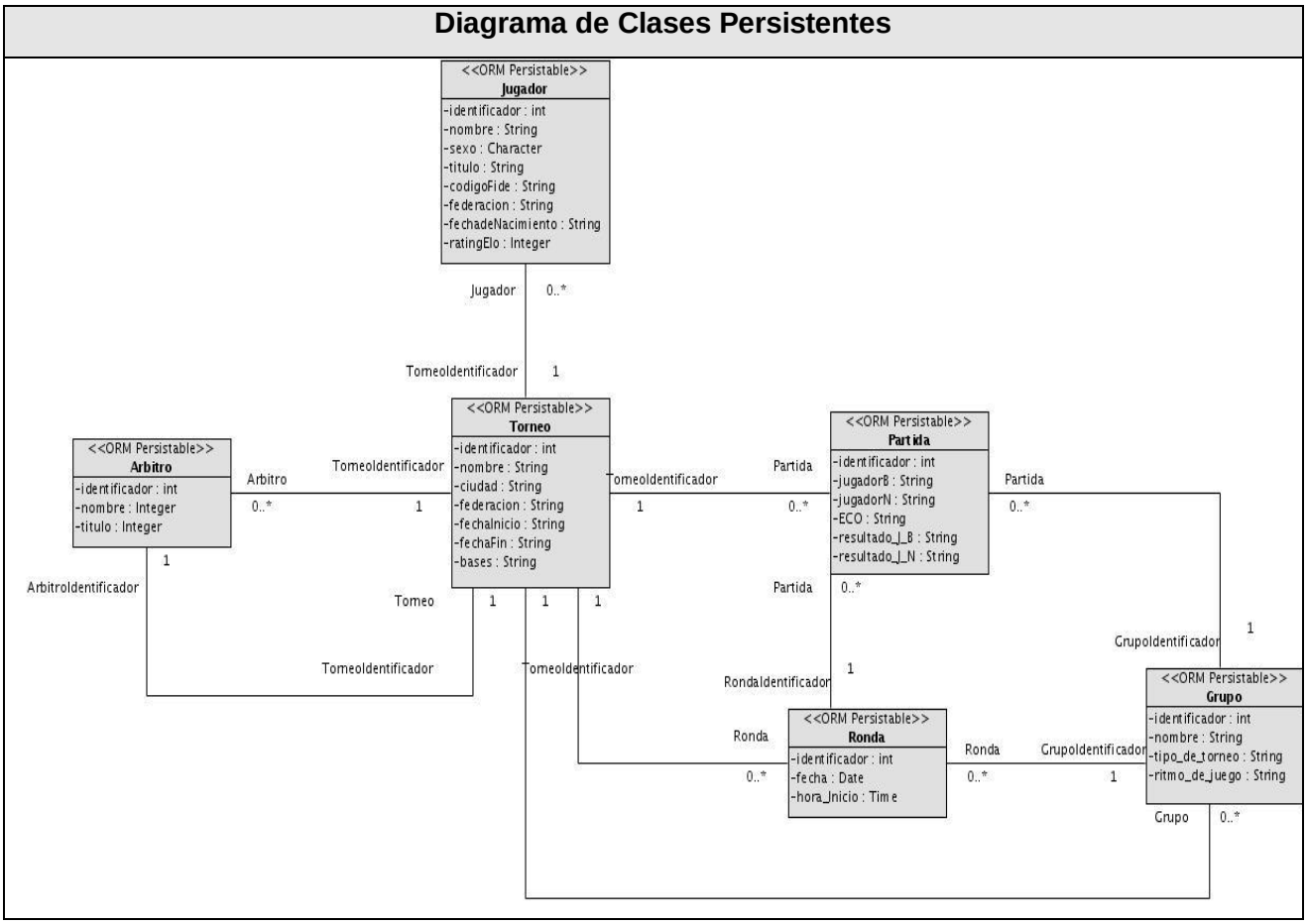

Figura 27: Diagrama de clases persistentes.

Conclusiones

En este capítulo fueron expuestos diferentes elementos que ilustran cómo está construido el sistema, en términos de clases del análisis y del diseño. Este último dio la posibilidad de comprender la lógica del sistema en general. Por otro lado, fueron detalladas textualmente cada una de las clases Web, lo cual trae consigo que se conozca la responsabilidad de cada una de estas y qué funcionalidad específica realizan, además de que fue presentado el diagrama de clases persistentes de la base de datos, que contiene la información física que se utilizó para la construcción de la aplicación.

CAPÍTULO 4: IMPLEMENTACIÓN

4.1 Introducción.

Para el desarrollo de este capítulo se tuvieron en cuenta algunos artefactos que fueron generados en el flujo de análisis y diseño. A partir de los cuales se desarrolló la implementación del sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares.

4.2 Modelo de Implementación.

El modelo de implementación constituye la vista de Implementación de la arquitectura, y como tal guía las labores de construcción del sistema. Este contiene fundamentalmente los subsistemas de implementación, incluyendo las dependencias y otras informaciones necesarias para su utilización.

Para lograr una mejor comprensión de los componentes que forman el sistema, se presentarán los diagramas de componentes que describen los elementos físicos y lógicos del sistema y sus relaciones.

