Universidad de las Ciencias Informáticas

Facultad 8

Confección y desarrollo del Módulo Gestor de Estadísticas del proyecto Infodrez

Trabajo de Diploma para optar por el título de
Ingeniero en ciencias Informáticas

Autor(es): Carlos Abel Capeáns Hurtado
Robert Mengana Niebla

Tutor: Ing. Alison Muñoz Capote

Cotutor: MSc. Rafael Jiménez Rodríguez

Ciudad de La Habana
Junio del 2008
"Año 50 de la Revolución"

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 firmamos la presente a los ____ días del mes de _____ del año _____.

_____	_____
Firma del Autor	Firma del Autor
Carlos Abel Capeáns Hurtado	Robert Mengana Niebla
_____	_____
Firma del Tutor	Firma del Cotutor
Ing. Alison Muñoz Capote	MSc. Rafael Jiménez Rodríguez

AGRADECIMIENTOS

A la Revolución en primer lugar, por habernos dado la posibilidad de alcanzar uno de nuestros mayores sueños en la vida, el de formarnos como profesionales, lo cual no hubiese sido posible si no existiese la misma.

A nuestro tutor Alison y a todos los miembros del proyecto Infodrez que hicieron posible la realización de este trabajo.

A todos nuestros amigos y compañeros de aula, por la ayuda imprescindible que recibimos de todos y por el constante apoyo y muy en especial a Levian, Yaismel, Danyer y Berrillo por su constante disposición para ayudarnos ante cualquier situación que se nos presentara.

De Robert

A toda mi familia por su apoyo y preocupación constante, especialmente a mi mamá y a mi papá quienes de no haber sido por ellos no lo hubiera logrado.

A mi novia por haberme ayudado de la manera en que lo hizo siendo capaz de dejar sus cosas a un lado para así ayudarme.

De Carlos

A todas las personas que de una forma u otra, contribuyeron en mi formación profesional así como al desarrollo de este trabajo.

A toda mi familia que me dio su apoyo constante, en especial a mis padres, que tantos sacrificios han hecho para que yo pudiera llegar a este momento y confiaron en mí todo el tiempo.

DEDICATORIA

De Carlos

A mis padres, especialmente a mi mamá que depositó toda su confianza en mí, realizó grandes sacrificios para verme convertido en un profesional, convirtiendo esto en su mayor sueño.

A todos los que de una forma u otra contribuyeron a que yo pudiera realizar mi máxima aspiración, convertirme en un profesional.

De Robert

A mi mamá y a mi papá, por constituir este trabajo uno de sus más anhelados sueños, al igual que el resto de mi familia, a los cuales siempre tengo muy presentes, porque constituyen para mí, lo más apreciado que posee todo ser humano en la vida.

A todos aquellos que siempre confiaron en mí y me brindaron su apoyo en cada momento.

RESUMEN

El presente trabajo surge en el proyecto Infodrez y propone un sistema para gestionar análisis estadísticos de partidas de Ajedrez, permitiendo tanto a jugadores como entrenadores hacer un estudio del rendimiento de un jugador a partir de los reportes obtenidos. El sistema está compuesto por dos módulos; un módulo principal que permite tanto obtener un reporte de las características de las partidas de un jugador, como realizar análisis estadísticos del rendimiento de dicho jugador con respecto a alguna de las características de sus partidas en un tiempo determinado. El otro módulo es el de administración que permite a un administrador previamente autentificado gestionar los datos relativos a los jugadores, las partidas, los torneos así como de los usuarios. El sistema fue desarrollado sobre la plataforma de software libre; Lenguaje de programación PHP, corriendo sobre servidor Apache y gestor de base de datos MySql, además se utilizó la metodología RUP para la modelación. Este trabajo es el resultado de varios estudios realizados en el marco nacional e internacional y culmina proponiendo varias recomendaciones para futuras iteraciones del proyecto.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El desarrollo de software para el ajedrez a nivel mundial, es una rama de la informática con varios años de experiencia. Desde hace algún tiempo se vienen desarrollando programas que permiten el desarrollo de juegos a distancia donde los jugadores pueden estar jugando al mismo tiempo sin estar uno cerca del otro.

Desde los inicios de la Universidad de las Ciencias Informáticas se ha fomentado el desarrollo de los deportes y con estos el juego ciencia no ha sido una excepción. Es por ello que surge la necesidad de un portal de ajedrez en la UCI donde los usuarios puedan jugar online. Ya en su segundo año se creó el concurso para presentar la Web del ajedrez: siendo ganadora UCIdrez.

Paralelamente y dando cumplimiento a una resolución Rectoral se oficializó la Cátedra Honorífica de Ajedrez, Remberto A. Fernández González.

Sin embargo debido a la necesidad que surge en la UCI de crear plataformas libres como parte de la estrategia nacional de migración hacia este tipo de tecnologías, unido a que UCIdrez no se regía por estándares Web, surge la necesidad de crear un nuevo portal para el ajedrez en la UCI. Se elaboró y conceptualizó la idea de Infodrez 1.0, que comprendería las funcionalidades del antiguo sistema con algunos arreglos de diseño, funcionamiento, rendimiento superior; proponiendo dos nuevas versiones, superiores en complejidad algorítmica, funcionalidades y campos de acción; y que fuera visible desde las plataformas Windows y GNU/Linux. Esta primera versión de Infodrez garantizaría una interactividad de mayor rango al propiciar el enlace con la cátedra de ajedrez de la universidad, que el UCIdrez no poseía.

Infodrez 1.0 estaba hecho a partir del CMS e107, el cual no había sido diseñado con módulos específicos para el ajedrez. Además, tenía otras desventajas propias de los sistemas gestores de contenidos actuales, entre las que se pueden mencionar la redundancia de código y no tener una programación orientada a objeto.

En la Universidad de las Ciencias Informáticas se han realizado 6 torneos de ajedrez con el objetivo de difundir entre la comunidad universitaria la importancia del deporte ciencia, sin embargo muchas de las

características de las partidas como son las aperturas y los cierres se han perdido por no contar la cátedra Remberto Fernández con un sistema que permita guardar la información histórica. No existe ningún software que permita realizar análisis estadísticos de los juegos realizados por los estudiantes de la universidad en las diferentes competencias convocadas, así como de los realizados online a través del sitio Infodrez.

Teniendo en cuenta esta situación el **problema científico** se puede formular de la siguiente manera: ¿Cómo realizar análisis estadísticos de las partidas de los jugadores del deporte ciencia para la cátedra de Ajedrez de la UCI, Remberto Fernández, mediante un sistema de gestión?

Se definió como **objeto de estudio** Proceso automatizado de desarrollo de software de gestión estadísticos y como **campo de acción** el Proceso automatizado de desarrollo de un software para gestionar los diferentes contenidos relacionados con las estadísticas ajedrecísticas.

Como **Idea a defender** se parte de la idea de que si se desarrolla una aplicación Web basada en un gestor de Bases de Datos potente y de respuesta rápida y un intérprete eficiente es posible lograr un sistema que permita gestionar estadísticas de partidas de Ajedrez.

Se ha propuesto como **objetivo general**: Desarrollar una aplicación Web para dar soporte a análisis estadísticos de bases de datos de partidas de Ajedrez.

Como **objetivos específicos** se plantean los siguientes:

- Realizar una investigación para determinar los patrones más utilizados en el Ajedrez para la elaboración de análisis estadísticos de partidas.
- Elaborar análisis y diseño del sistema y posteriormente la implementación del mismo, el cual debe regirse por la arquitectura del Gestor de Contenidos de Ajedrez.
- Conformar la documentación del Modulo de Estadística para el desarrollo de posteriores versiones.

Para dar cumplimiento a los objetivos se plantean un grupo de **tareas**:

- Realizar una búsqueda bibliográfica detallada acerca de los patrones más utilizados en el Ajedrez para la elaboración de análisis estadísticos de partidas.
- Analizar las tendencias, herramientas y tecnologías actuales a utilizar.
- Realizar una investigación acerca de las metodologías existentes con el objetivo de seleccionar la más adecuada para nuestro sistema.
- Realizar el análisis y diseño e implementación de la aplicación Web.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

El siguiente capítulo aborda la fundamentación teórica acerca del módulo Gestor de Estadísticas ajedrecísticas del proyecto Infodrez, 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 SISTEMAS AUTOMATIZADOS SIMILARES EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN

En la actualidad hay una tendencia cada vez mayor de usar programas informáticos que analicen estadísticas en los distintos deportes. Está demostrado que el análisis estadístico ayuda a los jugadores a mejorar su rendimiento y a conocerse a sí mismos tanto como a sus contrincantes. El ajedrez no es la excepción, por lo que se pueden encontrar herramientas muy poderosas para el análisis de partidas como son ChessBase o Fritz., pero desafortunadamente no son software libre.

En la red de redes también se pueden encontrar sitios dedicados a analizar partidas de ajedrez entre los que vale destacar www.chessgames.com, el cual permite filtrar partidas de ajedrez por distintos parámetros y obtener todas las que cumplan con lo que se requiere. Además, permite visualizar una partida determinada así como descargarla. Sin embargo, este sitio tiene el inconveniente de no contar una cantidad significativa de partidas de jugadores cubanos.

En nuestro país existen muy pocos sitios dedicados al ajedrez. No existe ninguna web que realice análisis estadísticos de las partidas de nuestros jugadores en torneos regionales, es muy poca la información que puede encontrar un usuario o un ajedrecista en materia de estadística.

1.3 TENDENCIAS Y TECNOLOGÍAS ACTUALES

A continuación se describen algunas de las tendencias y tecnologías actuales posibles a utilizar para darle solución a los problemas planteados anteriormente de manera eficiente, teniendo en cuenta las necesidades existentes y el entorno donde se aplicará el sistema que se va a construir.

1.3.1 POLÍTICA DE MIGRACIÓN AL SOFTWARE LIBRE

El término «software libre» es la traducción del inglés «free software»; en inglés el término «free» tiende a confundir, ya que significa tanto «libre» como «gratuito». Por ello es necesario aclarar que la palabra «libre» en inglés, no se refiere a términos de coste económico, sino al término libertad (del inglés freedom). Este término se ha utilizado desde los años 80, y su primera definición apareció documentada en el año 1989 en el boletín GNU's Bulletin, vol.1 no. 6. En el se enumeran cuatro «libertades» que definen el software libre:

- La libertad de usar el programa con cualquier propósito.
- La libertad de estudiar cómo funciona el programa y adaptarlo a tus necesidades.
- La libertad de distribuir copias con lo que puedes ayudar a tu vecino.
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie.

Estas libertades son derechos, no obligaciones. Toda persona es libre de no usar estas libertades, pero también podría elegir el uso de todas ellas.

Es importante conocer que el software libre no excluye el uso comercial ni la distribución comercial, ya que no es software gratuito. Por ejemplo un número creciente de empresas basa su negocio completamente o al menos parcialmente en el software libre. El software libre hace legal proveer ayuda y asistencia, pero no lo hace imprescindible.

En este sentido el software libre es a menudo la mejor opción para la empresa. Ya no en términos económicos, debido a que por ejemplo la implementación, ayuda o asistencia para software libre generalmente es más barata, sino en términos de control y despersonalización.

Una gran ventaja es que la mayoría de los programas de software libre tienen una política de mejora basada en contribuciones de personas no relacionadas directamente con la creación y desarrollo del programa. Cuando se crea un módulo o herramienta con código fuente público hace posible que otros usuarios lo utilicen o mejoren.

Actualmente el gran problema de la implementación de software libre en las empresas es el miedo a que estos programas dejen de mejorarse o desarrollarse. Pero esto es una falsa creencia, ya que la filosofía del software libre impide que se llegue a este punto, en cambio, esta situación crítica puede darse con el software propietario.

Es por todo ello que se plantea la necesidad del uso del software libre en nuestro país, no solo por los beneficios económicos, sino también por la posibilidad de usar un software que es libre en su definición y no en el sentido financiero.

1.3.2 APLICACIONES WEB

Una aplicación web es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una Intranet. Las aplicaciones web son populares debido a la practicidad del navegador web como cliente ligero. La facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad. [1]

Ventajas de las aplicaciones Web

- ✓ Aportan interoperabilidad entre aplicaciones de software de sus propiedades o de las plataformas sobre las que se instalen.
- ✓ Los servicios web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- ✓ Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- ✓ Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- ✓ Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándares.

1.3.3 LENGUAJES DE PROGRAMACIÓN

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. [2]

En la actualidad existen varios lenguajes de programación que son utilizados para construir páginas web. A continuación se describen algunos de los mismos.

Lenguajes del lado del servidor

El lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, y se le envía una respuesta al cliente a través de una página. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, tratamiento de la información entre otras funciones.

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.

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. Perl esta licenciado bajo la licencia artística y la GNU General Public License.

Python

Es desarrollado como proyecto de software libre, manejado por la Python Software Foundation. Varias de sus características son semejantes a las de Lisp. [3]

Algunos consideran a Python como la "oposición leal" a Perl, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a éste mucho más limpio y elegante para programar. Sin embargo esto es más un punto de vista de sus usuarios que una realidad. [3]

Python es considerado como un lenguaje exitoso debido a su facilidad de aprendizaje, su orientación a programadores promedio y su limpieza de código. [3]

PHP

El lenguaje PHP es un pre-procesador de páginas HTML, su principal objetivo está encaminado a la construcción de páginas Web.

PHP se ha venido desarrollando con el tiempo, contiene varias bibliotecas para funciones matemáticas y de bases de datos. Actualmente se encuentra en consolidación, se esta centrando en madurar en aspectos relacionados con la integración de sus partes y ha dejado un tanto atrás la fase expansiva.

Es una gran alternativa en el trabajo de creación de portales Web dinámicos, con acceso a base de datos. Es un lenguaje de alta potencia y fácil de usar e incluye la programación orientada a objetos.

Hoy día grandes empresas usan PHP como herramienta Web, entre ellas Cisco, NTT DoCoMo, CMG, Vodafone, Motorola, Siemens, Ericsson, CBS, Unilever, Philips, BMC, NTT, Air Canadá, JAL, Lufthansa, OnVista, Lycos Europe y Deutsche Bank.

Los principales usos del PHP son los siguientes: [4]

- Programación de páginas Web dinámicas, habitualmente en combinación con el motor de base datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl o Shell scripting.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK (GIMP Tool Kit), lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Ventajas de PHP [4]

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los formularios para la Web.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Cada lenguaje de programación tiene su características propias que los distingue a continuación realizamos una comparación con el objetivo de seleccionar el más adecuado para el desarrollo de nuestra aplicación.

• Características multiplataforma: Todos estos lenguajes pueden ser utilizados en varias plataformas.
• Velocidad de ejecución: La velocidad es mayor en PHP, seguidos por PERL y Python.
• Disponibilidad de recursos: actualmente el lenguaje software libre más utilizado en la Internet es el PHP. PHP tiene una de las comunidades más grandes en Internet.

Lenguajes del lado del cliente

HTML

El lenguaje llamado HTML indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página. [5]

El lenguaje consta de etiquetas que tienen esta forma <B> o <P>. Cada etiqueta significa una cosa, por ejemplo <B> significa que se escriba en negrita (bold) o <P> significa un párrafo, <A> es un enlace, etc. Casi todas las etiquetas tienen su correspondiente etiqueta de cierre, que indica que a partir de ese punto no debe de afectar la etiqueta. Por ejemplo </B> se utiliza para indicar que se deje de escribir en negrita. Así que el HTML no es más que una serie de etiquetas que se utilizan para definir la forma o estilo que queremos aplicar a nuestro documento. <B>Esto está en negrita</B>. [5]

JAVASCRIPT

JavaScript es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario. Las sentencias escritas en javascript se encapsulan entre las etiquetas <script> y </script>.

CSS

CSS, es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias a las CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como incluir márgenes, tipos de letra, fondos, colores... Incluso podemos definir nuestros propios estilos en un archivo externo a nuestras páginas; así, si en algún momento queremos cambiar alguno de ellos, automáticamente se nos actualizarán todas las páginas vinculadas de nuestro sitio.

CSS son las siglas de Cascading Style Sheets, en español Hojas de estilo en Cascada.

Selección del lenguaje a utilizar

Del lado del servidor

Aunque PHP es un lenguaje relativamente nuevo, fue diseñado desde cero con el único fin de diseñar aplicaciones web. Esto quiere decir que las tareas más habituales en el desarrollo de estas aplicaciones, pueden hacerse con PHP de forma fácil, rápida y efectiva. Es fácil dar los primeros pasos y ver los resultados rápidamente.

PHP combina excelentemente con otras buenas herramientas, como son el servidor Apache y la base de datos MySQL (o SQL), todas ellas gratuitas y usadas en la Universidad de Ciencia Informáticas.

PHP, con todas las ventajas que presenta por su propia esencia (es el más rápido de todos los analizados, es multiplataforma, con una sintaxis familiar a un gran número de usuarios, y cuenta con gran disponibilidad de recursos en Internet). Además, es fácil de aprender. Por ello se propone su uso como lenguaje del lado del servidor.

Del lado del cliente:

Como lenguajes por la parte del cliente serán utilizados Javascript, CSS y HTML. El lenguaje principal será HTML (Lenguaje de etiquetas de hipertexto) este será utilizado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

JavaScript será utilizado como apoyo a HTML especialmente para la validación de datos ya que permite ejecutar instrucciones como respuesta a las acciones del usuario. Es un lenguaje rápido y sencillo que no requiere complicación ya que está orientado a eventos e interpretado. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) serán utilizadas para dar estilo a los documentos HTML separando el estilo de la presentación. CSS tiene grandes ventajas ya que permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo.

Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

1.3.4 SERVIDORES WEB

Un 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. [6]

Un servidor web se encarga de mantenerse a la espera de peticiones HTTP llevada a cabo por un cliente HTTP que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido que el cliente solicita.

Servidor Web Apache

Hoy en día Apache es el servidor Web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma. Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad, surgió en abril de 1996 y ya en julio del 2002 era utilizado por el 57% de los sitios Web de Internet. Tiene capacidad para servir páginas tanto de contenido estático, para lo que nos serviría sencillamente un viejo ordenador 486, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información, además de que es muy potente y altamente configurable.

El servidor Apache es un software que está estructurado en módulos, es decir, está dividido en muchas porciones de código que hacen referencia a diferentes aspectos o funcionalidades del servidor Web. Esta modularidad es intencionada ya que la configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías:

- Módulos Base: Módulo con las funciones básicas del Apache.
- Módulos Multiproceso: Son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- Módulos Adicionales: Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiprocesos para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código.

El resto de funcionalidades del servidor se consigue por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.

Lighttpd

Lighttpd es un servidor web diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Está optimizado para entornos donde la velocidad es muy importante, y por eso consume menos CPU y memoria RAM que otros servidores. Por todo lo que ofrece, lighttpd es apropiado para cualquier servidor que tenga problemas de carga. 7]

Lighttpd es software libre y se distribuye bajo la licencia BSD. Funciona en GNU/Linux y UNIX de forma oficial. Para Microsoft Windows actualmente hay una distribución conocida como Lighttpd For Windows. [7]

Fundamentación del Servidor Web escogido

Se decidió utilizar como servidor Web a Apache debido a la gran experiencia de uso de este tanto en la UCI como en el mundo y a la gran compatibilidad de este con el lenguaje de programación PHP, brinda además algunas ventajas, como son: una gran modularidad, es gratuito y multiplataforma al igual que el PHP, además de que es muy configurable.

1.3.5 AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios 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 aplicación.

AJAX es una combinación de tecnologías ya existentes:

XHTML(o HTML) y CSS, para crear una presentación basada en estándares.
DOM, para la interacción y manipulación dinámica de la presentación.
XML, XSLT y JSON, para el intercambio y la manipulación de información.
XMLHttpRequest, para el intercambio asíncrono de información.
JavaScript, para unir todas las demás tecnologías.

Como el DHTML, LAMP o SPA, AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

Desarrollar aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores.

En las aplicaciones web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

Desde su aparición, se han creado cientos de aplicaciones web basadas en AJAX. En la mayoría de casos, AJAX puede sustituir completamente a otras técnicas como Flash. Además, en el caso de las aplicaciones web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio.

1.3.6 SISTEMAS GESTORES DE BASE DE DATOS (SGBD)

Los Sistemas de Gestión de Base de Datos (SGBD) se pueden definir como el «conjunto de herramientas que suministra a todos (administrador, analistas, programadores, usuarios) los medios necesarios para describir, recuperar y manipular los datos almacenados en la BD, manteniendo la seguridad, integridad y confidencialidad de los mismos».

Entre los sistemas Gestores de Bases de Datos más usados actualmente encontramos MySQL, Oracle, PostgreSQL.

Características de los SGBD

Un sistema de gestión de bases de datos se puede definir como una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. Es un conjunto coordinado de programas, procedimientos, lenguajes, etc. que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad.

Existen dos grandes modelos de sistemas de gestión de bases de datos:

Sistemas de Gestión de Bases de Datos Relacionales (SGBDR): Las bases de datos que generan se construyen con información muy estructurada (datos) acerca de una organización o empresa determinada. Cuando un usuario realiza una consulta en una base de datos relacional, el sistema presenta como resultado la respuesta exacta a lo que se busca. A este tipo de bases de datos se les denomina bases de datos relacionales, y a los sistemas que las gestionan, Sistemas de Gestión de Bases de Datos Relacionales (SGBDR).