Los diagramas que se presentan a continuación tienen como objetivo figurar la estructura general de la aplicación en desarrollo, en términos de componentes.

4.2.1 Diagrama de Componentes del CUS Realizar Sorteo.

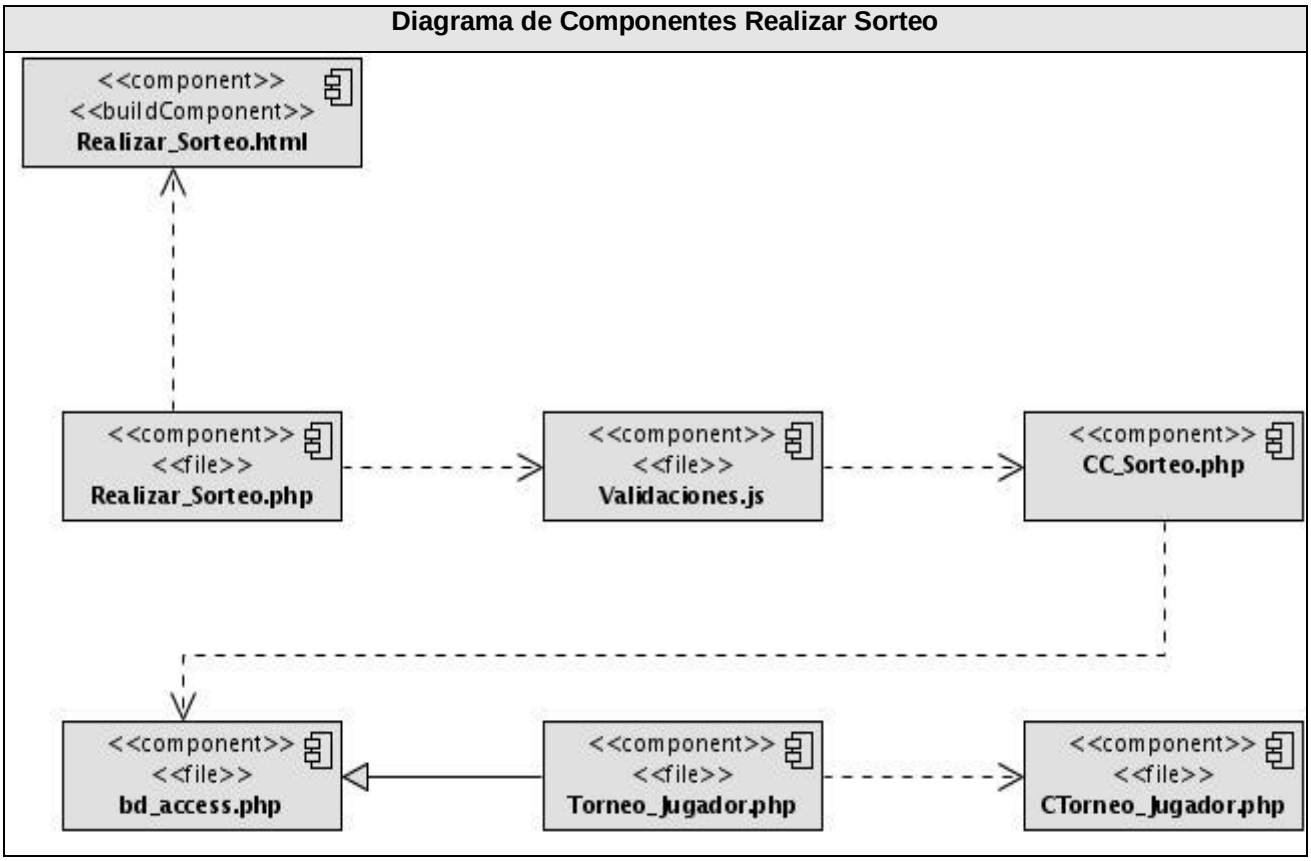

Figura 28: Diagrama de Componentes del CUS Realizar Sorteo.

4.2.2 Diagrama de Componentes del CUS Autenticar.

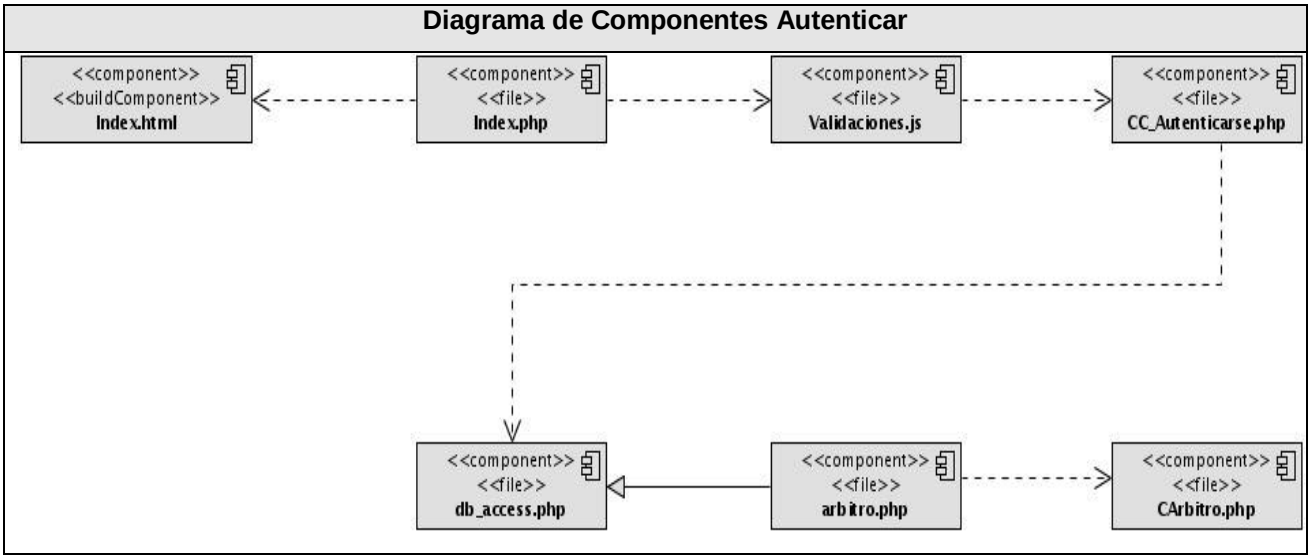

Figura 29: Diagrama de Componentes del CUS Autenticar.

4.2.3 Diagrama de Componentes del CUS Elaborar Bases Técnicas del Torneo.

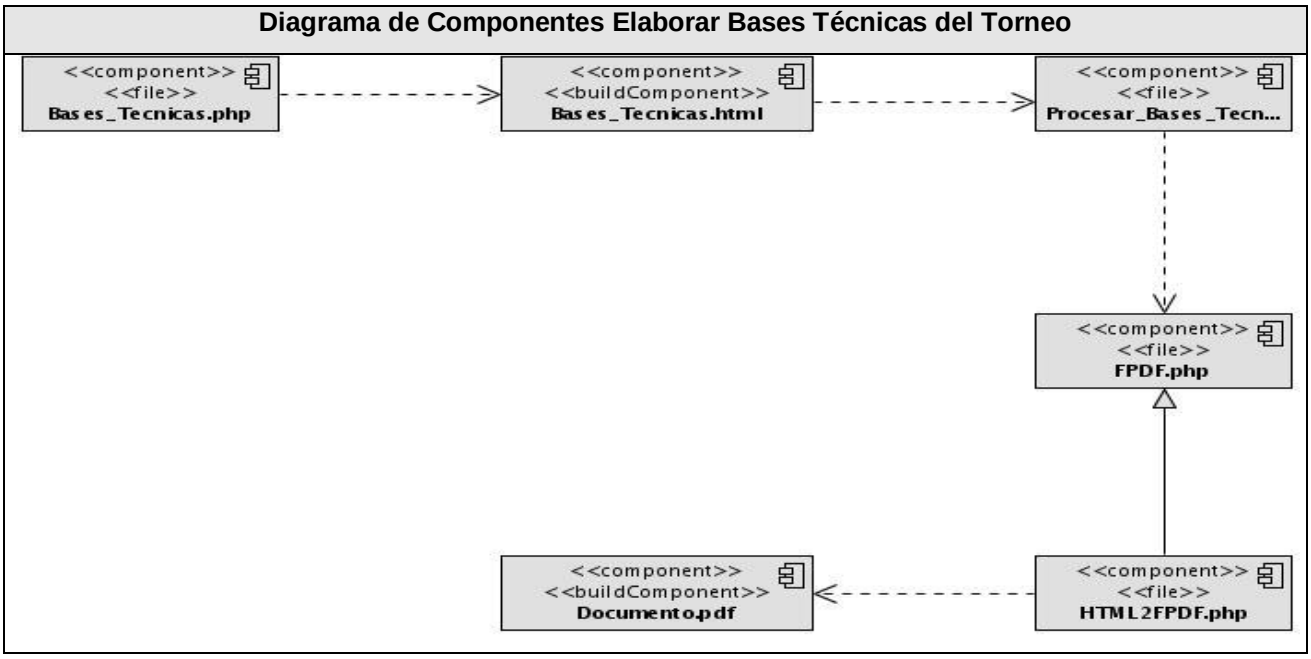

Figura 30: Diagrama de Componentes del CUS Elaborar Bases Técnicas del Torneo.