Sistemas de Gestión de Bases de Datos Documentales (SGBDD) o Sistemas de Recuperación de Información (SRI): Las bases de datos que generan se construyen con información no estructurada tipo texto (documentos) sobre uno o varios temas. Cuando un usuario realiza una consulta en una base de datos documental, el sistema presenta como resultado, no una respuesta exacta, sino documentos útiles para satisfacer la pregunta del usuario. A este tipo de bases de datos se les denomina bases de datos documentales, y a los sistemas que las gestionan, Sistemas de Gestión de Bases de Datos Documentales (SGBDD) o Sistemas de Recuperación de Información (SRI).

MySQL

MySQL es un SGBD multiplataforma, es un software de fuente abierta lo que significa que cualquier persona puede usarlo y modificarlo para satisfacer sus necesidades, puede ser utilizado gratuitamente. Se encuentra bajo la licencia GPL.

Por su sencillez y sus características es usado por muchas personas, consume muy pocos recursos, se usa tanto en aplicaciones sencillas como complejas. Es utilizado en aplicaciones Web como Drupal, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla, además sus conexiones son muy seguras.

PostgreSQL

Es un magnífico gestor de bases de datos, es multiplataforma. Se encuentra bajo la licencia BSD. Permite una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Sirve de soporte a los lenguajes más populares como PHP, C, C++, Java, Python, Ruby, etc., y al protocolo de comunicación encriptado por SSL. El número de base de datos que puede contener es ilimitado.

Fundamentación de la selección del SGBD.

A partir de las características que ofrecen los distintos SGBD se decidió escoger para la realización de las BD de la aplicación a MySQL como gestor por los buenos resultados que en la práctica se han obtenido complementado con PHP y el servidor Web Apache. Incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, permite administrar el sistema y protegerlo, así como, hacer volcados de datos. Puede desarrollar sus propias aplicaciones de base de datos en la mayor parte de los lenguajes de programación utilizados en la actualidad y ejecutarlos en casi todos los sistemas operativos. 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 manejar multitud de tipos para columnas, así como registros de longitud fija o variable.

Por otra parte, permite contar con la disponibilidad de otras plataformas y sistemas. 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.

1.3.7 METODOLOGÍAS DE DESARROLLO DE SOFTWARE.

Proceso de desarrollo de software.

Un proceso define «quién» está haciendo «qué», «cuándo» y «cómo» para alcanzar un de terminado objetivo. Un Proceso de Desarrollo de Software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

El proceso de desarrollo de software tiene la misión de transformar los requerimientos del usuario en un producto de software; de manera que los integrantes del equipo y todo aquel que pueda estar interesado en el producto final, tenga la misma visión y no ocurra cuando no se aplica un proceso de desarrollo.

Por lo tanto, las piedras angulares del proceso de desarrollo del software son: el proyecto, las personas y el producto; siendo las características del cliente, el entorno de desarrollo y las condiciones del negocio, elementos que influyen en el proceso. Existe una estrecha relación entre personas, proyecto, producto y proceso. Estos términos son conocidos como las cuatro «P» en el desarrollo de software.

El resultado final de un proyecto software es un producto, donde intervienen personas a través de un proceso de desarrollo de software que guía los esfuerzos de las personas implicadas en el proyecto.

Metodologías.

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Una metodología es un proceso. [8]

No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable. [8]

En la actualidad existen varias metodologías OO (Orientada a Objetos) basadas en UML: Rational Unified Process (RUP), Extreme Programing (XP), OPEN, MÉTRICA 3, y otras más. [8]

Sabiendo que (UML) Lenguaje Unificado de Modelado es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software y que este está compuesto por diversos elementos gráficos que se combinan para conformar diagramas.

Este lenguaje que permite la modelación de sistemas con tecnología orientada a objetos y describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

Ya planteado esto, se hará un estudio de algunas de las diferentes metodologías planteadas.

El Proceso Unificado de Desarrollo (RUP)

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

Es una de las metodologías más generales y más usadas de las que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto. Constituye además, una propuesta de proceso para el desarrollo de software orientado a objeto, utilizando UML (del inglés Unified Model Language), para describir todo el proceso, basándose en componentes. Este lenguaje es estándar, con él se puede visualizar, especificar, construir y documentar los artefactos de un sistema.

Las principales características de esta metodología son:

- Centrado en los Modelos: Los diagramas son un vehículo de comunicación más expresivo que las descripciones en lenguaje natural. Se trata de minimizar el uso de descripciones y especificaciones textuales del sistema.
- Guiado por los casos de uso: Los casos de uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- Centrado en la arquitectura: Los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar.

- Iterativo e incremental: Durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

El RUP incluye cuatro etapas importantes que son: Inicio, Elaboración, Construcción y Transición, cada una de ellas compuesta de una o varias iteraciones. Estas etapas revelan que para producir una versión del producto en desarrollo se emplean todas las actividades de ingeniería pero con diferente énfasis; en las primeras versiones se hace más énfasis en el modelado del negocio, requisitos, análisis y diseño; mientras en las posteriores el énfasis recae sobre las actividades de implementación, pruebas y despliegue. Además contempla flujos de trabajo de soporte que involucran actividades de planificación de recursos humanos tecnológicos y financieros.

Como RUP es un proceso, en su modelación se define como sus principales elementos:

- Trabajadores (¿Quien?): Definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Actividades (¿Cómo?): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Artefactos (¿Qué?): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Flujo de actividades (¿Cuándo?): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.
- En él se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales.
- Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

Flujos de trabajo:

- Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba (Testeo): Busca los defectos a los largo del ciclo de vida.
- Instalación: Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Extreme Programing (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo, equipo pequeño y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. [9]

Características de XP, la metodología se basa en: [9]

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos

estándares, siendo más flexible al cambio.

- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

¿Qué es lo que propone XP? [9]

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.
- Derechos del Cliente.
- Decidir que se implementa.
- Saber el estado real y el progreso del proyecto.
- Añadir, cambiar o quitar requerimientos en cualquier momento.
- Obtener lo máximo de cada semana de trabajo.
- Obtener un sistema funcionando cada 3 o 4 meses.

Entre los derechos del Desarrollador encontramos que este es capaz de decidir como se implementan los procesos y tiene la libertad de crear el sistema con la mejor calidad posible. Le puede pedir al cliente en cualquier momento aclaraciones de los requerimientos y estima el esfuerzo para implementar el sistema. Cambiar los requerimientos en base a nuevos descubrimientos siempre que estos aparezcan en el ciclo de vida del software. [9]

Lo fundamental en este tipo de metodología es: [9]

- La comunicación, entre los usuarios y los desarrolladores
- La simplicidad, al desarrollar y codificar los módulos del sistema
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Fundamentación de la selección de la metodología de desarrollo de software.

El software es un componente esencial de toda actividad basado en el uso de la informática, es por ello que la calidad en su desarrollo y mantenimiento resulta hoy en día uno de los principales objetivos estratégicos de las organizaciones. En los últimos años se han publicado diversos estudios en los que se presentan los principios que se deben seguir para mejorar los procesos de software, y de esta forma evitar las grandes catástrofes que conllevan al fracaso de un gran número de proyectos.

La metodología para el desarrollo del software surgió de la construcción del marco conceptual que integra el enfoque Constructivista, el Pensamiento Sistémico junto con la Dinámica de Sistemas. Ésta constituye un conjunto de procedimientos, reglas, herramientas y aspectos de formación para los desarrolladores de aplicaciones Informáticas.

Para controlar, y concebir la propuesta de este sistema, se decide utilizar como metodología el Proceso Unificado de Modelado (RUP), por todas las ventajas de organización que brinda y debido a que goza de un grupo de características y facilidades, que hacen mas dinámico el desarrollo del trabajo.

1.3.8 LENGUAJE UNIFICADO DE MODELADO (UML)

UML (Unified Modeling Language) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software. [10]

Sus creadores pretendieron con este lenguaje, unificar las experiencias acumuladas sobre técnicas de modelado e incorporar las mejores prácticas en un acercamiento estándar.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. Un modelo UML indica que es lo que supuestamente hará el sistema pero no como lo hará. [11]

El UML permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas que estén involucradas en el proceso de desarrollo de los sistemas, esto se lleva a cabo mediante un conjunto de símbolos y diagramas. [11]

Herramientas CASE de Modelado con UML

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. [12]

Visual Paradigm

El Visual Paradigm es una poderosa herramienta CASE de modelación visual. Utiliza UML para el modelado, permitiendo crear tipos diferentes de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar. Genera código para varios lenguajes. Es una herramienta que puede ser utilizada en la creación de software libre, es nombrada por muchas bibliografías como la herramienta CASE por excelencia del software libre, siendo esta una de las características que dio lugar a que se seleccionara para el desarrollo del sistema, sin dejar de mencionar que el “Visual Paradigm, además posee:

- ✓ 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 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.

Por otra parte, posibilita la representación gráfica de los diagramas permitiendo ver el sistema desde diferentes perspectivas, como el de componentes, despliegue, secuencia, casos de uso, clase, actividad, entre otros. Además, se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos, también, permite ver las relaciones entre los componentes del diseño y mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico.

Tiene disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community. Facilita licencias especiales para fines académicos.

1.3.9 HERRAMIENTAS DE DESARROLLO

Como herramientas de desarrollo de apoyo a la programación por su facilidad de uso son muy utilizados los entornos de desarrollo integrado. Un entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

Zend Studio

Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, como no, en lenguaje PHP. 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. [13]

El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y

rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más. [13]

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración.

Eclipse

Eclipse es una plataforma de software de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent Azureus.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Fundamentación de la herramienta de desarrollo escogida

Se seleccionó como herramienta de desarrollo del software el Zend Studio ya que este incluye todos los componentes necesarios durante el ciclo de vida de una aplicación en PHP. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Zend Studio nos permite agiliza el desarrollo web y permite simplificar proyectos complejos.

1.3.10 FRAMEWORKS Y LIBRERÍAS

En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Los Frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Symfony

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Zend Framework

Es un framework código abierto para aplicaciones web orientado a objetos desarrollado en PHP 5 y registrado bajo la nueva licencia BSD. Zend framework (a menudo denominado ZF) se desarrolla con el objetivo de simplificar el desarrollo web y al mismo tiempo promover las mejores prácticas en la comunidad de desarrolladores de PHP.

La arquitectura de ZF permite a los desarrolladores reutilizar los componentes en sus aplicaciones cuándo y dónde tenga sentido sin necesidad de otros componentes de ZF.

JpGRAPH

Es una librería que incluye una serie de clases -código orientado a objetos- que sirven para crear imágenes con todo tipo de gráficas, dinámicamente desde páginas PHP. [14]

El sistema está muy depurado y soporta multitud de funcionalidades, por lo que seguramente encontraremos solución a casi cualquier necesidad en el ámbito de creación de gráficas. Además, la mayoría de las configuraciones de las gráficas vienen con opciones por defecto, así que resulta bastante sencillo obtener resultados rápidamente. [14]

Algunas de las características del sistema son: [14]

- ✓ Reducido peso en bytes de las imágenes resultado. Habitualmente unas pocas KB.
- ✓ Soporte a las librerías GD1 o GD2.
- ✓ Uso de la Interpolación matemática para obtener curvas a partir unos pocos valores.
- ✓ Diversos tipos de gráficas 2D o 3D, como de puntos, líneas, tartas, barras, cajas...

- ✓ Escalas flexibles tanto en el eje X como el Y, que se ajustan al juego de datos que se tenga que representar.
- ✓ Soporte para generar gráficas con varios juegos de valores a la vez.
- ✓ Configurable con distintos tipos de colores, leyendas, tipografías, imágenes de fondo, etc.

Greybox

GreyBox, es una pequeña utilidad Javascript que mediante ajax nos permite cargar en una ventana emergente diferentes enlaces, entre ellos una web, imágenes, y otras cosas.

Algunas de sus ventajas son las siguientes:

- ✓ No entra en conflicto con los navegadores que bloquean los pop up.
- ✓ Es fácil de instalar y de utilizar
- ✓ Su estilo puede ser modificado fácilmente a través de CSS.

Fundamentación del framework y las librerías seleccionados

Se escogió como framework a utilizar el Symfony, debido a la calidad de su código fuente y a la gran cantidad de documentación disponible, dos ventajas muy importantes sobre los otros frameworks estudiados, además Symfony es una herramienta ideal para el desarrollo de aplicaciones rápidas y se adapta fácilmente a los cambios que puedan surgir durante el desarrollo del software.

Se escogió JpGraph para la construcción de gráficos estadísticos, debido a que es una herramienta muy potente a través de la cual se pueden generar gráficos de manera sencilla y rápida. Además de esto incluye una documentación muy completa con gran cantidad de ejemplos. Se decidió utilizar además la herramienta Greybox, la cual puede ser utilizada para mostrar a través de ventanas emergentes imágenes, animaciones o páginas sin tener que preocuparse de los bloqueadores de pop up que traen por defecto algunos navegadores.

1.3.11 ARQUITECTURA

Una Arquitectura 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. [15]

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 de computación. [15]

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. [15]
Cliente-servidor. Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones. [15]

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. [15]

1.- Capa de presentación: es la que ve el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz grafica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario.

2.- Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

3.- Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

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.

Modelo Vista Controlador (MVC): Es un patrón de diseño 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. [16]

- 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.

El ciclo de vida de MVC es normalmente presentado con los tres componentes presentados anteriormente y el cliente normalmente conocido como usuario.

El primer paso en el ciclo de vida empieza cuando el usuario hace una solicitud al controlador con información sobre lo que el usuario desea realizar. Entonces el controlador decide quien debe delegar la tarea y es ahí donde el modelo empieza su trabajo. En esta etapa, el Modelo se encarga de realizar operaciones sobre la información que maneja para cumplir lo que le solicita el controlador. Una vez que termina su labor, le regresa al Controlador la información resultante de sus operaciones el cual redirige a la Vista. La Vista se encarga de transformar los datos en información visualmente entendible para el usuario. Finalmente, la representación gráfica es transmitida de regreso al Controlador y este se encarga de transmitírsela al usuario. El ciclo entero puede comenzar de nuevo si el usuario así lo requiere.

Fundamentación de la arquitectura escogida

Se seleccionó como patrón de diseño de arquitectura Modelo Vista Controlador ya que este está diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. El MVC es el ideal para trabajar en aplicaciones con una gran cantidad de datos y transacciones complejas donde se requiere una buena separación de los

conceptos para que el desarrollo este estructurado de una mejor manera, facilitando la programación en diferentes capas de forma paralela e independiente.

Modelo Vista Controlador ofrece varias ventajas entre estas están las siguientes:

La separación del Modelo y la Vista, es decir, la separación de los datos y la representación visual de los mismos.

- Es mucho más sencillo agregar múltiples representaciones de los mismos datos o información.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento en caso de errores
- Ofrece maneras más sencillas de probar el correcto funcionamiento del sistema.

1.4 CONCLUSIONES

En este capítulo se hizo una investigación detallada de las tecnologías informáticas existentes y a partir del estudio realizado fueron seleccionadas las herramientas y tecnologías a utilizar para la conformación de un sistema capaz de cumplir con las expectativas esperadas. Se seleccionó RUP como metodología para el modelado del sistema con UML como lenguaje de representación visual. Se escogieron PHP, MYSQL y Apache como herramientas básicas para el desarrollo, debido a que la unión de estas tecnologías ha demostrado ofrecer eficientes resultados.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 INTRODUCCIÓN

En el presente capitulo se identifican cuales son los requisitos funcionales y los no funcionales del sistema, se obtienen y describen los casos de uso que guiaran la solución propuesta. Ha sido imprescindible la utilización de la herramienta Case Visual Paradigm que asiste al desarrollo del software para una mayor calidad del mismo.

Presentamos un modelo de dominio como alternativa al modelo del negocio debido a la poca estructuración de los procesos que describen el negocio. Su objetivo es el de contribuir a la comprensión del contexto del sistema, y por lo tanto de los requisitos funcionales que se desprenden de este contexto.

2.2 DESCRIPCIÓN GENERAL DE LA PROPUESTA DE SISTEMA

Para dar cumplimiento a los objetivos previamente planteados, el sistema propuesto constará de dos módulos: un módulo principal y uno de administración; tres roles: usuario anónimo, usuario registrado y administrador.

A través de la base de datos se controlará la información relativa a cada usuario. Una vez que alguien que nunca ha entrado al sistema lo visita, se le da la opción de registrarse. El usuario anónimo solo puede acceder a la página principal para registrarse o autentificarse.

Cuando un usuario registrado se haya autentificado, puede acceder a todos los servicios brindados por el módulo principal. En este módulo se le da la posibilidad de analizar el rendimiento de un jugador en dependencia de algunas de las características de sus partidas en un periodo determinado. También el usuario podrá realizar una búsqueda filtrada de las partidas de un jugador de acuerdo a determinados parámetros.

El módulo de administración solo puede ser accedido por un administrador correctamente autentificado, es aquí donde se realiza la gestión de jugadores, partidas, torneos y usuarios.

2.3 MODELO DEL DOMINIO

2.3.1 DESCRIPCIÓN DEL MODELO DE DOMINIO

Haciendo un análisis exhaustivo del problema en cuestión se llega a la conclusión de que el negocio del presente trabajo tiene un bajo nivel de estructuración, no se definen concretamente los procesos del mismo, de ahí que se decide dar un nuevo enfoque a todo el proceso. Para ello se utiliza un modelo del dominio, ya que permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Este modelo contribuye posteriormente a identificar algunas clases que se utilizarán en el sistema, lo que ayuda a los clientes y desarrolladores a utilizar un vocabulario común que permita entender el contexto en que se emplaza el sistema.

Para esto es necesario tener un conocimiento sólido del funcionamiento del objeto de estudio para hacer una correcta captura de los requisitos y construir un sistema correcto. Ahora bien, para un mayor entendimiento del diagrama de clases del dominio se dejan claros todos los conceptos que intervienen en el mismo:

Usuario: Persona que interactúa con nuestra entidad.

Reporte: Contiene información solicitada por una persona.

Jugador: Jugador de Ajedrez acerca del cual se va a solicitar información.

Partida: Partida de Ajedrez.

Torneo: Evento donde participan dos o más jugadores.

2.3.2 DIAGRAMA DEL MODELO DEL DOMINIO

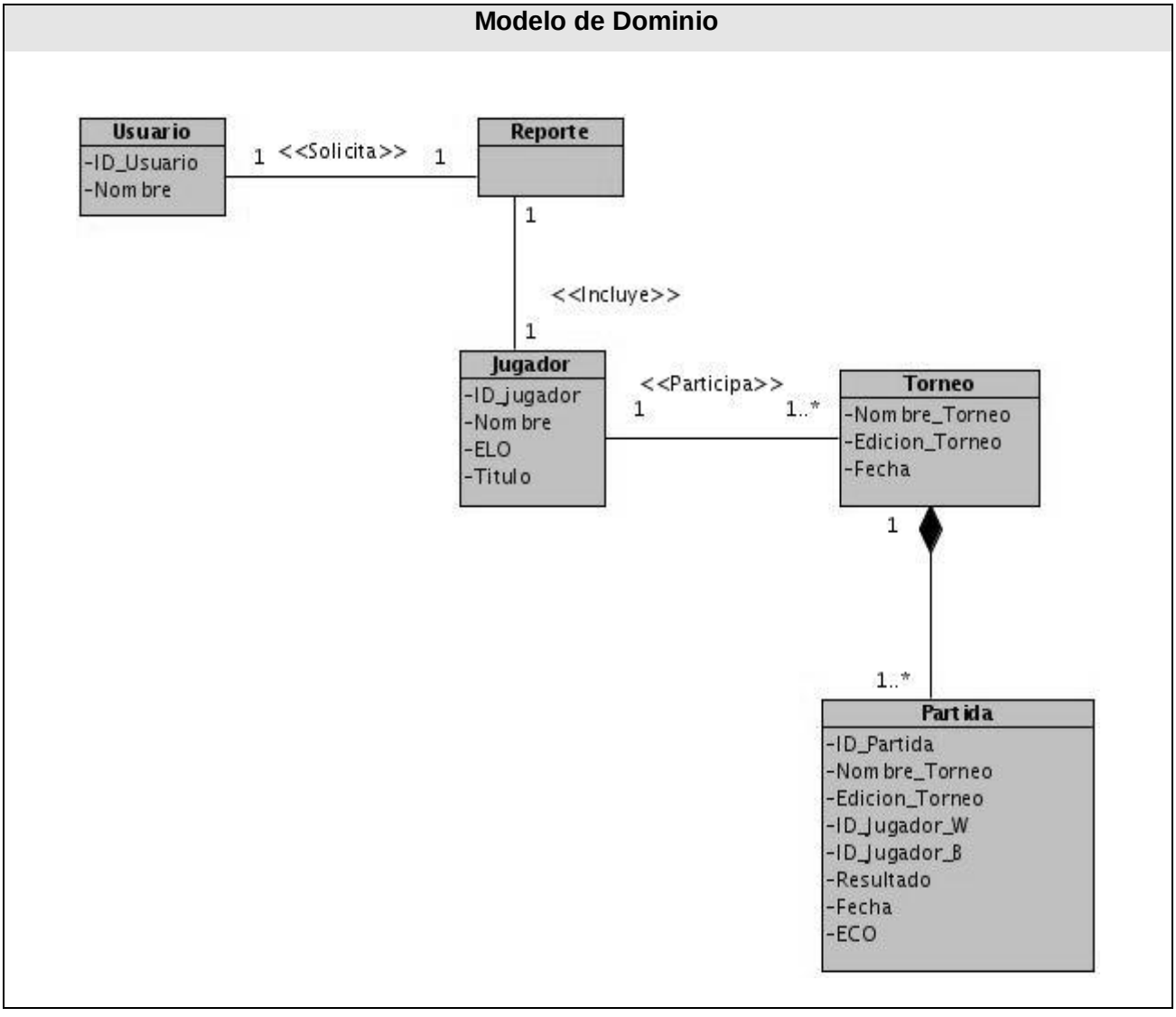

Figura 1 Modelo de Dominio

2.4 DEFINICIÓN DE LOS REQUISITOS FUNCIONALES

Los requerimientos o requisitos funcionales, son capacidades o condiciones que el sistema debe cumplir, es decir, define que es lo que el sistema debe hacer, para lo cual se identifican las

funcionalidades requeridas. Con estos se pretende determinar de manera mas clara y concisa lo que debe hacer el sistema siguiendo un enfoque funcional.