4.2.4 Diagrama de Componentes del CUS Mostrar Cuadro Sinóptico.

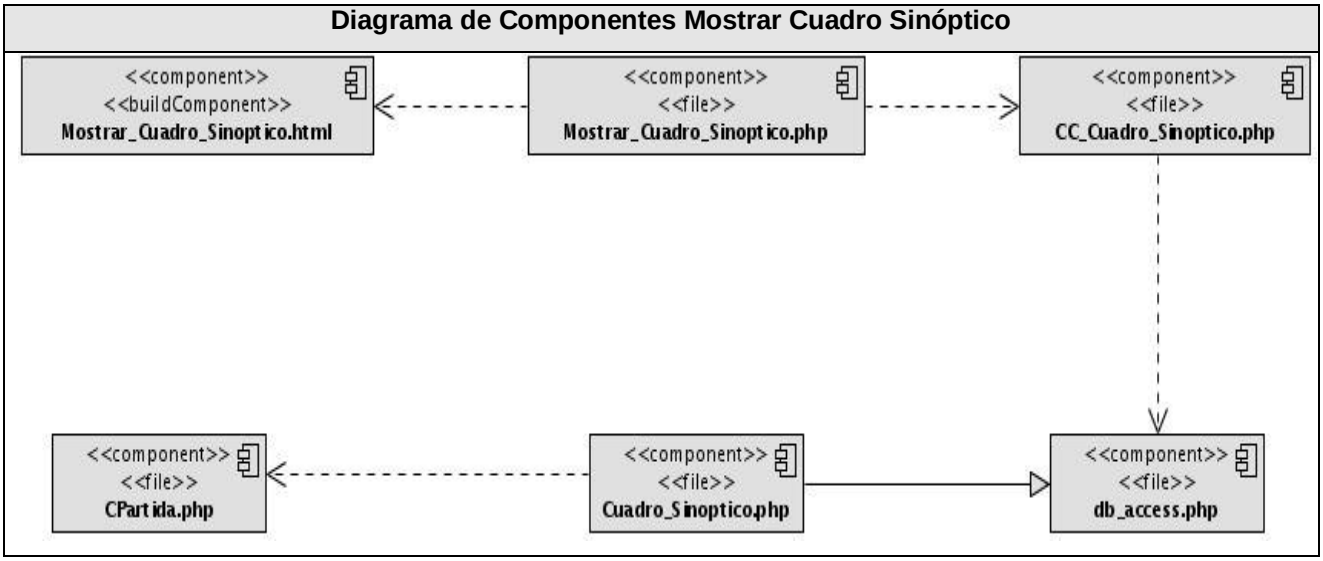

Figura 31: Diagrama de Componentes del CUS Mostrar Cuadro Sinóptico.