A continuación se presentan las funcionalidades que debe cumplir el sistema en cuestión:

RF1 - Solicitar reporte de las características de las partidas de un jugador
RF2 -Mostrar reporte de las características de las partidas de un jugador

RF3- Mostrar reporte de las características de un jugador

RF4- Mostrar gráfico de rendimiento

RF5- Mostrar comportamiento de un jugador por resultado

RF6- Mostrar comportamiento de un jugador por ECO

RF7- Mostrar comportamiento de un jugador por el color de las piezas

RF8- Gestionar Jugador

RF9- Permitir autenticarse

RF10- Gestionar Torneo

RF11- Gestionar Partida

RF12- Gestionar Usuario

RF11- Permitir registrarse

2.5 DEFINICIÓN DE LOS REQUISITOS NO FUNCIONALES

Los requerimientos o requisitos no funcionales son propiedades o cualidades que el producto debe tener, que posibilitan que este sea mas atractivo, usable, rápido, confiable entre otras características. También, puede ser alguna restricción que este debe tener para cumplir una determinada funcionalidad.

A continuación se presentan los Requisitos no Funcionales:

Usabilidad

RNF 1. El sistema podrá ser usado por cualquier persona que acceda a él que tenga algún conocimiento básico de computación y trabajo en la Web.

RNF 2. Rápido acceso de la información en tiempos cortos.

Seguridad

RNF 3. Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que este activo.

RNF 4. Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Eficiencia

RNF 4. El sistema deberá ser capaz de gestionar toda la información y dar respuesta a las solicitudes lo más rápido posible.

RNF 5. Debe ser eficiente a la hora de gestionar las solicitudes logrando que sin mucha navegación por el sitio se obtenga los resultados deseados.

Soporte

RNF 6. El sistema debe ser de fácil instalación y configuración, con vista a poder darle un mantenimiento asequible en caso de fallos.

Restricciones de diseño e implementación

RNF 7. Para la programación en PHP se recomienda el editor Zend Studio.

RNF 8. Se recomienda el uso de la arquitectura MVC.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF 9. Documentación de ayuda para uso del sistema, la cual estará asequible desde cualquier parte del mismo para satisfacer cualquier duda que el usuario presente con el manejo y uso de la aplicación.

Interfaces de usuario

RNF 10. El diseño de la interfaz debe ser sencillo y claro. Debe contener elementos visibles que identifiquen cada una de sus acciones.

Interfaces Hardware

RNF 11. El Servidor debe tener como mínimo las siguientes características de hardware: Procesador Pentium II 450 MHz o superior, 128 Mb de memoria RAM (incluye la utilizada por el SO) y 2Gb de capacidad en disco duro.

Interfaces Software

RNF 12. Este sitio correrá en los sistemas operativos siguientes: Linux y Windows NT/2000/XP. Además se utiliza la herramienta de modelado Visual Paradigm y el lenguaje de programación PHP.

Requisitos Legales, de Derecho de Autor y otros.

RNF 13. 11.1 Reconocido y autorizado por instancias superiores tales como la directiva de la UCI.

2.6 ACTORES DEL SISTEMA

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

A continuación se detallan los actores del sistema:

Tabla 1 Actores del sistema

Actor	Descripción
Usuario	Este es el usuario que interactúa con el sistema ya sea solicitando algún reporte o algún análisis estadístico en particular.
Administrador	Es el encargado de la gestión de la información.

2.7 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 otros beneficios.

A continuación se muestra la figura 2 correspondiente al diagrama de casos de uso del sistema:

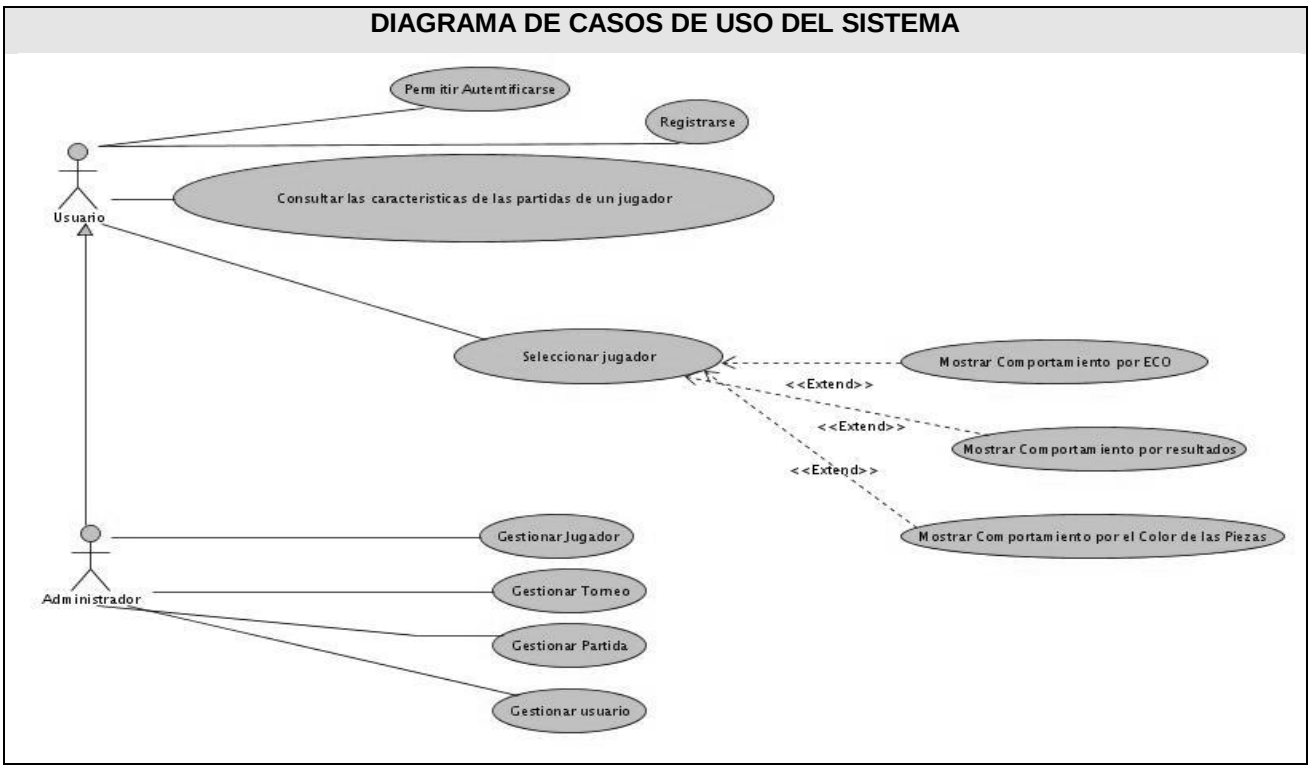

Figura 2 Diagrama de Casos de Uso del Sistema

2.7.1 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).

2.7.1.1 Descripción textual del Caso de Uso de sistema: Consultar las características de las partidas de un jugador

Tabla 2 Descripción del CUS: Consultar las características de las partidas de un jugador

Caso de Uso:	Consultar las características de las partidas de un jugador
Actores:	Usuario (inicia)
Resumen:	El CUS se inicia cuando el usuario selecciona las características de las partidas de un jugador que desea conocer, el sistema le muestra lo solicitado o no y finaliza el CUS.
Precondiciones:	El usuario debe estar previamente autentificado en el sistema.
Referencias	RF1 RF2
Prioridad	Crítico
Flujo Normal de Eventos	

Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona al jugador y algunas características que son opcionales como son: - Jugador contrario - Color de las piezas del jugador - Resultado - ECO - Fecha - Torneo Después de esto presiona el botón “Aceptar”	1.1- El sistema busca la información solicitada por el usuario.
	1.2- El sistema muestra las partidas que cumplen las características seleccionadas por el usuario y muestra los siguientes datos: - Jugador-Oponente - Resultado de la Partida - Tiempo

	- ECO - Fecha
Flujos Alternos **4.a-El sistema no encuentra ninguna partida**	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra que no se encontró ninguna partida y da la opción de ir a la página principal de consultar partidas con el botón que esta en el menú llamado “Consultar partidas”.
2-El usuario presiona el botón “Principal”.	
Poscondiciones	Las partidas realizadas por el jugador seleccionado son mostradas según las características seleccionadas.

2.7.1.2 Descripción textual del Caso de Uso de sistema: Seleccionar jugador

Tabla 3 Descripción del CUS: Seleccionar jugador

Caso de Uso:	Seleccionar jugador
Actores:	Usuario
Resumen:	El CUS se inicia cuando el usuario selecciona un jugador de entre un conjunto de jugadores, el sistema le muestra las características de este jugador y finaliza el CUS.
Precondiciones:	El usuario debe estar previamente autentificado en el sistema.
Referencias	RF3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 Solicita consultar los jugadores existentes para seleccionar uno.	1.1 Muestra un listado con todos los jugadores.
2 Selecciona el jugador deseado de entre los mostrados y da clic en el botón “Mostrar”.	2.1 El sistema muestra las características del jugador seleccionado por el usuario las cuales

	son : -Nombre -Sexo -Fecha Nacimiento -ELO -Titulo -Id Fide -Federación
3 Si desea que le sea mostrado el comportamiento del jugador con respecto a un ECO determinado selecciona la opción : “Mostrar Comportamiento por ECO”	4.1 Para esto va al CUS extendido “Mostrar Comportamiento por ECO”
4 Si desea que le sea mostrado el comportamiento de un jugador con respecto a sus resultados selecciona la opción: “Mostrar Comportamiento por resultados”.	5.1 Para esto va al CUS extendido “Mostrar Comportamiento por resultados”
5 Si desea que le sea mostrado el comportamiento de un jugador con respecto al color de las piezas selecciona la opción: “Mostrar Comportamiento por el Color de las Piezas”	6.1 Para esto va al CUS extendido “Mostrar Comportamiento por el Color de las Piezas”
Poscondiciones	Se muestran las principales características del jugador seleccionado por el usuario.

2.7.1.3 Descripción textual del Caso de Uso de sistema: Mostrar comportamiento por ECO

Tabla 4 Descripción del CUS: Mostrar comportamiento por ECO

Caso de Uso:	Mostrar comportamiento por ECO
Actores:	Usuario
Resumen:	El CUS extendido se inicia cuando el usuario después de haber seleccionado un jugador solicita que le sea mostrado su comportamiento con un ECO

	determinado en un periodo de tiempo, se le muestra un gráfico de rendimiento correspondiente a los parámetros introducidos y culmina el CUS.	
Precondiciones:	El usuario debe estar previamente autentificado en el sistema. Tiene que haberse ejecutado el CUS “Seleccionar jugador”	
Referencias	RF6 RF4	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
		1.1 Muestra al usuario una interfaz dándole la posibilidad de seleccionar el período de tiempo en que se quiere conocer el comportamiento con un ECO determinado y el torneo en que se desea consultar.
2 Selecciona el período de tiempo y el torneo dando clic en el botón “Mostrar”.		2.1 Muestra un gráfico de rendimiento con el comportamiento del jugador con respecto a un determinado ECO.
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
2 Si se desea cancelar los resultados seleccionados doy clic en el botón “Cancelar”.		2.1 El sistema muestra en el formulario los datos iniciales.
Poscondiciones	Es mostrado correctamente el gráfico de rendimiento con el comportamiento de un jugador con respecto a un ECO determinado.	

2.7.1.4 Descripción textual de Casos de Uso de sistema: Mostrar Comportamiento por resultados

Tabla 5 Descripción de CUS: Mostrar Comportamiento por resultados

Caso de Uso:	Mostrar Comportamiento por resultados
Actores:	Usuario

Resumen:	El CUS extendido se inicia cuando el usuario después de haber seleccionado un jugador solicita que le sea mostrado su comportamiento con un color de piezas determinado en un periodo de tiempo, se le muestra un gráfico de rendimiento correspondiente a los parámetros introducidos y culmina el CUS.	
Precondiciones:	El usuario debe estar previamente autentificado en el sistema. Tiene que haberse ejecutado el CUS “Seleccionar jugador”	
Referencias	RF5 RF4	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
		1.1 Muestra al usuario una interfaz dándole la posibilidad de seleccionar un período de tiempo en el que se quiere conocer el comportamiento por los resultados y el torneo
2 Selecciona el período de tiempo y el torneo dando clic en el botón “Mostrar”.		2.1 Muestra un gráfico de rendimiento el que enseña el % de victorias, el % de derrotas y el % de tablas en ese periodo de tiempo.
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
2 Si se desea cancelar los resultados seleccionados doy clic en el botón “Cancelar”.		2.1 El sistema muestra en el formulario los datos iniciales.
Poscondiciones	Es mostrado correctamente el gráfico de rendimiento con el comportamiento de un jugador con respecto a sus resultados.	

2.7.1.5 Descripción textual de Casos de Uso de sistema: Mostrar Comportamiento por el Color de las Piezas

Tabla 6 Descripción de CUS: Mostrar Comportamiento por el Color de las Piezas

Caso de Uso:	Mostrar Comportamiento por el Color de las Piezas	
Actores:	Usuario	
Resumen:	El CUS extendido se inicia cuando el usuario después de haber seleccionado un jugador solicita que le sea mostrado su comportamiento por resultados en un periodo de tiempo, se le muestra un gráfico de rendimiento correspondiente a los parámetros introducidos y culmina el CUS.	
Precondiciones:	El usuario debe estar previamente autentificado en el sistema. Tiene que haberse ejecutado el CUS “Seleccionar jugador”	
Referencias	RF7 RF4	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
		1.1 Muestra al usuario una interfaz dándole la posibilidad de seleccionar un período de tiempo en el que se quiere conocer el comportamiento por el color de las piezas además brinda la opción de seleccionar el color de las piezas y el torneo.
2 Selecciona el período de tiempo, el color de las piezas y el torneo dando clic en el botón “Mostrar”.		2.1 Muestra un grafico de rendimiento de un jugador con respecto a color de las piezas determinado.
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
2 Si se desea cancelar los resultados seleccionados doy clic en el botón “Cancelar”.		2.1 El sistema muestra en el formulario los datos iniciales.
Poscondiciones	Es mostrado correctamente el gráfico de rendimiento con el comportamiento de	

	un jugador con respecto a un color de las piezas determinado

2.7.1.6 Descripción textual del Caso de Uso de sistema: Gestionar Jugador

Tabla 7 Descripción del CUS: Gestionar Jugador

Caso de Uso:	Gestionar Jugador
Actores:	Administrador
Resumen:	El Administrador del sistema tendrá la posibilidad de insertar, modificar, eliminar o mostrar la información referente a los jugadores.
Precondiciones:	El administrador debe estar previamente autentificado en el sistema.
Referencias	RF8
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El Administrador del sistema solicita la interfaz correspondiente a la gestión de información para insertar, editar o eliminar la información correspondiente a un jugador.	1.1 El sistema muestra un listado de jugadores, permitiendo las siguientes opciones: Nuevo. Editar. Eliminar.
2 El Administrador del sistema da clic en el botón “Nuevo”.	2.1 Para esto va al escenario “Insertar Jugador”.
3 El Administrador del sistema da clic en el botón “Editar”, en el jugador que deseo modificar.	3.1 Para esto va al escenario “Modificar Jugador”.
4 El Administrador del sistema da clic en el botón “Eliminar”, en el jugador que deseo eliminar.	4.1 Para esto va al escenario “Eliminar Jugador”. Terminando el CUS.
Flujo Normal de Eventos **Sección “Insertar Jugador”**	
Acción del Actor	**Respuesta del Sistema**
	2.1 El sistema muestra los campos a llenar para

	adicionar un nuevo jugador, los mismos son: Id de jugador. Nombre. Sexo. Fecha de Nacimiento. Elo. Id fide. Federación. Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado.
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón "Guardar".	3.1 El sistema adiciona al jugador y muestra un mensaje "Tus modificaciones han sido guardadas", permitiendo además modificar los datos insertados.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón "Guardar y adicionar nuevo".	3.1 El sistema muestra un mensaje "Tus modificaciones han sido guardadas", dando la opción de insertar un nuevo jugador o ir al listado de los jugadores.
3 El Administrador del sistema da clic en el botón "Listado".	3.1 El sistema muestra el listado de los jugadores.
Poscondiciones	El nuevo jugador fue adicionado exitosamente.
Flujo Normal de Eventos **Sección "Eliminar Jugador"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra un mensaje "Estas

	seguro".
2 El Administrador del sistema acepta la operación.	2.1 El sistema elimina el jugador seleccionado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 El Administrador del sistema cancela la operación.	2.1 El sistema cancela la operación y se mantiene en la misma página mostrando el listado de los jugadores.
Poscondiciones	El jugador seleccionado ha sido eliminado exitosamente.
Flujo Normal de Eventos **Sección "Modificar Jugador"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra los campos a llenar para adicionar un nuevo jugador los mismos son: Id de jugador. Nombre. Sexo. Fecha de Nacimiento. Elo. Id fide. Federación. Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado. Eliminar.
2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón "Guardar".	2.1 El sistema guarda los cambios y muestra un mensaje "Tus modificaciones han sido guardadas".
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón “Guardar y adicionar nuevo”.	2.1 El sistema muestra un mensaje “Tus modificaciones han sido guardadas”, dando la opción de insertar un nuevo jugador o ir al listado de los jugadores.
2 El Administrador del sistema da clic en el botón “Listado”.	2.1 El sistema muestra el listado de los jugadores.
Poscondiciones	Los datos del jugador han sido modificados.

2.7.1.7 Descripción textual del Caso de Uso de sistema: Gestionar Torneo

Tabla 8 Descripción del CUS: Gestionar Torneo

Caso de Uso:	Gestionar Torneo
Actores:	Administrador
Resumen:	El Administrador del sistema tendrá la posibilidad de insertar, modificar o eliminar la información referente a los torneos.
Precondiciones:	El administrador debe estar previamente autentificado en el sistema.
Referencias	RF10
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El Administrador del sistema solicita la interfaz correspondiente a la gestión de información para insertar, modificar o eliminar la información correspondiente a un torneo	1.1 El sistema muestra un listado de torneos, permitiendo las siguientes opciones: Nuevo. Editar. Eliminar.
2 El Administrador del sistema da clic en el botón “Nuevo”.	2.1 Para esto va al escenario “Insertar Torneo”.

3 El Administrador del sistema da clic en el botón "Editar", en el jugador que deseo modificar.	3.1 Para esto va al escenario "Modificar Torneo".
4 El Administrador del sistema da clic en el botón "Eliminar", en el torneo que deseo eliminar.	4.1 Para esto va al escenario "Eliminar Jugador". Terminando el CUS.
Flujo Normal de Eventos **Sección "Insertar torneo"**	
Acción del Actor	**Respuesta del Sistema**
	2.1 El sistema muestra los campos a llenar para adicionar un nuevo jugador, los mismos son: Nombre. Fecha de Inicio. Fecha de Fin. Bases. Id Federación. Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado.
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón "Guardar".	3.1 El sistema adiciona al torneo y muestra un mensaje "Tus modificaciones han sido guardadas", permitiendo además modificar los datos insertados.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón "Guardar y adicionar nuevo".	3.1 El sistema muestra un mensaje "Tus modificaciones han sido guardadas", dando la opción de insertar un nuevo torneo o ir al listado de los torneos.
3 El Administrador del sistema da clic en el botón	3.1 El sistema muestra el listado de los torneos.

"Listado".	
Poscondiciones	El torneo ha sido insertado exitosamente.
Flujo Normal de Eventos **Sección "Eliminar Torneo"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra un mensaje "Estas seguro".
2 El Administrador del sistema acepta la operación.	2.1 El sistema elimina el torneo seleccionado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 El Administrador del sistema cancela la operación.	2.1 El sistema cancela la operación y se mantiene en la misma página mostrando el listado de los torneos.
Poscondiciones	El torneo ha sido eliminado exitosamente.
Flujo Normal de Eventos **Sección "Modificar Torneo"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra los campos a llenar para adicionar un nuevo torneo, los mismos son: Nombre. Fecha de Inicio. Fecha de Fin. Bases. Id Federación. Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado.

2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón “Guardar”.	2.1 El sistema guarda los cambios y muestra un mensaje “Tus modificaciones han sido guardadas”.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón “Guardar y adicionar nuevo”.	2.1 El sistema muestra un mensaje “Tus modificaciones han sido guardadas”, dando la opción de insertar un nuevo torneo o ir al listado de los torneos.
2 El Administrador del sistema da clic en el botón “Listado”.	2.1 El sistema muestra el listado de los torneos.
Poscondiciones	Los datos del torneo han sido modificados.

2.7.1.8 Descripción textual del Caso de Uso de sistema: Gestionar Partida

Tabla 9 Descripción textual del CUS: Gestionar Partida

Caso de Uso:	Gestionar Partida
Actores:	Administrador
Resumen:	El Administrador del sistema tendrá la posibilidad de insertar, modificar o eliminar la información referente a las partidas.
Precondiciones:	El administrador debe estar previamente autentificado en el sistema.
Referencias	RF11
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El Administrador del sistema solicita la interfaz correspondiente a la gestión de información para insertar, modificar o eliminar la información correspondiente a una partida.	1.1 El sistema muestra un listado de partidas, permitiendo las siguientes opciones: Nuevo. Editar.

	Eliminar.
2 El Administrador del sistema da clic en el botón “Nuevo”.	2.1 Para esto va al escenario “Insertar Partida”.
3 El Administrador del sistema da clic en el botón “Editar”, en la partida que deseo modificar.	3.1 Para esto va al escenario “Modificar Partida”.
4 El Administrador del sistema da clic en el botón “Eliminar”, en el partida que deseo eliminar.	4.1 Para esto va al escenario “Eliminar Partida”. Terminando el CUS.
Flujo Normal de Eventos **Sección “Insertar partida”**	
Acción del Actor	**Respuesta del Sistema**
	2.1 El sistema muestra los campos a llenar para adicionar una nueva partida, los mismos son: Eco. Id Jugador N. Id Jugador B. Fecha de Nacimiento. Fecha. Resultado B. Tiempo. Planilla. Resultado N Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado.
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón “Guardar”.	3.1 El sistema adiciona la partida y muestra un mensaje “Tus modificaciones han sido guardadas”, permitiendo además modificar los datos insertados.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón "Guardar y adicionar nuevo".	3.1 El sistema muestra un mensaje "Tus modificaciones han sido guardadas", dando la opción de insertar una nueva partida o ir al listado de partidas.
3 El Administrador del sistema da clic en el botón "Listado".	3.1 El sistema muestra el listado de partidas.
Poscondiciones	La partida ha sido insertada exitosamente.
Flujo Normal de Eventos **Sección "Eliminar Partida"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra un mensaje "Estas seguro".
2 El Administrador del sistema acepta la operación.	2.1 El sistema elimina la partida seleccionada.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 El Administrador del sistema cancela la operación.	2.1 El sistema cancela la operación y se mantiene en la misma página mostrando el listado de las partidas.
Poscondiciones	La partida ha sido eliminada exitosamente.
Flujo Normal de Eventos **Sección "Modificar Partida"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra los campos a llenar para adicionar una nueva partida, los mismos son: Eco. Id Jugador N.

	Id Jugador B. Fecha de Nacimiento. Fecha. Resultado B. Tiempo. Planilla. Resultado N Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado.
2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón “Guardar”.	2.1 El sistema guarda los cambios y muestra un mensaje “Tus modificaciones han sido guardadas”.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón “Guardar y adicionar nuevo”.	2.1 El sistema muestra un mensaje “Tus modificaciones han sido guardadas”, dando la opción de insertar una nueva partida o ir al listado de partidas.
2 El Administrador del sistema da clic en el botón “Listado”.	2.1 El sistema muestra el listado de las partidas.
Poscondiciones	Los datos de la partida han sido modificados.

2.7.1.9 Descripción textual del Caso de Uso de sistema: Gestionar Usuario

Tabla 10 Descripción del CUS: Gestionar Usuario

Caso de Uso:	Gestionar Usuario
Actores:	Administrador
Resumen:	El Administrador del sistema tendrá la posibilidad de insertar, modificar o

	eliminar la información referente a los usuarios.
Precondiciones:	El administrador debe estar previamente autentificado en el sistema.
Referencias	RF12
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 El Administrador del sistema solicita la interfaz correspondiente a la gestión de información para insertar, modificar o eliminar la información correspondiente a un usuario.	1.1 El sistema muestra un listado de usuarios, permitiendo las siguientes opciones: Nuevo. Editar. Eliminar.
2 El Administrador del sistema da clic en el botón “Nuevo”.	2.1 Para esto va al escenario “Insertar Usuario”.
3 El Administrador del sistema da clic en el botón “Editar”, en la partida que deseo modificar.	3.1 Para esto va al escenario “Modificar Usuario”.
4 El Administrador del sistema da clic en el botón “Eliminar”, en el partida que deseo eliminar.	4.1 Para esto va al escenario “Eliminar Usuario”. Terminando el CUS.
Flujo Normal de Eventos **Sección “Insertar usuario”**	
Acción del Actor	**Respuesta del Sistema**
	2.1 El sistema muestra los campos a llenar para adicionar un nuevo usuario, los mismos son: Usuario. Nombre. Correo. Rol. Contraseña. Nueva Contraseña.

	Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado.
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón "Guardar".	3.1 El sistema adiciona el usuario y muestra un mensaje "Tus modificaciones han sido guardadas", permitiendo además modificar los datos insertados.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El Administrador del sistema llena los campos mostrados por el sistema y da clic en el botón "Guardar y adicionar nuevo".	3.1 El sistema muestra un mensaje "Tus modificaciones han sido guardadas", dando la opción de insertar un nuevo usuario o ir al listado de usuarios.
3 El Administrador del sistema da clic en el botón "Listado".	3.1 El sistema muestra el listado de usuarios.
Poscondiciones	El usuario ha sido adicionado exitosamente.
Flujo Normal de Eventos **Sección "Eliminar Usuario"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra un mensaje "Estas seguro".
2 El Administrador del sistema acepta la operación.	2.1 El sistema elimina el usuario seleccionado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 El Administrador del sistema cancela la operación.	2.1 El sistema cancela la operación y se mantiene en la misma página mostrando el listado de los usuarios.

Poscondiciones	El usuario ha sido eliminado exitosamente.
Flujo Normal de Eventos **Sección "Modificar Usuario"**	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema muestra los campos a llenar para adicionar un nuevo usuario, los mismos son: Usuario. Nombre. Correo. Rol. Contraseña. Nueva Contraseña. Dando la opción de: Guardar. Guardar y adicionar nuevo. Listado.
2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón "Guardar".	2.1 El sistema guarda los cambios y muestra un mensaje "Tus modificaciones han sido guardadas".
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 El Administrador del sistema llena los campos que deseo modificar y da clic en el botón "Guardar y adicionar nuevo".	2.1 El sistema muestra un mensaje "Tus modificaciones han sido guardadas", dando la opción de insertar un nuevo usuario o ir al listado de usuarios.
2 El Administrador del sistema da clic en el botón "Listado".	2.1 El sistema muestra el listado de los usuarios.
Poscondiciones	Los datos del usuario han sido modificados.

2.7.1.10 Descripción textual del Caso de Uso de sistema: Permitir autentificarse

Tabla 11 Descripción textual CUS: Permitir autentificarse

Caso de Uso:	Permitir autentificarse	
Actores:	Usuario	
Resumen:	El CU se inicia cuando el usuario introduce sus datos en el sistema para autenticarse; y en este, según los datos introducidos le son asignados al usuario sus privilegios y en caso de que no sean correctos se le niega el acceso mostrando un mensaje de error.	
Precondiciones:		
Referencias	RF9	
Prioridad	Secundario	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1 El usuario entra sus datos de acceso al sistema		1.1 El sistema verifica los datos.
		1.2 Si los datos están correctos y el usuario está registrado, el sistema le asigna sus privilegios según su rol y le da entrada.
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
		1.2 Si el sistema no identifica los datos del usuario muestra un mensaje denegando acceso.
Poscondiciones	La autenticación se realice de forma correcta.	

2.7.1.10 Descripción textual del Caso de Uso de sistema: Registrarse

Tabla 12 Descripción del CUS: Registrarse

Caso de Uso:	Registrarse	
Actores:	Usuario	
Resumen:	El CU se inicia cuando un usuario Anónimo entra al sistema y selecciona la opción de registrarse.	
Precondiciones:		
Referencias	RF13	
Prioridad	Secundario	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1 El usuario da clic en el botón registrarse.		1.1 El sistema muestra los campos a llenar, los mismos son: Usuario. Nombre. Correo. Rol. Contraseña. Nueva Contraseña.
2 El usuario llena los datos y da clic en el botón “Salvar”		2.1 El sistema verifica si los datos están correctos.
		2.2 El sistema registra el nuevo usuario
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
		2.2 Los datos no están correctos y el sistema muestra un mensaje especificando los campos incorrectos.
Poscondiciones	El registro del usuario se realizó de forma correcta.	

2.8 CONCLUSIONES

Durante este capítulo fueron expuestas las características que contendrá el sistema, apoyándose para ello en el análisis de los actuales procesos de dominio. Se identificaron además los requisitos funcionales y no funcionales que debe cumplir el sistema en cuestión, y con ello fueron presentados los casos de uso a tratar durante el desarrollo del mismo, con la correspondiente descripción textual de cada uno. Esto brinda una visión general de qué debe hacer el sistema, por lo que se está en condiciones de pasar a ver cómo es y qué operaciones antes descritas va a realizar para dar solución a los problemas planteados.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 INTRODUCCIÓN

En este capítulo se expone la concepción general referente al análisis y diseño del sistema propuesto, donde el análisis tiene como objetivo mantener un modelo eficiente de la solución propuesta que sirva de base para el diseño. Este último tiene a su vez como objetivo presentar cómo es que está construido el sistema, lo cual se realiza en este caso a partir de los diagramas de clases Web, que tienen la finalidad de describir la interacción entre las distintas páginas de la aplicación. Por otro lado, se presenta el diagrama de clases persistentes de la base de datos del sistema, además de los principios de diseño y programación utilizados.

3.2 ANÁLISIS

3.2.1 DIAGRAMAS 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.

A continuación se muestran algunos diagramas de clases del análisis :(Ver Anexo 1)

3.2.1.1 Caso de uso: Consultar las características de las partidas de un jugador

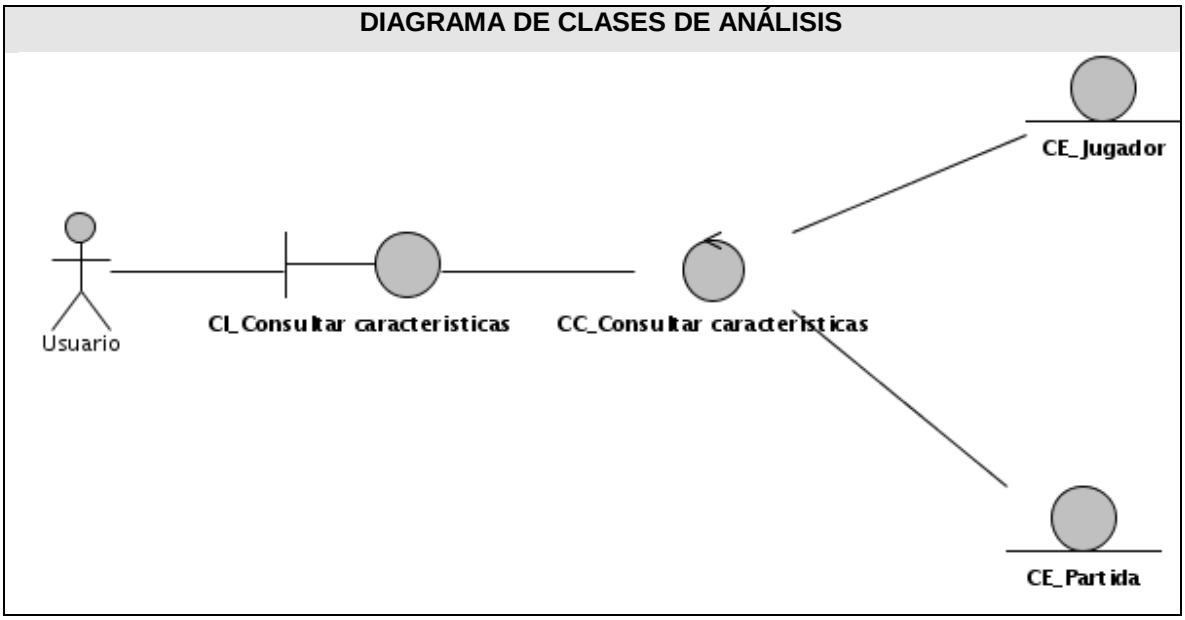

Figura 3 Diagrama de clases del análisis del CUS: Consultar las características de las partidas de un jugador

3.2.1.2 Caso de uso: Seleccionar Jugador

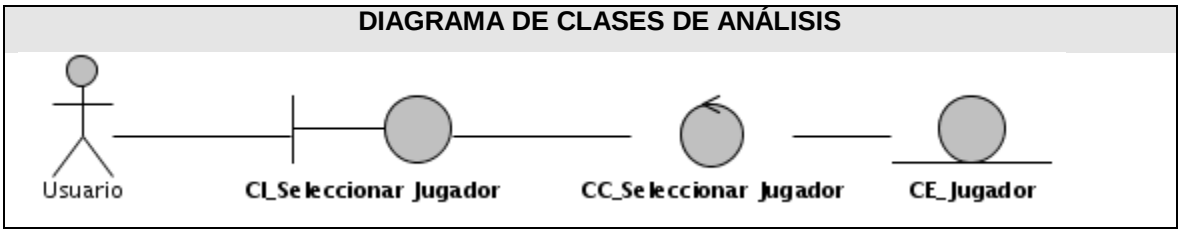

Figura 4 Diagrama de clases del análisis del CUS: Seleccionar Jugador

3.2.1.3 Caso de uso: Mostrar comportamiento por ECO

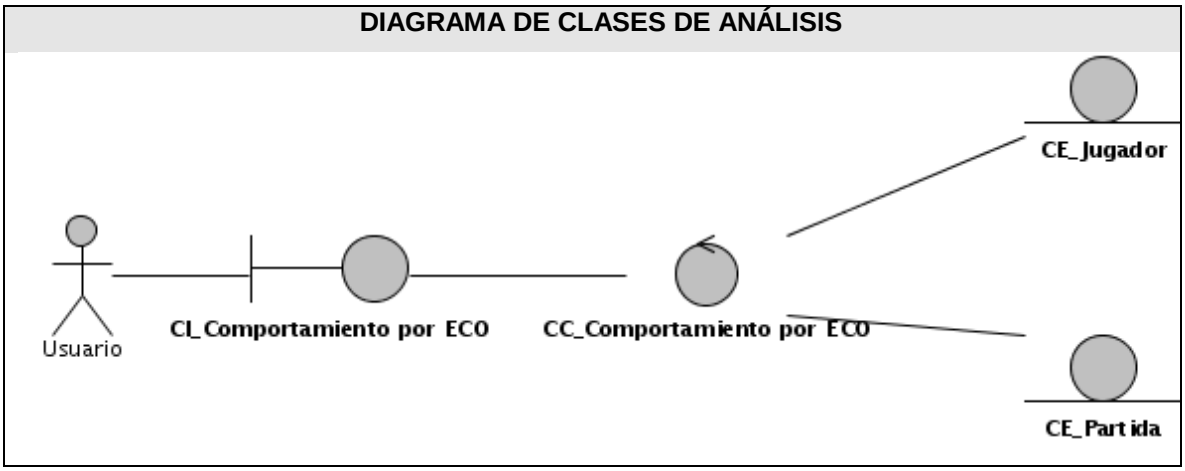

Figura 5 Diagrama de clases del análisis del CUS: Mostrar comportamiento por ECO

3.2.1.4 Caso de uso: Mostrar comportamiento por Resultados

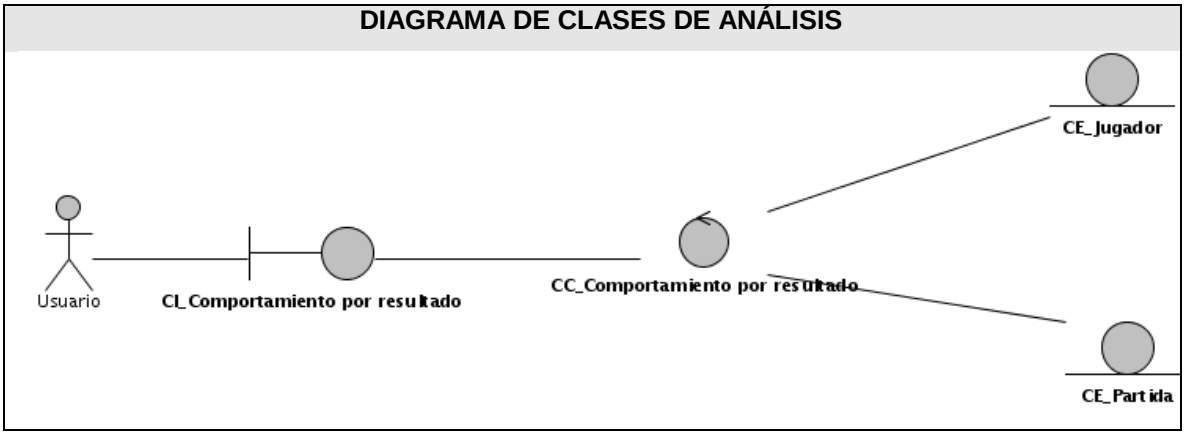

Figura 6 Diagrama de clases del análisis del CUS: Mostrar comportamiento por Resultados

3.2.1.5 Caso de uso: Mostrar comportamiento por color de las piezas

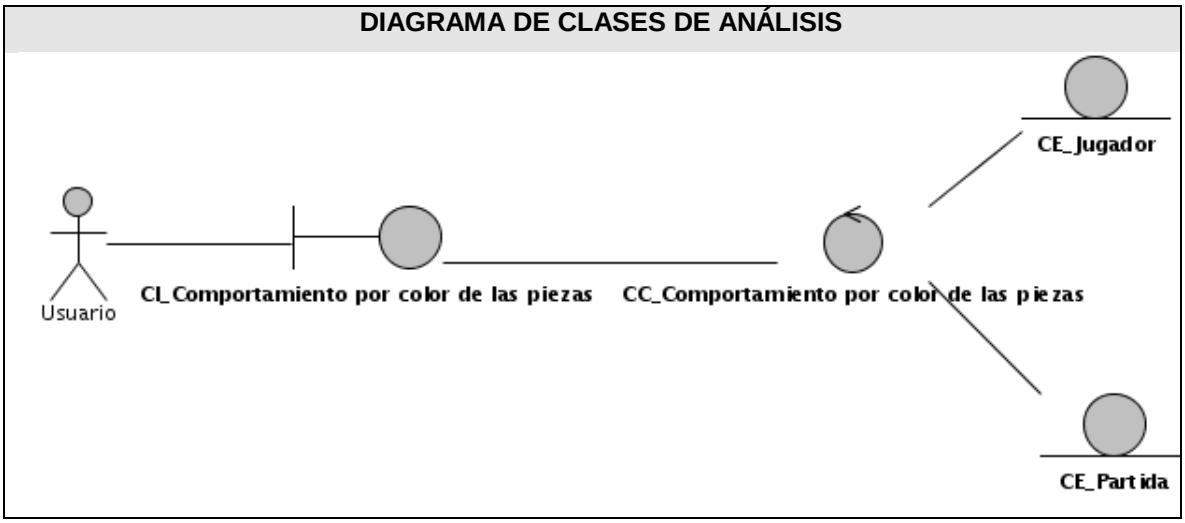

Figura 7 Diagrama de clases del análisis del CU: Mostrar comportamiento por color de las piezas

3.3 DISEÑO

3.3.1 DIAGRAMAS DE INTERACCIÓN

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, estos muestran una interacción que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes; un diagrama de colaboración es un diagrama de interacción que expone la organización estructural de los objetos que envían y reciben mensajes.