4.2.5 Diagrama de componentes del CUS Mostrar Pareo.

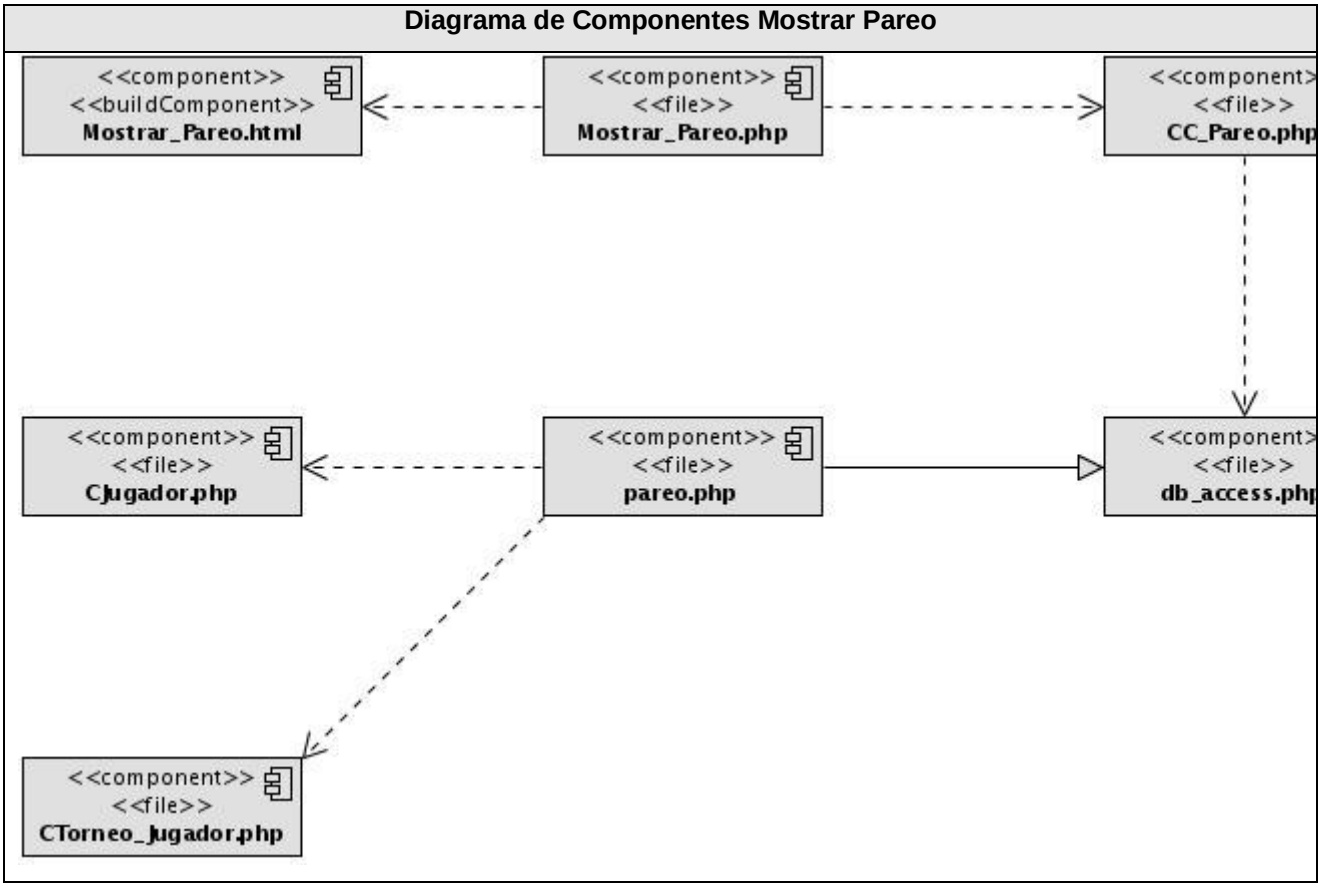

Figura 32: Diagrama de Componentes del CUS Mostrar Pareo.

4.3 Modelo de Despliegue.

El diagrama de despliegue representa la arquitectura de tiempo de ejecución de los procesadores, dispositivos y los componentes de software que se ejecutan en esa arquitectura. Es la última descripción física de la topología del sistema y describe la estructura de las unidades de hardware. Además, representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Un nodo es un recurso de ejecución tal como un procesador, un dispositivo o memoria. En los procesadores es donde se encuentran alojados los componentes.

4.3.1 Diagrama de Despliegue.

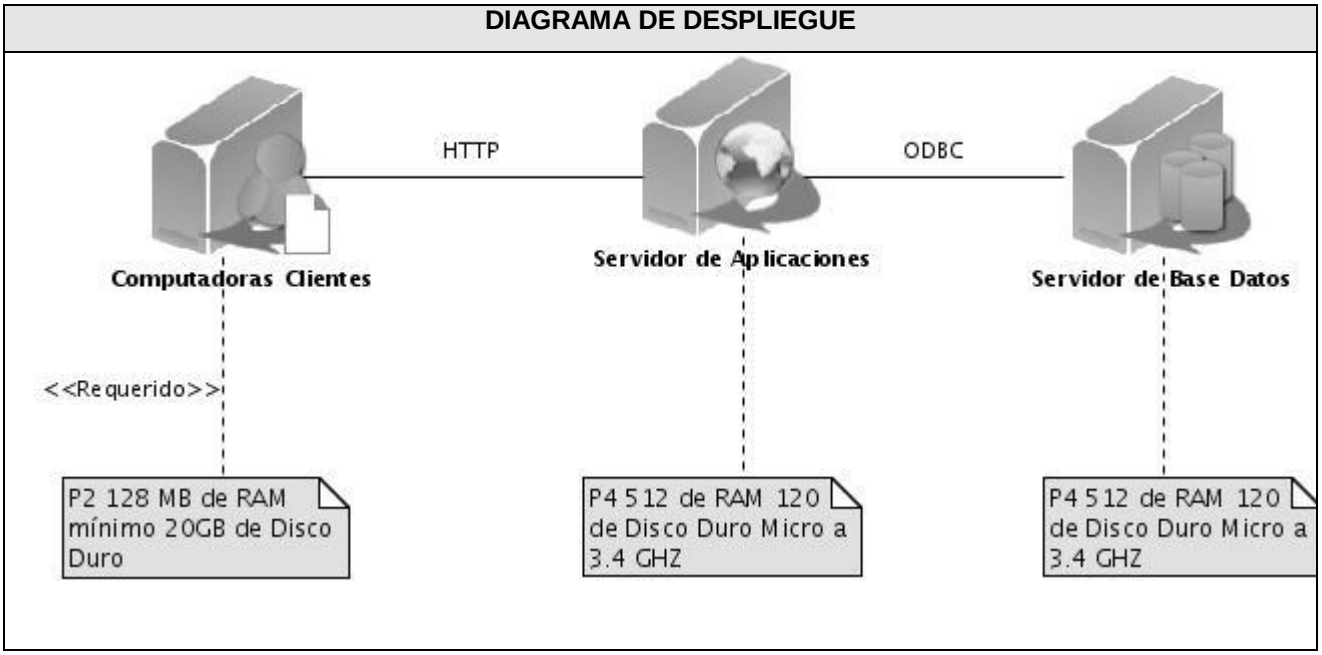

Figura 33: Diagrama de Despliegue.

Conclusiones.

En este capítulo se mostró como a través de la implementación, se produjo un refinamiento de la vista de la arquitectura del modelo de despliegue, donde los componentes ejecutables fueron asignados a nodos.

Se utilizaron diagramas de componentes para representar a través de un grafo los componentes de software unidos por medio de relaciones de dependencia; con los cuales se modeló la vista estática de un sistema. Además sirvieron para mostrar la organización y las dependencias lógicas entre un conjunto de componentes software. En este momento, ya se tiene el producto de software.

CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

5.1- Introducción.

En la realización de todo proyecto es de suma importancia realizar la estimación del esfuerzo humano, el tiempo de desarrollo que se empleará así como su costo. En el presente capítulo se llevará a cabo un estudio de la factibilidad del sistema utilizando el método de análisis por Punto de Casos de Uso.

5.2- Planificación mediante Puntos de Casos de Uso.

La estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner de Objector y posteriormente refinado por muchos otros autores. Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores.

Cálculo de Puntos de Caso de Uso sin ajustar.

UUCP = UAW + UUCW

Donde:

UUCP: Puntos de Casos de Uso sin ajustar.
UAW: Factor de Peso de los Actores sin ajustar.
UUCW: Factor de Peso de los Casos de Uso sin ajustar.

Calculando UAW: Factor de Peso de los actores.

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema.

Tabla 24: Factor de peso de los actores.

Tipo de Actor	Descripción	Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface).	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

En el caso del sistema desarrollado son 3 actores de tipo complejo: Árbitro, Administrador y Usuario.

UAW = 3 * 3

UAW = 9

Calculando UUCW: Factor de Peso de los Casos de Uso sin ajustar.

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo.

Una transacción está representada por uno o más pasos del flujo de eventos principal del Caso de Uso, pudiendo existir más de una transacción dentro del mismo Caso de Uso.

Tabla 25: Factor de Peso de los Casos de Uso sin ajustar.

Tipo de Caso de Uso	Descripción	Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones.	5
Medio	El Caso de Uso contiene de 4 a 7 transacciones.	10
Complejo	El Caso de Uso contiene más de 8 transacciones.	15

Tabla 26: Transacciones y peso de los casos de uso.

Caso de Uso	Transacciones	Peso
Autenticar Usuarios	2	5
Gestionar Árbitro	4	10
Gestionar Partidas	4	10
Gestionar Torneo	4	10
Gestionar Jugador	4	10
Gestionar Federación	6	10
Mostrar Cuadro Sinóptico	4	10
Gestionar Título de Árbitro	4	10
Elaborar Bases Técnicas	2	5
Conformar Reporte	2	5
Realizar Sorteo	4	10
Buscar Jugador	2	5
Realizar Pareo	8	15
Mostrar Partidas por Ronda	6	10

UUCW = 4*5+9*10+1*15=125

Cálculo de Puntos de Caso de Uso sin ajustar.

UUCP = UAW + UUCW
UUCP = 9+125
UUCP = 134

Calculando lo Puntos de Caso de Uso ajustados.

Los puntos de Casos de Uso se calculan con la siguiente fórmula:

UCP = UUCP * TCF *EF

Donde:

UCP: Puntos de Casos de Uso ajustados.
UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.
EF: Factor ambiente.

Calculando TCF: Factor de complejidad técnica.

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante.

En la siguiente tabla se muestran los factores, el peso de cada uno de ellos y el valor asignado:

Tabla 27: Factor de complejidad técnica.

Factor	Descripción	Peso	Valor asignado
T1	Sistema distribuido	2	0
T2	Tiempo de respuesta	1	3
T3	Eficiencia del usuario final	1	3
T4	Procesamiento interno complejo	1	2
T5	El código debe ser reutilizable	1	1
T6	Facilidad de instalación	0.5	1
T7	Facilidad de uso	0.5	4
T8	Portabilidad	2	2
T9	Facilidad de cambio	1	3
T10	Concurrencia	1	4
T11	Incluye objetivos especiales de seguridad	1	3
T12	Provee acceso directo a terceras partes	1	0
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	3

TCF = 0.6 + 0.01 * ∑ (Pi * Vi)

TCF = 0.6+0.01*28.5

TCF = 0.885

Calculando EF: Factor de ambiente.

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del Factor de ambiente. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5.

Tabla 28: Factor de ambiente.

Factor	Descripción	Peso	Valor asignado
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	3
E2	Experiencia en la aplicación.	0.5	3
E2	Experiencia en orientación a objetos	1	4
E4	Capacidad del analista líder.	0.5	4
E5	Motivación.	1	5
E6	Estabilidad de los requerimientos.	2	4
E7	Personal part-time.	-1	4
E8	Dificultad del lenguaje de programación.	-1	2

EF = 1.4 - 0.03 * ∑ (Pi * Vi)
EF = 1.4 - 0.03 * 20.5
EF = 0.785

Calculando los Puntos de Caso de Uso ajustados

UCP = UUCP * TCF *EF
UCP = 134 * 0.885 * **0.785**
UCP = 93
De los Puntos de Casos de Uso a la estimación del esfuerzo.

El esfuerzo en horas-hombre viene dado por:

E = UCP * CP

Donde:

E: Esfuerzo estimado en horas-hombre.
UCP: Puntos de Casos de Uso ajustados.
CF: Factor de conversión.

Para calcular el factor de conversión es necesario tener como guía el siguiente criterio:

- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.
- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.
- Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.
- Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre.
- Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

Resumiendo:

CF = 20 horas-hombre (si Total EF ≤ 2)
CF = 28 horas-hombre (si Total EF = 3 ó Total EF = 4)
CF = abandonar o cambiar proyecto (si Total EF ≥ 5)

En el presente caso no existen factores por debajo de 3 desde E1 hasta E6 y solo un factor de E7 y E8 esta por encima de 3, por lo que el total de factores a tener en cuenta es 1 y el factor conversión seria de 20 horas-hombre.