3.3.1.1 Caso de uso: Consultar las características de las partidas de un jugador

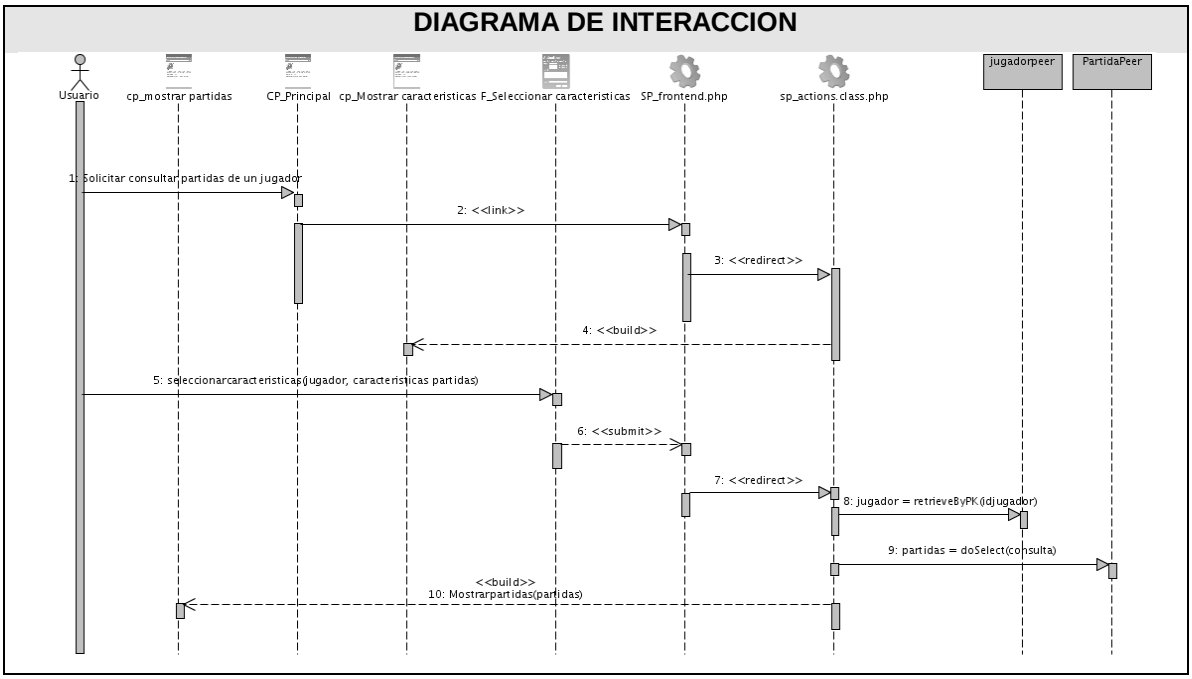

Figura 8 Diagrama de secuencia del CU: Consultar las características de las partidas de un jugador

3.3.1.2 Caso de uso: Seleccionar jugador

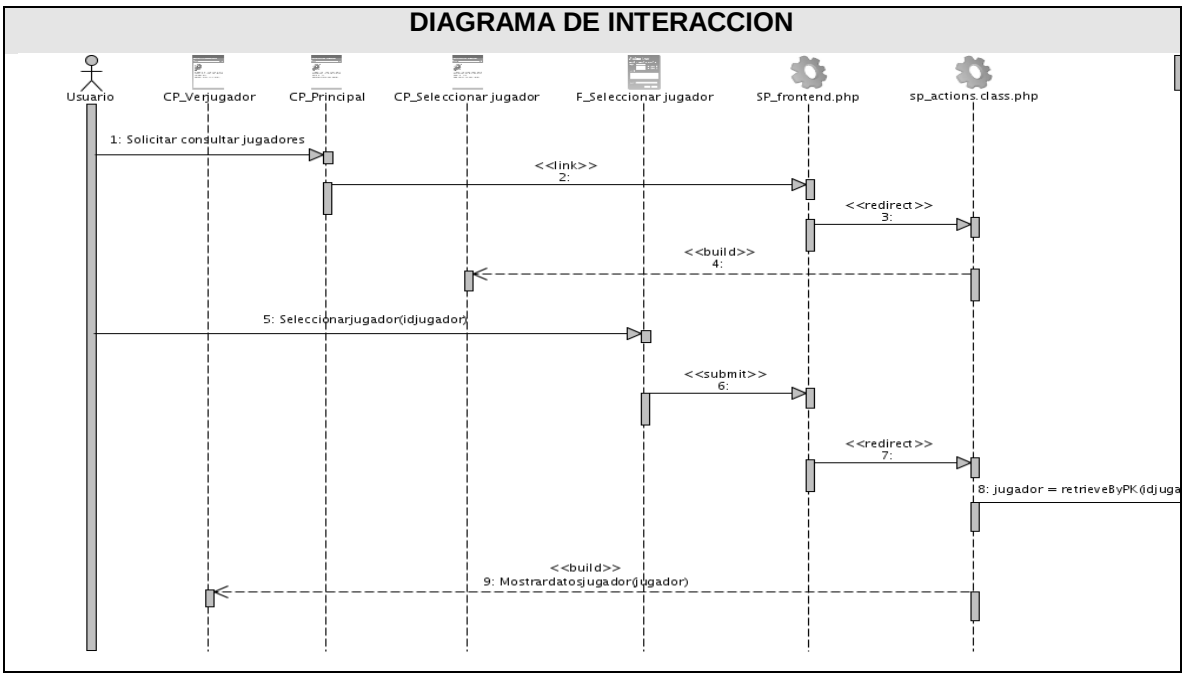

Figura 9 Diagrama de secuencia del CU: Seleccionar jugador

3.3.1.3 Caso de uso: Mostrar comportamiento por ECO

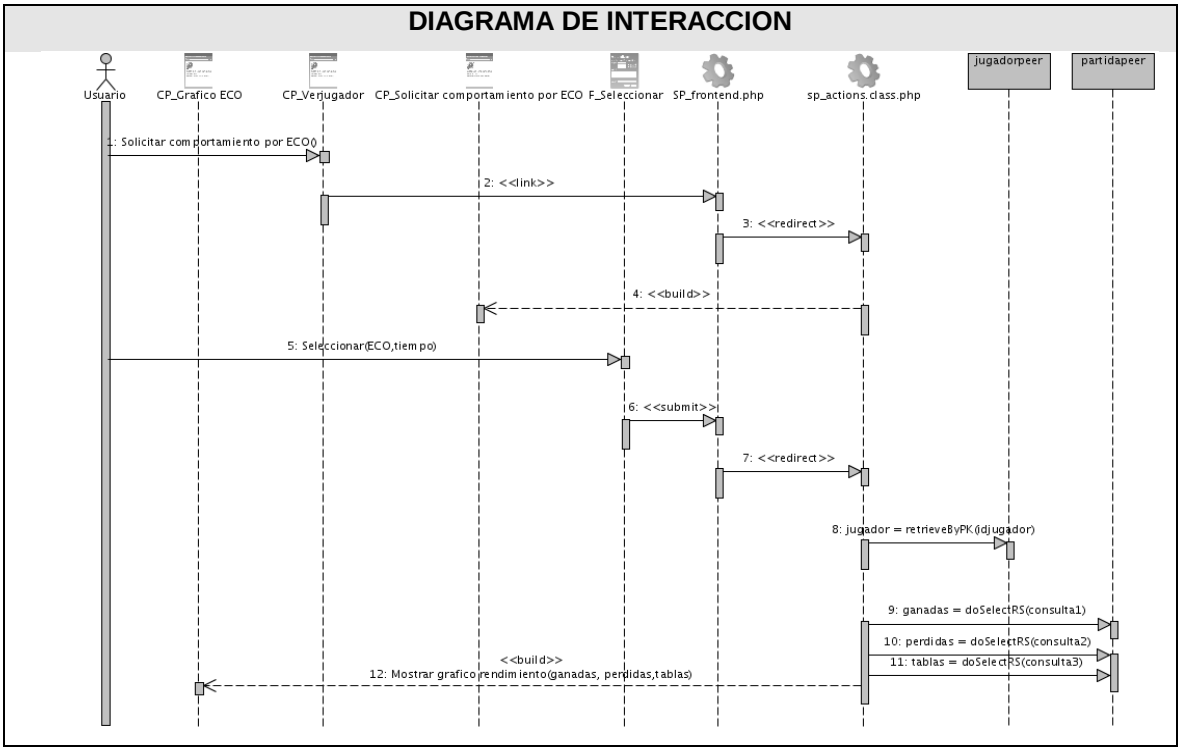

Figura 10 Diagrama de secuencia del CU: Mostrar comportamiento por ECO

3.3.1.4 Caso de uso: Mostrar comportamiento por Resultados

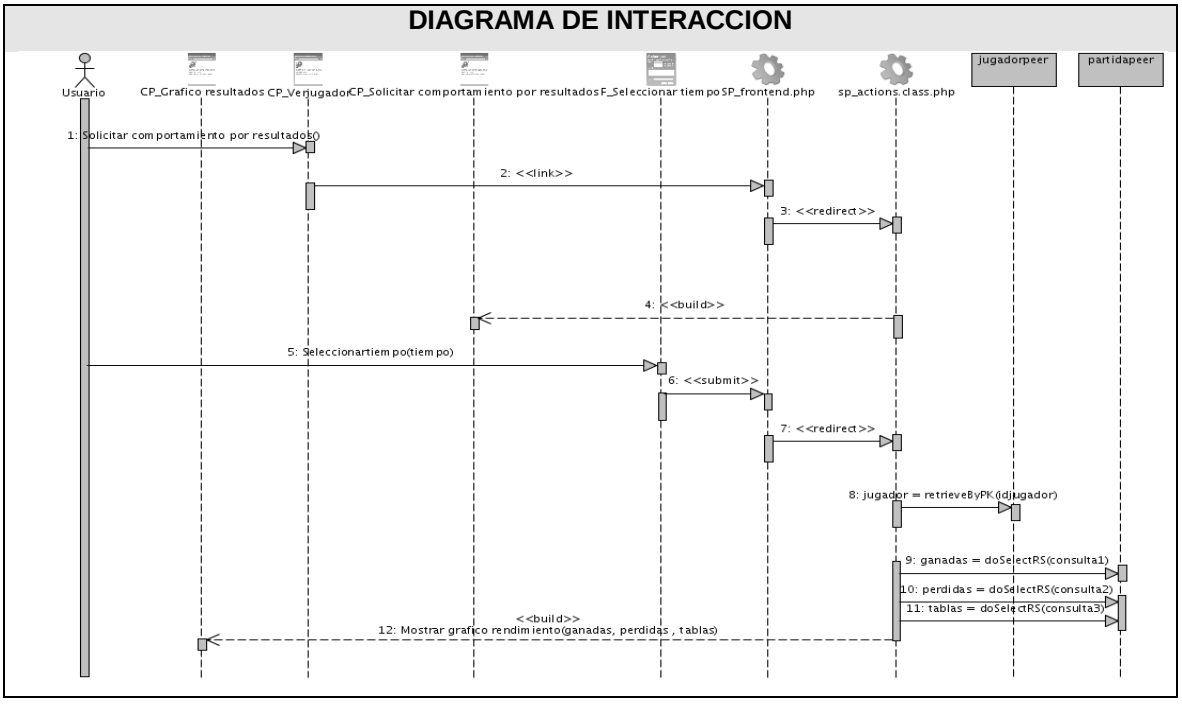

Figura 11 Diagrama de secuencia del CU: Mostrar comportamiento por resultados

3.3.1.5 Caso de uso: Mostrar comportamiento por color de las piezas

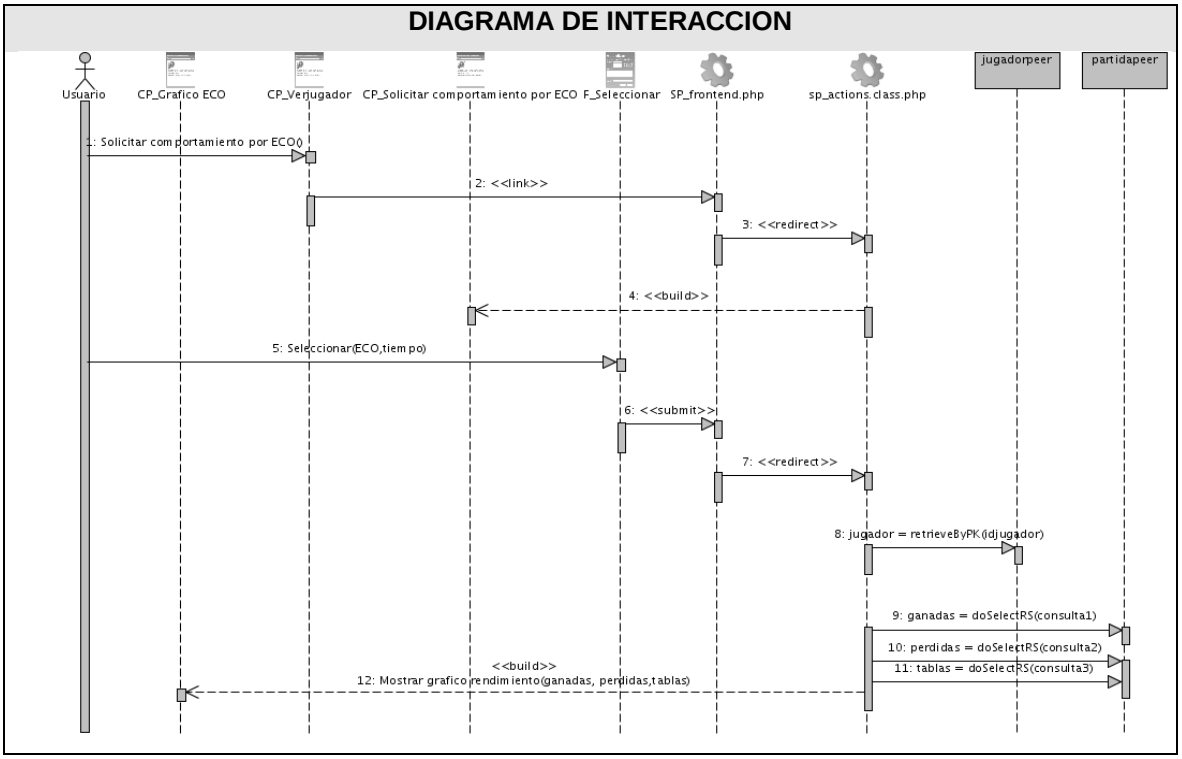

Figura 12 Diagrama de secuencia del CU: Mostrar comportamiento por color de las piezas

3.3.2 DIAGRAMAS 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 pagina lógica puede ser representada como una clase.

A continuación se brinda una explicación de qué representa cada estereotipo en el diseño:

Tabla 13 Estereotipos de diseño Web

 SP_<NombreClase Servidora>	<<Server Page>>: Representa la clase que tiene código que se ejecuta en el servidor, la cual se encarga de construir (build) o generar el resultado HTML y/o realizar peticiones a la capa inferior.

 CP_<NombreClase Cliente>	<<Client Page>>: Es una página Web con formato HTML/XHTML. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para cualquier función dentro de la página. Cada página cliente es construida por una sola página de servidor.
 F_<Nombre>	<<Form>>: Es una colección de elementos de entrada que están contenidos en la página cliente. Sus atributos son los elementos de entrada del formulario (input boxes, text areas, radio buttons, check boxes, hidden fields, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST y se comunican con las páginas servidores mediante submit.

Relaciones utilizadas entre clases:

<<build>>: Representa la relación existente entre las páginas cliente, que de forma general expresa cómo las páginas que se encuentran en el servidor construyen las páginas en el cliente. Es una relación direccional, donde una página servidor construye una o más páginas cliente.

<<redirect>>: Una página servidora puede redireccionar el procesamiento a otra página, es decir, enviar información para que la otra ejecute la acción.

<<Link>> Expresa las asociaciones más comunes entre las páginas, en este caso la del hipervínculo; esta asociación siempre se origina desde una página cliente y apunta hacia otra página cliente o una página de servidor.

<<Submit>> Es la relación que se crea siempre entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.

A continuación se muestran los diagramas de clases Web para algunos de los casos de uso del sistema: (Ver Anexo 2)

3.3.2.1 Diagrama de clases Web para el caso de uso "Consultar las características de las partidas de un jugador"

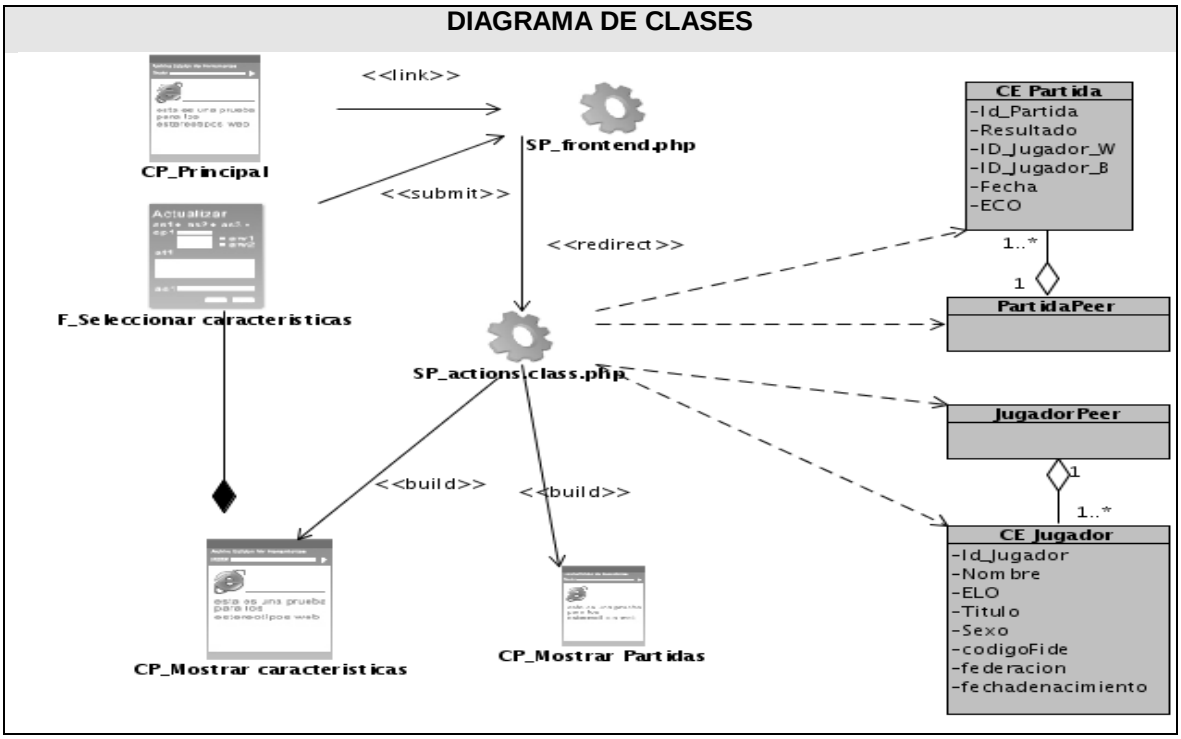

Figura 13 Diagrama de clases Web para el caso de uso "Consultar las características de las partidas de un jugador"

3.3.2.2 Diagrama de clases Web para el caso de uso "Seleccionar Jugador"

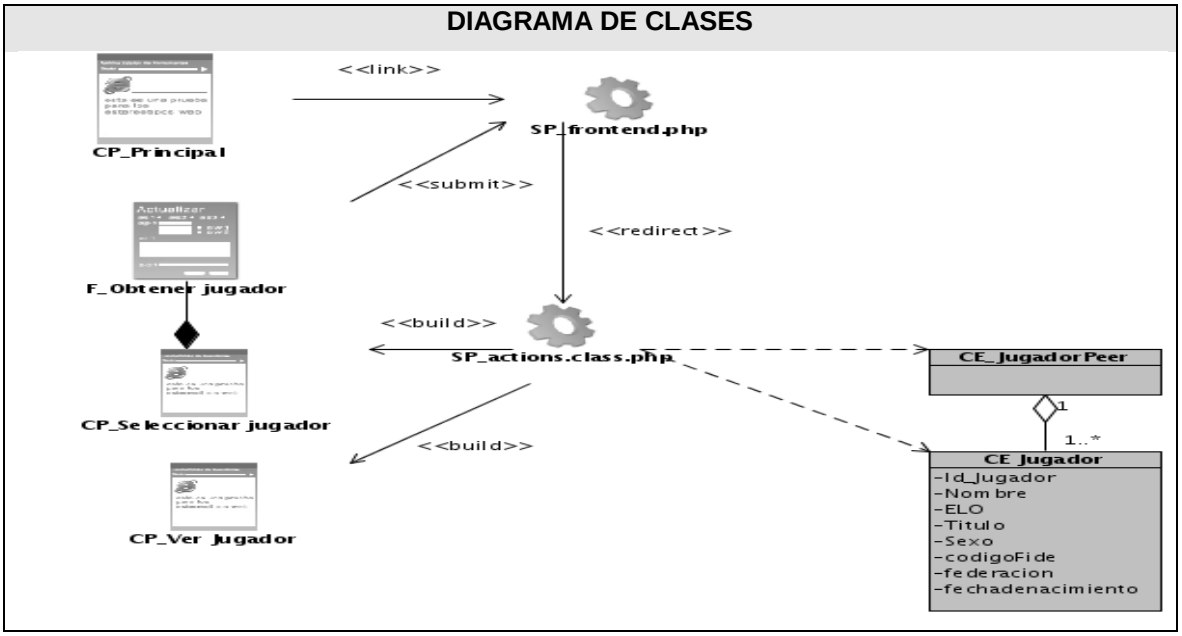

Figura 14 Diagrama de clases Web para el caso de uso "Seleccionar Jugador"