Calculando el esfuerzo del flujo de trabajo:

E = UCP * CP
E = 93 * 20
E = 1860 horas-hombre.

Se debe tener en cuenta que éste método proporciona una estimación del esfuerzo en horas-hombre contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso.

Finalmente, para una estimación más completa de la duración total del producto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software. Además se considera que este esfuerzo representa un porcentaje del esfuerzo total del proyecto, de acuerdo a los valores porcentuales:

Tabla 29: Distribución porciento-horas-hombre.

Actividad	Porcentaje	Horas/Hombre
Análisis	20 %	744
Diseño	30 %	1616
Programación	50 %	1860
Total	100 %	3720

Convirtiendo a Horas/Hombre.

Como la jornada laboral de un día de trabajo es de 8 horas y en un mes se trabaja un aproximado de 24 días, entonces una persona en 1 mes trabaja 192 horas, por tanto:
Et= E (Horas-Hombres)/192horas-mes
Quedaría, Et= 3720horas-Hombres/192horas-mes = 19,4 =20 mes-hombres.

Si en el proyecto trabajan dos hombres entonces el tiempo de desarrollo es:
Tiempo de desarrollo= Et /cantidad de hombres
Tiempo de desarrollo= 20/2= 10 = 10 meses
El tiempo a emplear para el desarrollo de la aplicación es de 10 meses.

Salario

Para determinar el salario mensual se tiene en cuenta que los desarrolladores del sistema pueden ser ingenieros recién graduados, por lo que se toma como salario mensual: $225.

5.3- Costo.

Como se definió anteriormente el salario promedio de un ingeniero, y son dos personas los desarrolladores del proyecto, entonces para hallar el costo total sería:

Ct= Salario mensual*Cantidad de hombres*Tiempo de desarrollo
Ct= $225*2*10
Ct= $4500

5.4 Tangibles.

Teniendo en cuenta que la aplicación en cuestión no es un producto desarrollado para la comercialización, ya que surge con la idea de apoyar a la universidad, específicamente, a la Cátedra de Ajedrez no es válido mencionar beneficios económicos.

Se puede decir que el costo por desarrollar la aplicación es de $4500 MN (moneda nacional) ó 180 CUC (convertible), el cual es perfectamente reparable si en un futuro se comercializara.

5.5 Intangibles.

Como beneficios intangibles de la aplicación Módulo Arbitraje se señalan los siguientes:

- Se agiliza el proceso de arbitraje para los torneos de ajedrez todos contra todos.
- Permite garantizar una mayor centralización de la información que se maneja en un torneo de ajedrez.
- Se realizan todas las actividades que normalmente un árbitro las tendría que hacer de forma manual.

5.6 Análisis costo-beneficio.

Dadas las características de la aplicación se puede decir que se requiere de un tiempo aceptable para el desarrollo del software, además no necesita de grandes gastos de recursos.

El ambiente del software se ve favorecido con un diseño que permite al usuario desarrollar una perfecta navegabilidad sin motivos de perderse dentro del mismo.

Es factible desarrollar una aplicación para informatizar y centralizar la información que manejan en los torneos los árbitros, ya que hace este trabajo menos tedioso.

Conclusiones.

En este capítulo se realiza un estudio de la factibilidad del producto, se puntualizan los costos a incurrir, los recursos humanos implicados, el tiempo de desarrollo y los beneficios intangibles que aporta la terminación del producto “Concepción y Desarrollo del Módulo Arbitraje del Proyecto Infodrez”. Además con la realización de este capítulo se expresa claramente la ventaja que implica la implementación de este producto.

Los resultados obtenidos al concluir los cálculos y estimaciones realizadas en este capítulo, se muestran en la siguiente tabla:

Tabla 30: Resumen general.

Parámetros	Valores
Esfuerzo	20 mes-hombre
Tiempo de desarrollo	10 meses
Cantidad de hombres	2 hombres
Salario	$225
Costo Total MN	$4500
Costo Total CUC	$180

CONCLUSIONES

Como resultado de este trabajo se obtuvo un sistema informático, el cual le facilitará las actividades que desempeñan los árbitros en la cátedra de ajedrez “Remberto Fernández” de la Universidad de las Ciencias Informáticas, como es el caso de registrar todos los torneo que en la universidad se realizan, así como automatizar las diferentes actividades que ellos realizan durante el transcurso de cualquier torneo.

Dentro de estas actividades se encuentran: realizar el sorteo, mostrar el pareo, mostrar el cuadro sinóptico del torneo en el transcurso de cualquier ronda, elaborar las bases técnicas entre otras actividades de vital importancia y que por la cantidad de trabajo que estas llevan se torna un poco tediosa el desarrollo de las mismas. Con el desarrollo de la aplicación estas actividades se desarrollarán de forma automatizada de una manera rápida y eficiente.