3.3.2.3 Diagrama de clases Web para el caso de uso "Mostrar comportamiento por ECO"

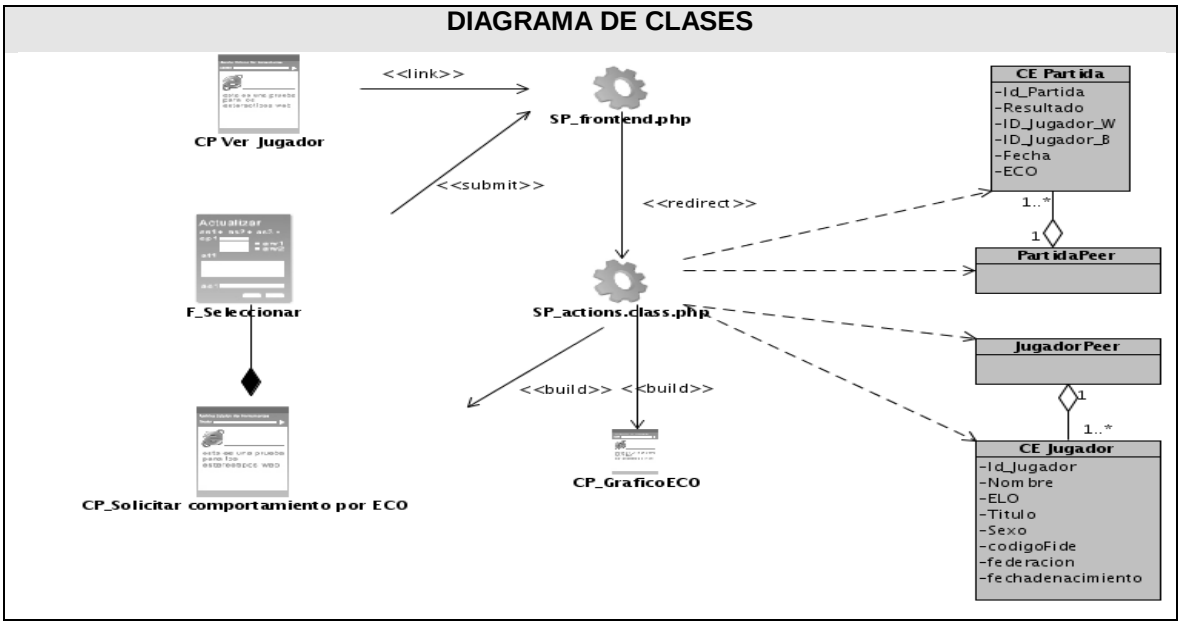

Figura 15 Diagrama de clases Web para el caso de uso "Mostrar comportamiento por ECO"

3.3.2.4 Diagrama de clases Web para el caso de uso "Mostrar comportamiento por Resultados"

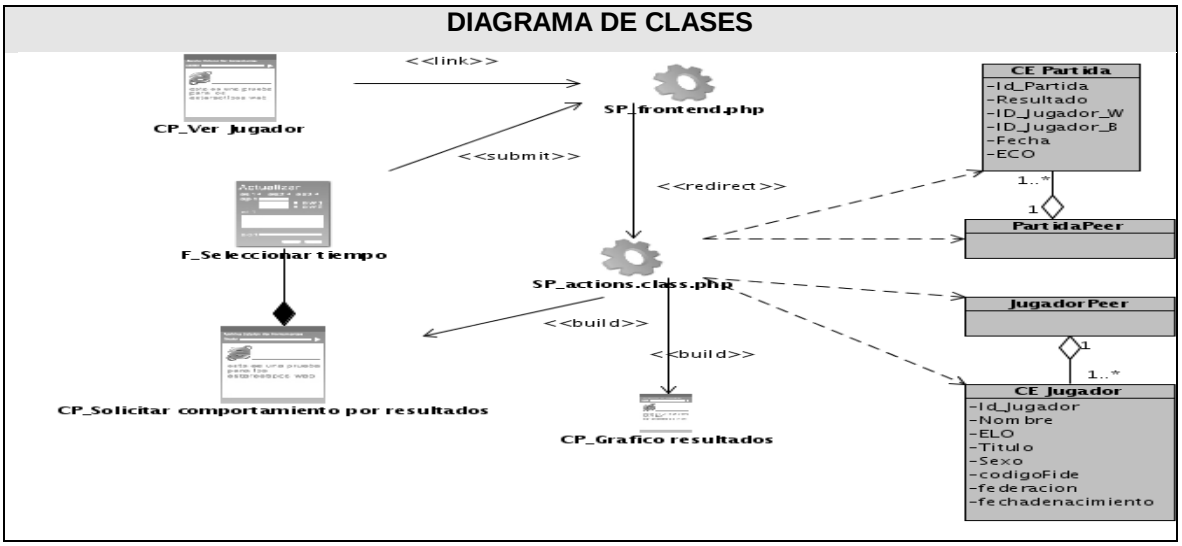

Figura 16 Diagrama de clases Web para el caso de uso "Mostrar comportamiento por Resultados"

3.3.2.5 Diagrama de clases Web para el caso de uso "Mostrar comportamiento por color de las piezas"

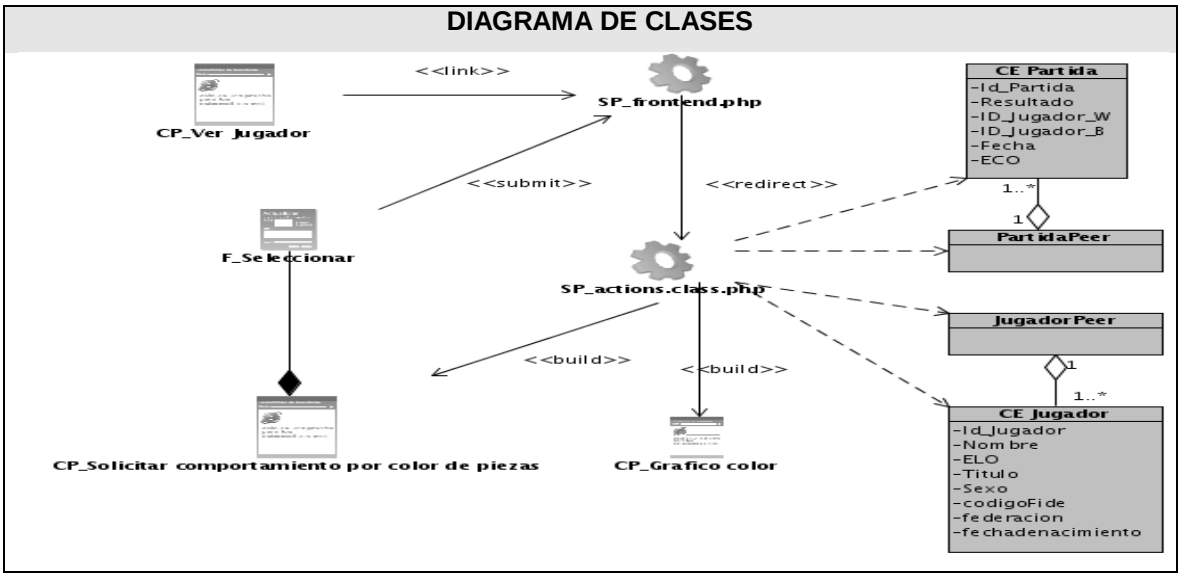

Figura 17 Diagrama de clases Web para el caso de uso "Mostrar comportamiento por color de las piezas"

3.4 DESCRIPCIÓN TEXTUAL DE LAS CLASES WEB

La descripción textual de las clases del diseño ayudan a comprender el alcance y la responsabilidad de cada una de estas dentro del sistema, es decir, con dicha descripción se

puede conocer qué funcionalidad específica realiza cada clase, además de la información que cada una de esta maneja.

Clases Entidad

Nombre	CE_Partida	
Tipo de Clase	Entidad	
Atributo		**Tipo**
idpartida		Int
eco		Varchar
idjugadorw		Int
idjugadorb		Int
fecha		Date
Resultado_B		Int
tiempo		float
planillla		longtext
Resultado_N		Int
Responsabilidad		
Nombre	getidpartida()	
	geteco()	
	getidjugadorw()	
	getidjugadorb()	
	getfecha()	
	getResultado_B()	
	gettiempo()	
	getplanillla()	
	getResultado_N()	
	setIdpartida($v)	
	seteco($v)	
	setIdjugadorw($v)	
	setIdjugadorb($v)	
	setfecha($v)	
	setResultado_B($v)	

	setResultado_N($v)
	setTiempo($v)
	setplanilla($v)
Descripción	Clase entidad que representa la tabla partida

Nombre	CE_Jugador	
Tipo de Clase	Entidad	
Atributo		**Tipo**
idjugador		Int
idtitulojugador		Int
nombre		Varchar
sexo		Varchar
fechanacimiento		Date
elo		Varchar
idfide		Int
federacion		Int
Responsabilidad		
Nombre	getidjugador()	
	getidtitulojugador()	
	getnombre()	
	getsexo()	
	getfechanacimiento()	
	getElo()	
	getidfide()	
	getfederacion()	
	setidjugador($v)	
	setidtitulojugador($v)	
	setnombre($v)	
	setsexo($v)	
	setfechanacimiento($v)	

	setElo($v)
	setidfide($v)
	setfederacion($v)
Descripción	Clase entidad que representa la tabla jugador

Clases Acceso a datos

Nombre	PartidaPeer	
Tipo de Clase	Acceso a datos	
Atributo		**Tipo**
DATABASE_NAME		
TABLE_NAME		
CLASS_DEFAULT		
NUM_COLUMNS		
NUM_LAZY_LOAD_COLUMNS		
IDPARTIDA		
ECO		
IDJUGADORW		
IDJUGADORB		
FECHA		
RESULTADO_B		
TIEMPO		
PLANILLA		
RESULTADO_N		
Responsabilidad		
Nombre	getMapBuilder()	
	getPhpNameMap()	
	translateFieldName($name, $fromType, $toType)	
	getFieldNames($type = BasePeer::TYPE_PHPNAME)	

	addSelectColumns(Criteria $criteria)
	doCount(Criteria $criteria, $distinct = false, $con = null)
	doSelectOne(Criteria $criteria, $con = null)
	doSelect(Criteria $criteria, $con = null)
	populateObjects(ResultSet $rs)
	doInsert($values, $con = null)
	doUpdate($values, $con = null)
	doDeleteAll($con = null)
	doDelete($values, $con = null)
	retrieveByPK($pk, $con = null)
	retrieveByPKs($pks, $con = null)
Descripción	Clase acceso a datos que accede a los datos de la tabla Partida en la base de datos

Nombre	JugadorPeer
Tipo de Clase	Acceso a datos
Atributo	**Tipo**
DATABASE_NAME	
TABLE_NAME	
CLASS_DEFAULT	
NUM_COLUMNS	
NUM_LAZY_LOAD_COLUMNS	
IDJUGADOR	
IDTITULOJUGADOR	
NOMBRE	
SEXO	
FECHANACIMIENTO	
ELO	
IDFIDE	
FEDERACION	

Responsabilidad	
Nombre	getMapBuilder()
	getPhpNameMap()
	translateFieldName($name, $fromType, $toType)
	getFieldNames($type = BasePeer::TYPE_PHPNAME)
	addSelectColumns(Criteria $criteria)
	doCount(Criteria $criteria, $distinct = false, $con = null)
	doSelectOne(Criteria $criteria, $con = null)
	doSelect(Criteria $criteria, $con = null)
	populateObjects(ResultSet $rs)
	doInsert($values, $con = null)
	doUpdate($values, $con = null)
	doDeleteAll($con = null)
	doDelete($values, $con = null)
	retrieveByPK($pk, $con = null)
	retrieveByPKs($pks, $con = null)
Descripción	Clase acceso a datos que accede a los datos de la tabla jugador en la base de datos

Clases Controladoras

Nombre	Frontend.php	
Tipo de Clase	Controladora	
Atributo		**Tipo**
Responsabilidad		
Nombre	sfContext::getInstance()->getController()->dispatch();	
Descripción	Clase controladora que se encarga del manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares.	

Nombre	Aptions.class.php (consultar jugadores)
Tipo de Clase	Controladora

Atributo		**Tipo**
Responsabilidad		
Nombre	executeIndex()	
	executeResultados()	
	executeVerjugador()	
	executeColor()	
	validateGraficoeco()	
	executeEco()	
	handleErrorGraficoeco()	
	executeGraficoresultados()	
	executeGraficoeco()	
Descripción	Clase controladora que se encarga de controlar las páginas de la sección consultar jugadores, así como de la interacción con las clases del modelo.	

Nombre	Aptions.class.php (consultar partidas)	
Tipo de Clase	Controladora	
Atributo		**Tipo**
Responsabilidad		
Nombre	executeIndex()	
	executeVerpartida()	
Descripción	Clase controladora que se encarga de controlar las páginas de la sección consultar partidas, así como de la interacción con las clases del modelo.	

3.5 DIAGRAMA DE CLASES PERSISTENTES

Las clases persistentes son aquellas que necesitan ser capaz 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:

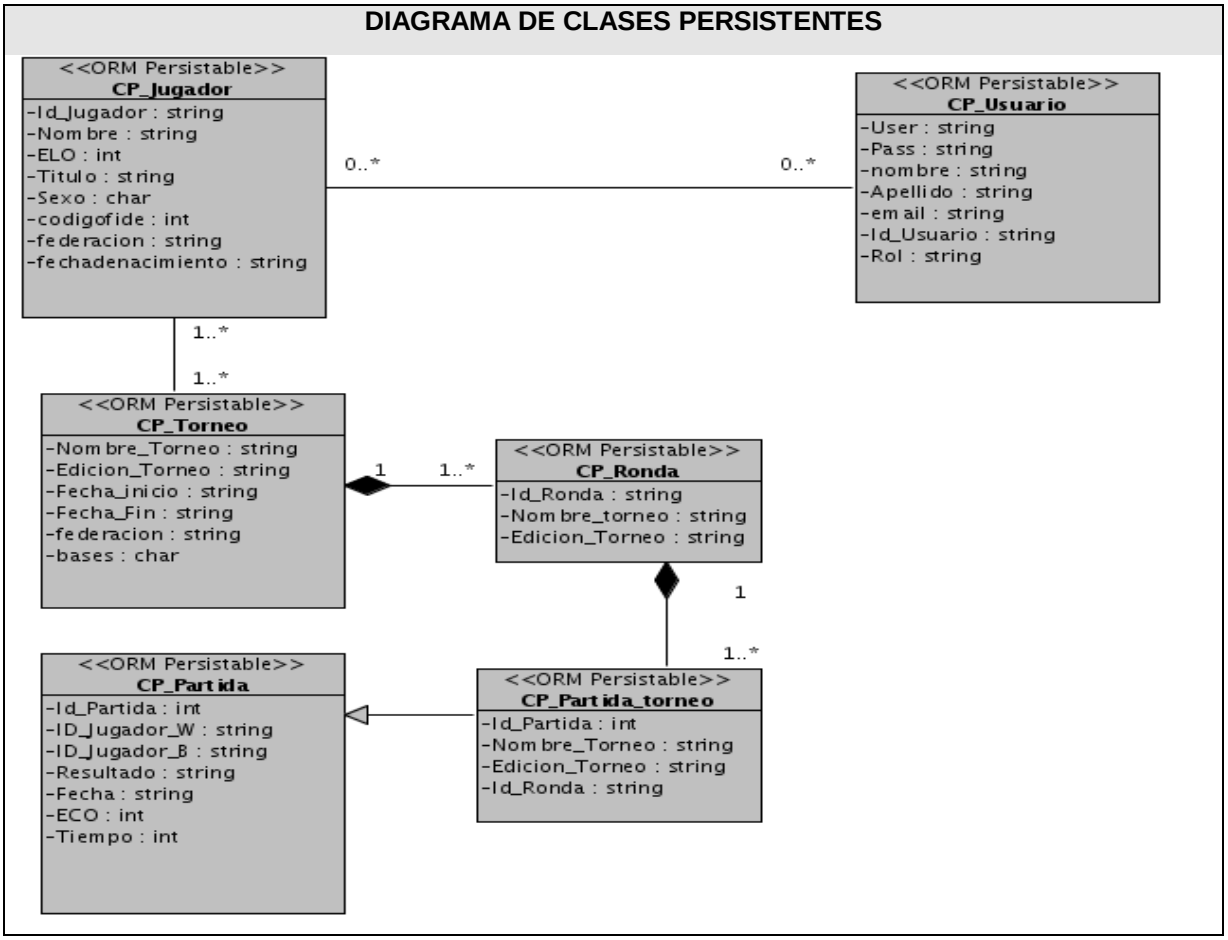

Figura 18 Diagrama de clases Persistentes

3.6 DEFINICIONES DE DISEÑO QUE SE APLICAN

Todo sistema, sin interesar el propósito para el cual fue creado, debe basar su diseño en el usuario que lo utilizará. En este caso es cualquier persona que desee obtener reportes estadísticos relativos a partidas o jugadores de Ajedrez. Estas personas no necesariamente deben poseer conocimientos relativos a cuestiones informáticas. Para garantizar la usabilidad, el sistema utiliza los siguientes principios generales de diseño:

Definiciones de diseño que se aplican

1. Principio de uso equiparable: plantea que las características de privacidad garantía y seguridad deben estar igualmente disponibles para todos los usuarios. Además el diseño debe ser atractivo para todos ellos.

2. Principio de uso flexible: manifiesta que debe ofrecer posibilidades de elección en los métodos de uso. Facilitar al usuario exactitud y precisión y adaptarse al paso o ritmo del usuario.

3. Principio de uso simple e intuitivo: plantea que debe eliminarse la complejidad innecesaria. Ser consistente con las expectativas e intuición del usuario. Proporcionar avisos eficaces y métodos de respuesta tras la finalización de las tareas.

4. Principio información perceptible: expone el uso de diferentes modos para presentar de manera redundante la información esencial. Proporcionar contraste suficiente entre la información esencial y sus alrededores.

5. Principio de tolerancia al error: manifiesta que debe disponer los elementos para minimizar los riesgos y errores como son: los elementos más usados, más accesibles; y los elementos peligrosos eliminados, aislados o tapados. Asimismo debe proporcionar advertencias sobre peligros y errores.

6. Principio de poco esfuerzo: plantea que debe permitir que el usuario mantenga una posición corporal neutra y que minimice las acciones repetitivas.

7. Principio de tamaño y espacio para el acceso y uso: Que proporcione una línea de visión clara hacia los elementos importantes tanto para un usuario sentado como de pie.

3.6.1 INTERFAZ DE USUARIO

El diseño de la interfaz de usuario se puede definir como: “el conjunto de trabajos y pasos que seguirá el usuario, durante todo el tiempo que se relacione con el programa, detallando lo que verá y escuchará en cada momento, y las acciones que realizará, así como las respuestas que el sistema dará”.

El diseño de todo sistema debe centrarse en el usuario que va a utilizarlo, que es quien determina el éxito o fracaso de este. Además de los principios antes mencionados, también se tuvo en cuenta lo siguiente:

- La utilización de un mismo formato y estilo en cada una de las páginas.

- Facilitarle al usuario la plena navegabilidad dentro de la aplicación.
- Evitar sobrecarga de colores e imágenes.
- Proporcionar un ambiente amigable.

Se utilizó también una hoja de estilo para guardar la configuración del diseño de todas las páginas del sistema, tanto para los botones como para el tipo y tamaño de letra, logrando una uniformidad en todas estas. Por otro lado, los formularios de entrada están centrados en la parte destinada para ello dentro de cada página, además de estar organizados según la prioridad de los datos que se necesitan.

3.6.2 TRATAMIENTO DE ERRORES

El correcto tratamiento de los errores en un sistema influye notablemente en el buen funcionamiento de este, a la vez que se garantiza la integridad de la información. Para lograr esto fueron previstos todos los errores posibles que pudiera generar el sistema a partir de la interacción del usuario con este, así como de problemas inesperados que pudieran surgir, como es el caso de la pérdida de la conexión con la base de datos por falta del fluido eléctrico, etc. Los mensajes de error son mostrados con un texto claro que señala de forma explícita y legible la respuesta del sistema ante cualquier acción que se ejecute. Además de esto, se muestran mensajes de confirmación ante acciones que son irreversibles, como es el caso de la eliminación de datos, a la vez que se muestran mensajes para indicar cuándo una acción fue realizada con éxito.