RECOMENDACIONES

A lo largo de este trabajo, se puede apreciar cómo se fueron cumpliendo cada uno de los objetivos trazados en el mismo, además de que se puede observar claramente el alcance y la forma en la que se le dio solución a varios de los problemas encontrados, no obstante, se realizan varias recomendaciones para aquellos que darán continuación a dicho trabajo las cuales se especifican a continuación:

- Desarrollar las funcionalidades de los diferentes tipos de torneos, ya que el presente trabajo solo concibe el torneo Todos contra Todos.
- Perfeccionar la forma en la que se realizan todas las validaciones en el sistema, con vista a dar solución a posibles imprevistos que puedan ocurrir durante la ejecución del mismo.
- En la elaboración de las bases técnicas que el reporte se realice en un documento editable y no como un .pdf.
- Generar un reporte del torneo tal y como lo solicita la FIDE.
- Implementar más sistemas de desempates.
- Migrar la base de datos a PostgreSQL.
- Mejorar la interfaz gráfica del sistema.

REFERENCIAS BIBLIOGRÁFICAS

CASTELLANO, F. JAVIER GARCIA. 2002. Perl. *http://flanagan.ugr.es/perl/index2.htm* [En línea] 2002.

CONCLASE. 1999. Hojas de Estilo en Cascada, nivel 1. *http://html.conclase.net/w3c/css1-es.html* [En línea] 1999.

DESARROLLOWEB. Manuales de Desarrollo web, diiseño y programación. *www.desarrolloweb.com* [En línea]

EGUILUZ PEREZ, JAVIER. Introducción a JavaScript.*http://www.librosweb.es/javascript/index.html* [En línea]

FOWLER, MARTIN. 1999-2006. La Nueva Metodología. *http://www.programacion.com/tutorial/nuevametodologia* [En línea] 1999-2006.

HETLAND, MAGNUS LIE. 1999. Python instantáneo. *http://www.arrakis.es/~rapto/AprendaPython.html* [En línea] 1999.

HIPERTEXT. 2007. Introducción a XML para Documentalistas. *http://www.hipertext.net/web/pag100.htm* [En línea] 2007.

MENDEZ, GUILLERMO. 1996. Introducción a HTML. *http://www.eis.uva.es/GuiaHTML/introHTML.html* [En línea] 1996.

MOLPECERES, ALBERTO. 2002. Procesos de desarrollo. *http://www.javahispano.org/contenidos/es/procesos_de_desarrollo/;jsessionid=3BFE6140CF058557810A36F837CC2E41* [En línea] 2002.

MONTALVO, M.M. 2002. XML el nuevo lenguaje universal. *http://www.congreso-info.cu/UserFiles/File/Info/Info2002/Ponencias/97.pdf* [En línea] 2002.

MySQL. 2008. MySQL. *http://es2.php.net/mysql* [En línea] 2008.

PENIN, ARTURO J.MENENDEZ. Metodologías avanzadas de Desarrollo de Software. *http://creaweb.ei.uvigo.es/creaweb/Asignaturas/MADS/apuntes/presentacionMADS_0708.pdf* [En línea]

PHP. 1999. PHP en castellano. *http://www.programacion.com/php/direcciones* [En línea] 1999.

SORIANO, MANUEL. 1998. SQL: Introducción al SQL. Instalación de PostgreSQL. *http://www.sindominio.net/ayuda/postgresql/tutorial-postgresql-1.html* [En línea] 1998.

WIKIPEDIA. 2008. Servidor HTTP Apache. *http://es.wikipedia.org/wiki/Servidor_HTTP_Apache* [En línea] 2008.

DESARROLLOWEB. 2005. Arquitectura-cliente-servidor. *www.desarrolloweb.com*. [En línea] 2005. http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

Hernandis, José Alberto. 2005. *www.versioncero.com*. [En línea] 2005. http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.

WIKIPEDIA. 2008. http://*es.wikipedia.org*. [En línea] 2008. http://es.wikipedia.org/wiki/RUP.

—. 2008. http://*es.wikipedia.org*. [En línea] junio de 2008. http://es.wikipedia.org/wiki/CASE.

—. 2007. http://*es.wikipedia.org*. [En línea] 2007. http://es.wikipedia.org/wiki/Zend_Studio.

GLOSARIO DE TÉRMINOS.

Aplicación WEB: Es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de internet o de una intranet.

CASE: Computer Aided Software Engineering.

CUN: Caso de uso del negocio.

CUS: Caso de uso del Sistema.

Internet: Sistema de redes de computación ligadas entre sí, con alcance mundial, que facilita servicios de comunicación de datos como registro remoto, transferencia de archivos, correo electrónico y grupos de noticias. Internet es una forma de conectar las redes de computación existentes que amplía en gran medida el alcance de cada sistema participante.

Intranet: Es una red de computadoras dentro de una red de área local (LAN) privada, empresarial o educativa que proporciona herramientas de internet.

Lenguajes de Scripting: Fueron diseñados para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados.

RUP: *Rational Unified Process* (Proceso Unificado de desarrollo). Metodología para el desarrollo de Software.

Servidor Web: Es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

SGBD: *Sistema de Gestión de Bases de Datos*. Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.

Software: Se refiere al equipamiento lógico o soporte lógico de un computador digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.

UCI: Universidad de las Ciencias Informáticas.

UML: Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

XML: *Extensible Markup Language*. Es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium. Orientado principalmente al almacenamiento, procesamiento y transmisión de mensajes.