3.6.3 SEGURIDAD.

La seguridad del sistema se basa principalmente en el empleo de la autenticación como una acción de obligatorio cumplimiento para su uso posterior, por lo que cada usuario del sistema tendrá un nombre para identificarse y una contraseña, los cuales serán verificados antes de darle acceso a las funcionalidades del sistema y si algunos de estos datos son incorrectos, se denegará dicho acceso. La contraseña pasa por un proceso de encriptación en el cliente, siguiendo el algoritmo MD5 y posteriormente se verifica en el servidor, si la misma ha sido encriptada o no, para en caso negativo encriptarla y manejarla de esta forma. Por otra parte, cada usuario posee un rol, el cual es chequeado en cada página del sistema, con vistas a darle o no acceso a la misma en dependencia de cual sea este rol. También son validados cada uno de los campos de los formularios con el objetivo de evitar que se introduzcan o seleccionen datos no permisibles por el sistema.

3.7 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 que permite 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. Por último fueron expuestos los principios de diseño seguidos durante la implementación del sistema, entre los que se encuentran: Interfaz de usuario, Tratamiento de errores y Seguridad.

CAPÍTULO 4: IMPLEMENTACIÓN

4.1 INTRODUCCIÓN

En este capítulo se describen detalles sobre la implementación del sistema mediante los diagramas de componentes que representan los múltiples elementos físicos u archivos que conforman la aplicación. También se representa el diagrama de despliegue que de forma gráfica muestra la distribución del sistema en los diferentes elementos de hardware que le darán soporte.

4.2 DIAGRAMA DE DESPLIEGUE

En un diagrama de despliegue se muestran las relaciones físicas entre los componentes de hardware y de software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software, que no son más que los procesos que se ejecutan en ellos. Se puede decir también que un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación, donde un nodo puede contener instancias de componentes.

A continuación se muestra el diagrama de despliegue del sistema:

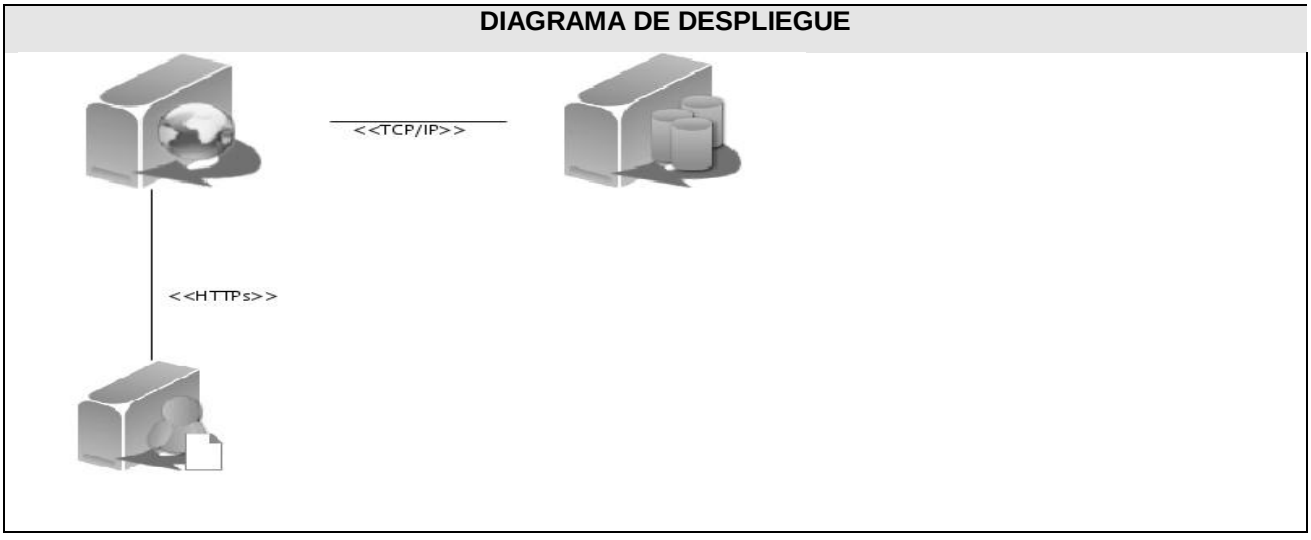

Figura 19 Diagrama de despliegue

4.3 DIAGRAMAS DE COMPONENTES

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Se presenta un

diagrama de componentes para el módulo principal (frontend) y otro para el módulo de administración (backend), así como uno por cada uno de los paquetes existentes en cada aplicación.

A continuación se muestra el diagrama de componentes perteneciente al módulo principal (frontend) (Ver Anexo 3)

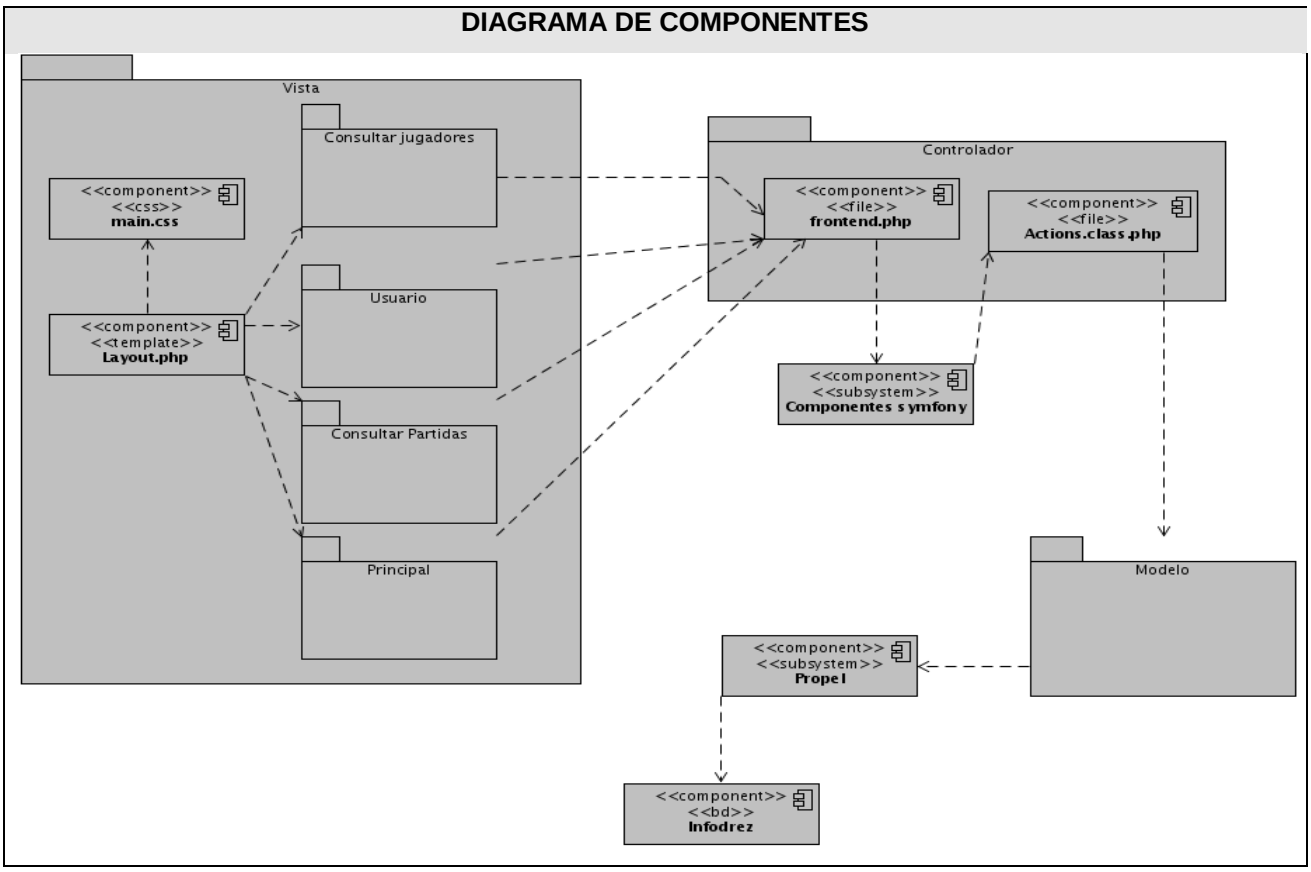

Figura 20 Diagrama de componentes del módulo" Frontend"

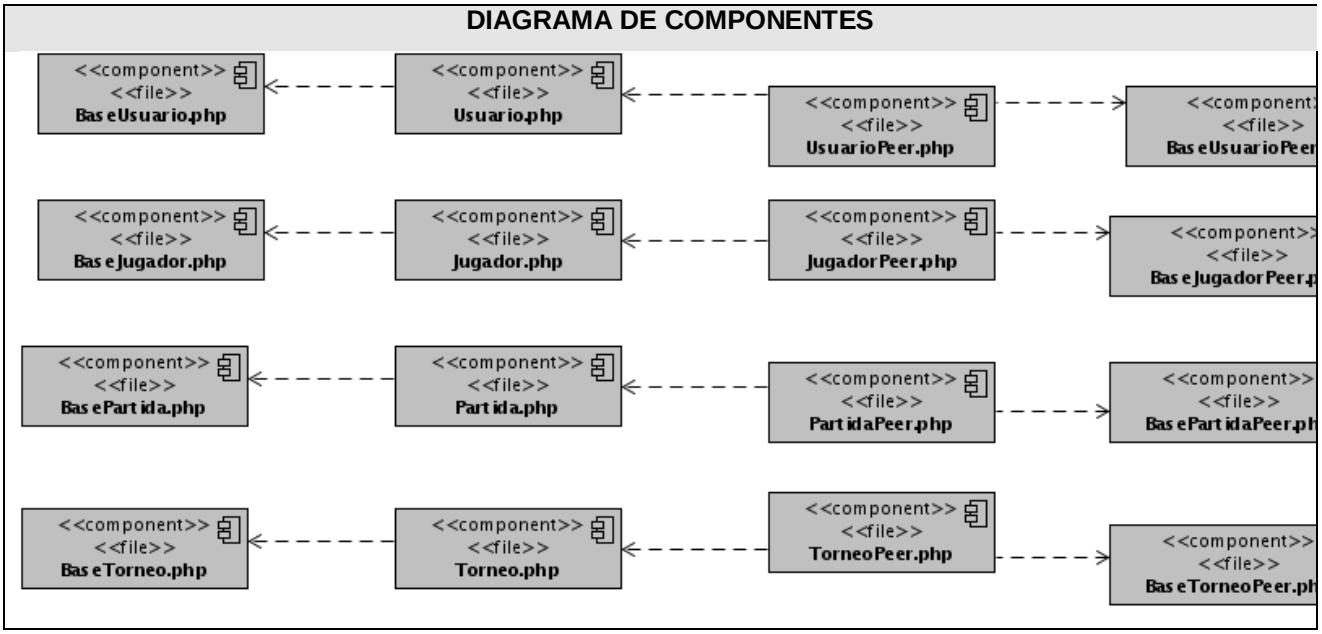

Figura 21 Diagrama de componentes del paquete "Modelo"

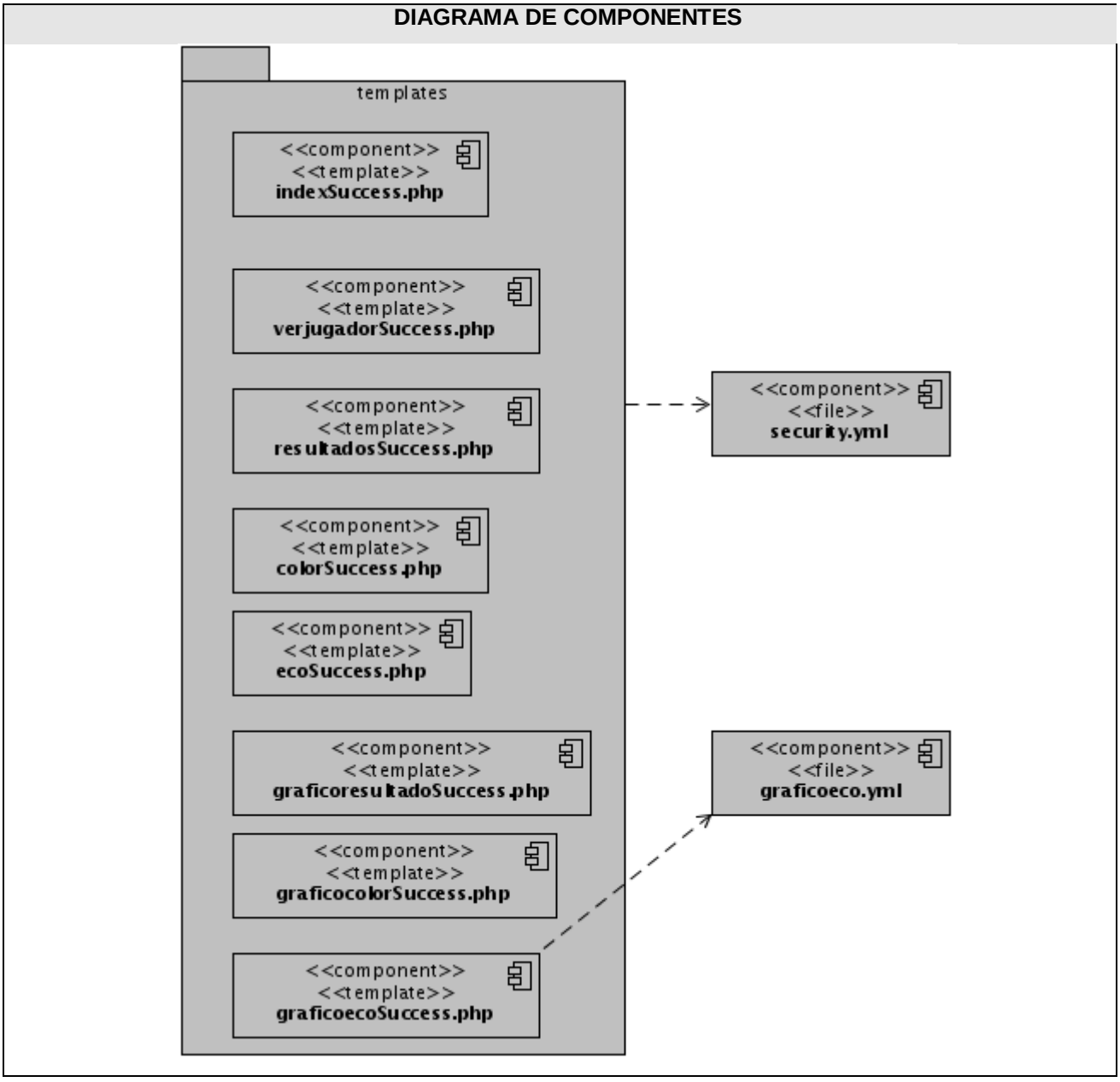

Figura 22 Diagrama de componentes del paquete "Consultar jugadores"

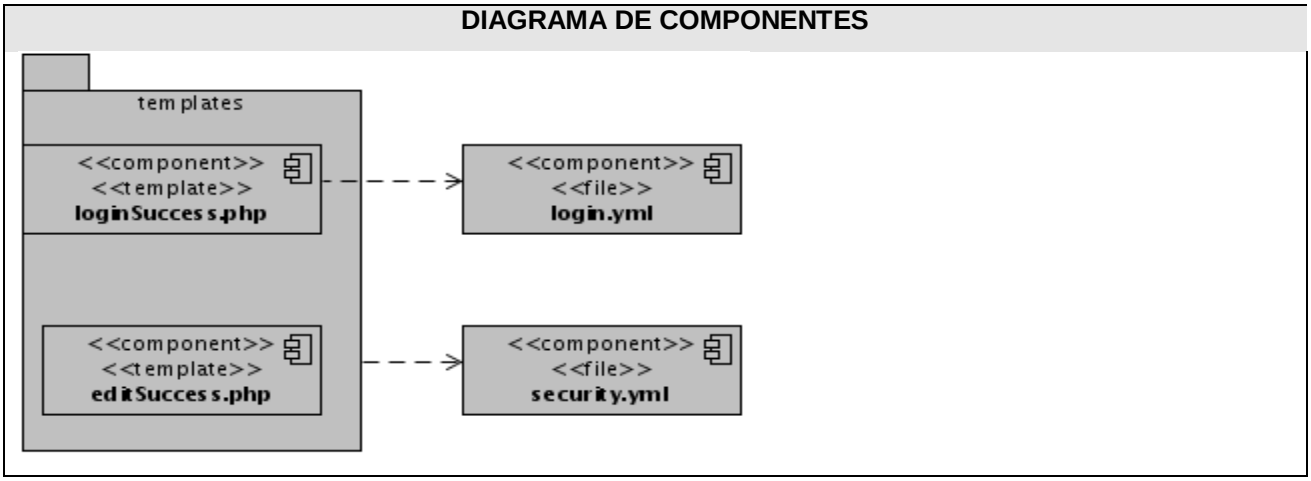

Figura 23 Diagrama de componentes del paquete "Usuario"

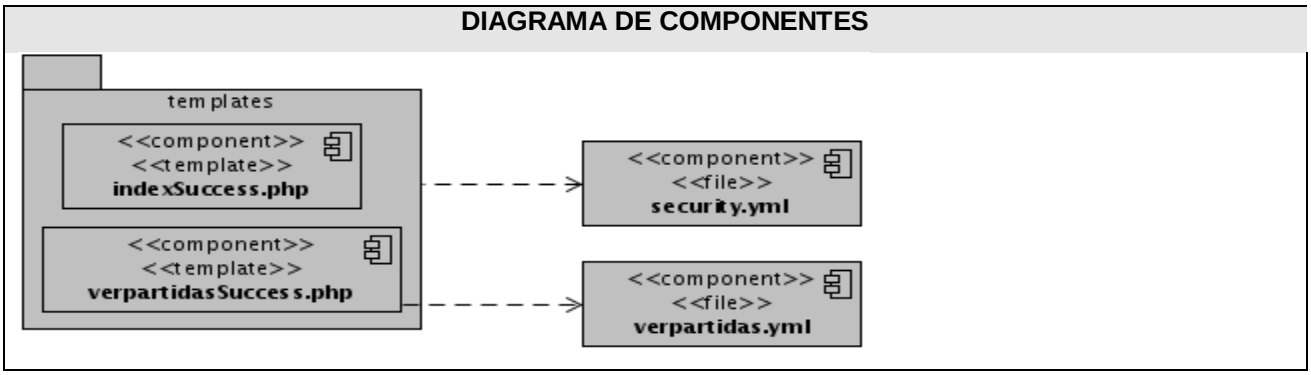

Figura 24 Diagrama de componentes del paquete "Consultar partidas"

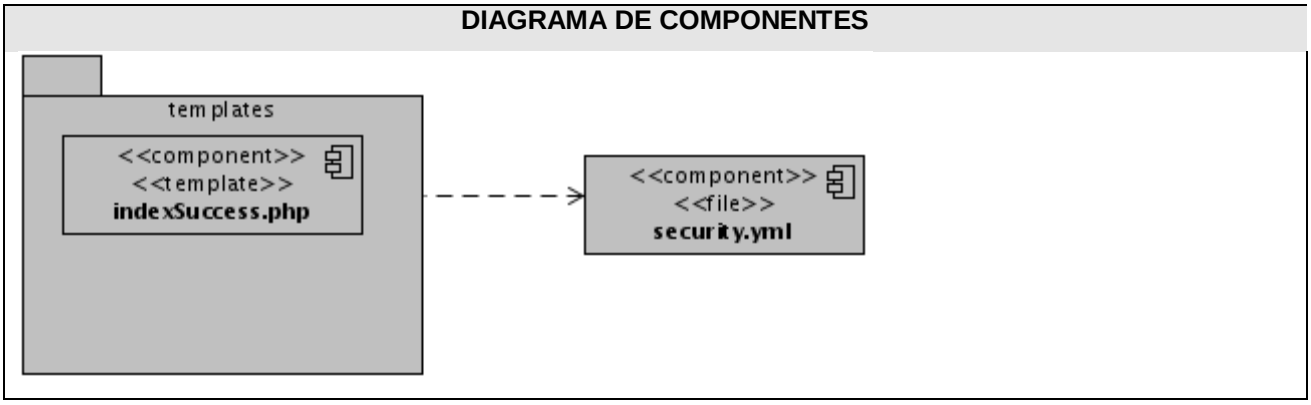

Figura 25 Diagrama de componentes del paquete "Principal"

4.4 CONCLUSIONES

En este capítulo fue presentado cómo está construido el sistema a partir de los diagramas de componentes de cada caso de uso, así como los diagramas de componentes de todo el sistema en general. También fue mostrado el diagrama de despliegue, el cual ilustra cuáles serán los nodos que serán usados para la implantación de la aplicación.

CAPÍTULO 5: ANÁLISIS DE LA FACTIBILIDAD

5.1 INTRODUCCIÓN

De la experiencia adquirida con el pasar de los años en la estimación de costos de Software, llega hasta nuestros días la necesidad real de estimar correctamente los esfuerzos de un proyecto, debido a que un elevado por ciento de ellos ya entregados suelen ser inutilizables, mientras que otro tanto se encuentra por encima del costo estimado o fuera de la fecha planificada.

El primer objetivo de la estimación es la determinación de la posibilidad de ejecutar el proyecto, o lo que es lo mismo, el estudio de la factibilidad, de acuerdo a las diferentes restricciones, las cuales están condicionadas por las características propias del proyecto y/o grupo de trabajo (organizativas, económicas, técnicas, tiempo).

5.2 ESTIMACIÓN DE ESFUERZOS BASADO EN CASOS DE USO

El modelo de casos de uso puede ser usado para la estimar el esfuerzo en el desarrollo de un proyecto de software. Existen elementos específicos que representan la diferencia entre un caso de uso y otro:

- La generalización entre los actores.
- Los casos de uso incluidos (include) y extendidos (extended).
- El nivel de detalle en las descripciones.

Estructura que define gradualmente cualquier estimación basada en casos de uso.
Si se quiere aplicar la estimación basada en casos de uso, se ha de tener un modelo donde se han identificado detalladamente cada caso de uso, específicamente, el número de transacciones, siendo esta un evento que ocurre entre un actor y el sistema.

A continuación pasaremos a analizar cada etapa del proceso, paso a paso.

Paso 1:
Determinar el número de actores del sistema. Almacenado en la variable UAW (Unadjusted Actor Weights) o Factor de peso de los actores sin ajustar.

Estos actores se encuentran clasificados en tres tipos:

- Simple: otro sistema que su sistema está unido a él por una interfaz de la programación de alguna clase, como una API (Application Programming Interface).
- Promedio: otro sistema que su sistema está unido a él por un protocolo o un texto basado en la interfaz del usuario. Estos actúan recíprocamente con el sistema a través de algunos protocolos (como HTTP, TCP/IP, etc.) o podría ser una tienda de datos.
- Complejo: una persona que actúa recíprocamente por medio de una interfaz gráfica. (Los usuarios finales).

Nuestro sistema cuenta con dos actores: usuario y administrador, estas son las personas que actúan recíprocamente con el sistema por medio de una interfaz gráfica. Por lo tanto nuestros actores están comprendidos en la clasificación: Complejo.

El siguiente paso de este proceso es pesar los actores identificados. Entonces se pasa a calcular el UAW total, contando el número de actores en cada categoría, anteriormente planteadas, multiplicando cada total por su factor de peso especificado, y adicionando todos estos productos.

Tabla 14 Factor de Peso de los Actores sin ajustar

Tipo de actor	Descripción	Factor	Número de actores	Resultado
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación(API, Application Programming Interface)	0	0	0
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	0	0	0

Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3	2	6
Total				6

Paso 2:

Determinar el número de casos de uso que tiene el sistema. Almacenado en la variable UUCW: (Unadjusted Use Case Weight) o Factor de peso de los casos de uso sin ajustar. Esta variable plantea que a los casos de uso se le asignan pesos de acuerdo al número de transacciones y/o escenarios. Los casos de uso se categorizan de acuerdo a su peso en: simple, promedio y complejo. Lo fundamental para eso es ver cuantas transacciones contiene un caso de uso. Siendo estas representadas por uno o más pasos del flujo de eventos principal del Caso de Uso.

El UUCW es calculado contando el número de casos de uso de cada categoría, multiplicando cada grupo de categoría por su peso, y sumando al final todos estos productos.

Tabla 15 Transacciones por CU

Nombre del CU	**Cantidad de transacciones**
Consultar las características de las partidas de un jugador	2
Seleccionar jugador	5
Mostrar comportamiento por ECO	1
Mostrar comportamiento por resultados	1
Mostrar comportamiento por color de las piezas	1
Gestionar partidas	8
Gestionar jugador	8
Gestionar torneo	8
Gestionar usuario	8

Permitir autentificarse	1
Registrarse	1

Tabla 16 Cálculo del UUCP

Tipo de CU	Descripción	Factor	Número de CU	Resultado
Simple	1 - 3 transacciones	2	9	18
Medio	4 - 7 transacciones	10	1	10
Complejo	8 - … transacciones	15	4	60
Total				88

El cálculo del UUCP: Puntos de casos de uso sin ajustar, se realiza sumando el valor encontrado en el paso 1 con el del paso 2.

$$UUCP = UAW + UUCW$$

$$UUCP = 6 + 88 = 94$$

Paso 3:

El cálculo del UUCP se basa en los factores técnicos, pero como en la realidad los casos de uso difieren mucho para dar factores tan estáticos, entonces se considera mayormente la complejidad técnica del sistema. Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema.

Se procede a llenar la tabla que se encuentra a continuación, asignándole valores entre 0 y 5 a cada factor, según su importancia en el sistema. Un factor de poca importancia, o lo que es lo mismo, que no es vital para el sistema, se le asigna el valor 0, mientras que a un factor de gran importancia se le asigna el valor 5.

Tabla 17 Factor de complejidad técnica

Factor	Descripción	Peso	Valor	Factor	Comentario
T1	Sistema Distribuido.	2	4	8	Sistema Central.

T2	Tiempo de la contestación y actuación de los objetivos.	1	4	4	Debe tener un alto tiempo de respuesta ya que se conectarán simultáneamente varios usuarios.
T3	Eficacia para el usuario final.	1	2	2	El usuario no requiere mucha preparación para usar el sistema.
T4	Proceso interno complejo.	1	1	1	No hay cálculos complejos.
T5	Reusabilidad del código.	1	4	4	El código puede ser utilizado para otras implementaciones
T6	Fácil de instalar.	0.5	2	1	Requiere la instalación de un servidor Web y un servidor de BD MySQL.
T7	Fácil de usar.	0.5	4	2	El sistema tendrá una interfaz amigable y fácil de usar.
T8	Portabilidad.	2	5	10	Podrá ser ejecutada en diferentes sistemas operativos
T9	Fácil de cambiar.	1	4	4	Costo de mantenimiento bajo.
T10	Concurrencia	1	0	0	No hay concurrencia.
T11	Incluye objetivos de seguridad especiales.	1	3	3	Los usuarios tendrán determinados permisos de acceso al sistema.
T12	Mantiene el acceso directo a terceras	1	5	5	Los usuarios tienen acceso directo.

	partes.				
T13	Se requiere facilidades de entrenamiento para usuarios especiales.	1	1	1	El personal no necesita alcanzar un alto nivel de capacitación.
Total (Tfactor):					**45**

TCF = 0.6 + 0.01 * Σ (Peso$_i$ * Valor$_i$)

TCF = 0.6 + 0.01 * 45

TCF= 1.05

Paso 4:

El entrenamiento y lograr habilidades en el personal involucrado en el desarrollo del sistema, también tienen un gran impacto en las estimaciones del tiempo. Esta variable se considera cuando se calcula el EF (Environment Factor) o Factor de Ambiente. Este se realiza de manera similar al cálculo de los factores técnicos como se muestra a continuación:

Tabla 18 Factor de Ambiente.

Factor	Descripción	Peso	Valor	Factor	Comentario
E1	Familiaridad con el proyecto que se ejecuta.	1.5	4	6	La mayoría del personal está familiarizado con el modelo.
E2	Experiencia en la aplicación.	0.5	2	1	Moderada experiencia en el trabajo con este tipo de sistema.
E3	Experiencia en la programación orientada a objetos.	1	4	4	La mayoría del personal tiene alguna experiencia.
E4	Capacidad del analista líder.	0.5	4	2	Altos conocimientos de Ingeniería de Software.

E5	Motivación	1	5	5	El equipo está altamente motivado.
E6	Requerimientos estables.	2	2	4	Se esperan algunos cambios.
E7	Personal de media jornada.	-1	4	-4	El equipo no trabaja a tiempo completo.
E8	Grado de dificultad del lenguaje de programación.	-1	3	-3	Se programará en php, y su complejidad es media.
Total (Efactor):				**15**	

EF = 1.4 - 0.03 * Σ (Peso$_i$ * Valor$_i$)

EF = 1.4-0.03*15

EF = 0,95

Paso 5:

Ahora se puede calcular el AUCP = UCP (Adjusted Use Case Points) o Puntos de casos de uso ajustados, de la siguiente manera:

UCP = UUCP x TCF x EF

UCP = 94 x 1.05 x 0.95

UCP = 93.77

Ya teniendo todas estas variables, finalmente arribamos al último esfuerzo. Para la estimación del Esfuerzo en horas-hombre (E), debe multiplicarse el UCP ajustado con un Factor de conversión (CF), el cual denota las personas-horas en esfuerzo.

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)

Total $_{EF}$ = Cant EF < 3 (entre E1, E6) + Cant EF > 3 (entre E7, E8)

Como Total $_{EF}$ =2+0

Total $_{EF}$ = 2

CF = 20 horas-hombre (porque Total $_{EF}$ ≤ 2)

Un esfuerzo estimado en horas-hombre vienen dado por:

E = **UCP** x **CF** = 93.77x 20 = 1875.3 horas-hombre

Debe tenerse en cuenta que este método proporciona una estimación contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso. Para una estimación más completa de la duración total del sistema, 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 del sistema. Para ello se debe tener en cuenta el criterio de distribuir el esfuerzo entre las diferentes actividades de un sistema según la siguiente aproximación:

Tabla 19 Distribución del esfuerzo entre las diferentes actividades del sistema

Actividad	Porcentaje
Análisis	10 %
Diseño	20 %
Implementación	40 %
Pruebas	15 %
Sobrecarga(otras actividades)	15 %

Con este criterio, y tomando como entrada la estimación de tiempo calculada a partir de los Puntos de Casos de Uso, se pueden calcular las demás estimaciones para obtener la duración total del sistema.

Tabla 20 Cálculo del esfuerzo de las diferentes actividades del sistema

Actividad	Porcentaje	Horas-Hombre
Análisis	10 %	468.82

Diseño	20 %	937,65
Implementación	40 %	1875.3
Pruebas	15 %	704.24
Sobrecarga(otras actividades)	15 %	704.24
Total	100%	4688.25

Teniendo en cuanta que un día tiene 8 horas y los días laborables del mes son aproximadamente 24, entonces una persona en un mes trabaja 192 horas por tanto:

- E_T = 24.41 mes-hombre
- Salario promedio de un desarrollador es $ 225
- Cantidad de hombres: 2
- Tiempo de desarrollo a emplear es: 24.41 /2 = 12,2 meses
- Costo Total(Salario): $5480

5.3 BENEFICIOS TANGIBLES E INTANGIBLES

La implantación del sistema propuesto trae consigo una serie de beneficios fundamentalmente intangibles para la cátedra de Ajedrez de la UCI, debido a que permitirá realizar análisis estadísticos a partidas de ajedrez, permitiendo que los aficionados puedan analizar cómo juega determinado jugador, así como las partidas de este. Este sistema posee una interfaz gráfica sencilla y amigable al usuario, en el cual la información puede buscarse rápida y eficientemente.

Entre los beneficios tangibles podemos mencionar la obtención del sistema, con la finalidad de que los aficionados al Ajedrez sean capaces de auto prepararse a partir del análisis de las partidas de determinados jugadores. El usuario puede obtener reportes acerca de partidas de Ajedrez de forma rápida y segura.

5.4 ANÁLISIS DE COSTOS Y BENEFICIOS

Desarrollar un producto informático cuesta, justificar entonces su desarrollo depende de los beneficios que reportaría su implantación y uso. Este sistema ha sido desarrollado utilizando software libre con el objetivo de que no fuera necesario hacer gastos de licencia ni para su implementación ni para su implantación. Como pudo verse anteriormente el producto tiene un costo de: $5480, siendo importante viéndolo desde el punto de vista de que no sería necesario importarlo de otro país, al ser producido en nuestro país.

Es factible desarrollar una aplicación web para automatizar el proceso de gestión estadística de partidas de Ajedrez, ya que trae consigo varios beneficios como por ejemplo permitir la autopreparación de los usuarios a partir del análisis de las partidas de otros jugadores y las suyas propias, así como darle la posibilidad a los entrenadores de analizar el rendimiento de sus jugadores.

5.5 CONCLUSIONES

En este capítulo se estimó el esfuerzo total para la realización del sistema concluyendo de este estudio el tiempo total de desarrollo, dado por la suma de los tiempos empleados en cada una de sus fases teniendo en cuenta que participan 2 personas.

Se señalaron los beneficios tangibles e intangibles de su puesta en marcha. Se realizó un análisis costo-beneficio que permitió confirmar la factibilidad del sistema propuesto.

CONCLUSIONES

Como resultado de este trabajo de este trabajo de diploma se obtuvo un sistema informático que permite realizar análisis estadísticos de partidas de Ajedrez. Para desarrollar el presente software se hizo un estudio de la situación existente en la actualidad, identificando las mejoras que nuestra propuesta brindaría. Se analizaron las posibles herramientas a utilizar, escogiendo las más acordes para desarrollar el proyecto con la calidad requerida de acuerdo a las disposiciones existentes sobre el uso de software libre.

RECOMENDACIONES

A partir del desarrollo de este trabajo se decidió realizar varias recomendaciones a aquellos que le darán continuación, entre las que se encuentran las siguientes:

- ✓ Poner a prueba el sistema durante un período de tiempo significativo, para comprobar su desempeño y que las funcionalidades del sistema se correspondan con la actividad que se está realizando.
- ✓ Continuar el estudio con el objetivo de añadir nuevas funcionalidades.
- ✓ Incorporar más información a los reportes que se generan, con el objetivo de que estos tengan una mayor utilidad.

REFERENCIAS BIBLIOGRÁFICAS

1. Aplicación Web. 2008 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

2. Programación para todos. 2006 [citado febrero 2008]; Disponible en: http://modulado.blogspot.com/2006_11_01_archive.html

3. Historia de los lenguajes de programación [citado febrero 2008]; Disponible en: http://www.erick.cibercalli.com/showpost?postid=17

4. PHP. 2008 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/PHP#Ventajas_de_PHP

5. Lenguaje HTML.2008 [citado febrero 2008]; Disponible en: http://www.desarrolloweb.com/articulos/711.php

6. Servidor Web.2008 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/Servidor_web

7. Lighttpd. 2007 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/Lighttpd

8. Metodologías de Desarrollo de Software. 2004 [citado febrero 2008]; Disponible en: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

9. MENDOZA, María. Metodologías de Desarrollo de Software.2004 [citado febrero 2008]; Disponible en: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

10. SCHMULLER, Joseph. Aprendiendo UML en 24 horas. Prentice Hall, 2003.

11. OMG Unified Modelling Language Specification. OMG, 2003.

REFERENCIAS BIBLIOGRÁFICAS

12. Herramientas CASE. 2007 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/CASE

13. Editores para PHP, Zend Studio. 2007 [citado febrero 2008]; Disponible en: http://www.adrformacion.com/cursos/php/leccion1/tutorial3.html

14. Creación de gráficas en PHP con JpGraph.2005 [citado febrero 2008]; Disponible en: http://www.desarrolloweb.com/articulos/1987.php

15. Arquitectura de software.2008 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/Arquitectura_de_software

16. Modelo Vista Controlador (MVC). 2008 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

BIBLIOGRAFÍA

1. CATALANI, Exequiel. ARQUITECTURA Modelo/Vista/Controlador. 2007 [citado febrero 2008]; Disponible en: http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/

2. Eclipse (software). 2008 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/Eclipse_(software)

3. EGUÍLUZ, Javier. Introducción a AJAX. 2008 [citado mayo 2008]; Disponible en: http://librosweb.es/ajax/

4. EGUÍLUZ, Javier. Introducción a CSS. 2008 [citado mayo 2008]; Disponible en: http://librosweb.es/css/

5. EGUÍLUZ, Javier. Introducción a JavaScript. 2008 [citado mayo 2008]; Disponible en: http://librosweb.es/javascript/

6. LAGO, Ramiro. Patrón "Modelo-Vista-Controlador". 2007 [citado febrero 2008]; Disponible en: http://www.proactiva-calidad.com/java/patrones/mvc.html

7. MySql con clase. 2004 [citado febrero 2008]; Disponible en: http://mysql.conclase.net/curso/index.php

8. PostgreSQL. 2008 [citado febrero 2008]; Disponible en: http://es.wikipedia.org/wiki/PostgreSQL

9. ¿Qué es un Sistema Gestor de Bases de Datos o SGBD? .2007 [citado febrero 2008]; Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/

10. RUMBAUGH, J., JACOBSON, I., BOOCH, G.El Lenguaje Unificado de Modelado. Manual de Referencia. Addison-Wesley, 2000.

11. RUMBAUGH, J., JACOBSON, I., BOOCH, G. El Proceso Unificado de Desarrollo de software. Addison-Wesley, 2000.

BIBLIOGRAFÍA

12. The Askeet Tutorial. 2008 [citado mayo 2008]; Disponible en: http://www.symfony-project.org/askeet/1_0/en/

13. Una Introducción a APACHE. 2006 [citado febrero 2008]; Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro/

14. Visual Paradigm for UML (ME). 2007 [citado febrero 2008]; Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/

15. ZANINOTTO, François. Symfony, la guía definitiva. 2008 [citado mayo 2008]; Disponible en: http://librosweb.es/symfony

16. Zend Framework. 2008 [citado febrero 2008]; Disponible en: http://en.wikipedia.org/wiki/Zend_Framework

GLOSARIO

- Ajedrecista: Un jugador que juega o practica el Ajedrez.
- Ajedrez: Juego de mesa jugado entre dos personas sobre un tablero de 64 casillas alternadas de color claro y oscuro (blancas y negras). Cada jugador dispone de 16 piezas de Ajedrez blancas y negras respectivamente y la finalidad del juego es de dar Jaque Mate al Rey del jugador contrario.
- Apertura: La secuencia de una serie de movidas realizadas por ambos jugadores al principio de un juego de Ajedrez. Existe una gran variedad de Aperturas pero el propósito es siempre el mismo: asegurar que un lado, o el otro, o ambos, que los mejores movimientos sean hechos al principio de una partida. La intención de la apertura es desarrollar rápidamente y correctamente las piezas del Ajedrez según un plan cuyo primer objetivo es ocupar y dominar el centro del tablero.
- Blancas: Piezas de Ajedrez de color claro y que por regla general inician una partida. Esto constituye una ventaja inicial sobre las negras, pues permite tomar la iniciativa y, en alguna medida, marcar la pauta de la partida en las dos primeras etapas del juego.
- CASE: *Computer Aided Software Engineering.*
- CUN: *Caso de uso del negocio*.
- CUS: *Caso de uso del sistema*.
- Defensa: Jugada o conjunto de jugadas del bando negro como réplica a la apertura. La defensa puede ser simétrica o asimétrica, según reproduzca o no los movimientos del blanco.
- ECO (Encyclopaedia of Chess Openings): El código ECO es un sistema de clasificar todas las aperturas en ajedrez ideado por la Enciclopedia de aperturas (Encyclopaedia of Chess Openings). Clasifica todas las jugadas posibles en cinco letras: A, B, C, D y E siempre en

mayúscula. Cada una de estas letras va del 00 al 99 en la que están las distintas variantes. En total son 500 aperturas codificadas.

- Elo, sistema: Un sistema de valoración de jugadores de Ajedrez que fue desarrollado por el Profesor Arpad Elo (1903 - 1993) de Milwaukee. A un jugador se le asigna una puntuación inicial que aumenta o disminuye de acuerdo a los resultados obtenidos en partidas oficiales.

- Estadística: Es una rama de la matemática que se refiere a la recolección, estudio e interpretación de los datos obtenidos en un estudio.

- FIDE: Fedération Internationale des Échecs, máximo organismo del Ajedrez mundial creado en el año de 1924 en París. Entre las facultades de la FIDE están todo lo relacionado a los títulos mundiales.

- HTML: *Hypertext Markup Language.* Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986. Es un lenguaje de marcas. Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que se pueden controlar desde cualquier editor ASCII.

- HTTP: *HyperText Transfer Protocol.* Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas Web.

- Herramientas CASE: Herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.

- Hardware: Componentes electrónicos, tarjetas, periféricos y equipo que conforman un sistema de computación; se distinguen de los programas (software) porque son tangibles.

- 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.

GLOSARIO

- Java: Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los 90.
- Jugador: Cada uno de los dos contendientes en una partida.
- Microsoft: Compañía que manufactura los sistemas de operación DOS y Windows.
- MySQL: Es un sistema de gestión de bases de datos relacional que cuentan con todas las características de un motor de BD comercial: transacciones atómicas, triggers, replicación, llaves foráneas entre otras. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar.
- Negras: Piezas de Ajedrez de color oscuro y que por regla general son las piezas del segundo jugador de un juego. Es muy común en los comentarios de Ajedrez sustituir el nombre de los jugadores por el color de las piezas que conducen.
- Partida de Ajedrez: Serie de jugadas o movimientos efectuados por los dos jugadores de Ajedrez con la finalidad de dar jaquemate al Rey del oponente.
- PHP: *PHP: Hypertext Preprocessor*. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas. Con PHP se pueden combinar páginas HTML y scripts. Con el objetivo de crear aplicaciones potentes.
- Pop up: Un pop-up o ventana pop-up, es una ventana nueva que es mostrada enfrente de de otra ventana o sitio Web ya existente.
- PostgreSQL: es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre.
- RUP: *Rational Unified Process* (Proceso Unificado de desarrollo). Metodología para el desarrollo de Software.
- Software: Programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.

GLOSARIO

- 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.
- Torneo: Encuentro de Ajedrez en la que intervienen varios jugadores.
- UCI: *Universidad de las Ciencias Informáticas*.
- UML: *Unified Modelling Language*. Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.

1