Universidad de las Ciencias Informáticas

Facultad 8

Ingeniería de Requerimientos aplicados a la plataforma de Infodrez 2.0

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es):

Ivette Catalá Matienzo

Tutor(es):

Ing. Alison Muñoz Capote

Proyecto al que pertenece:

Infodrez

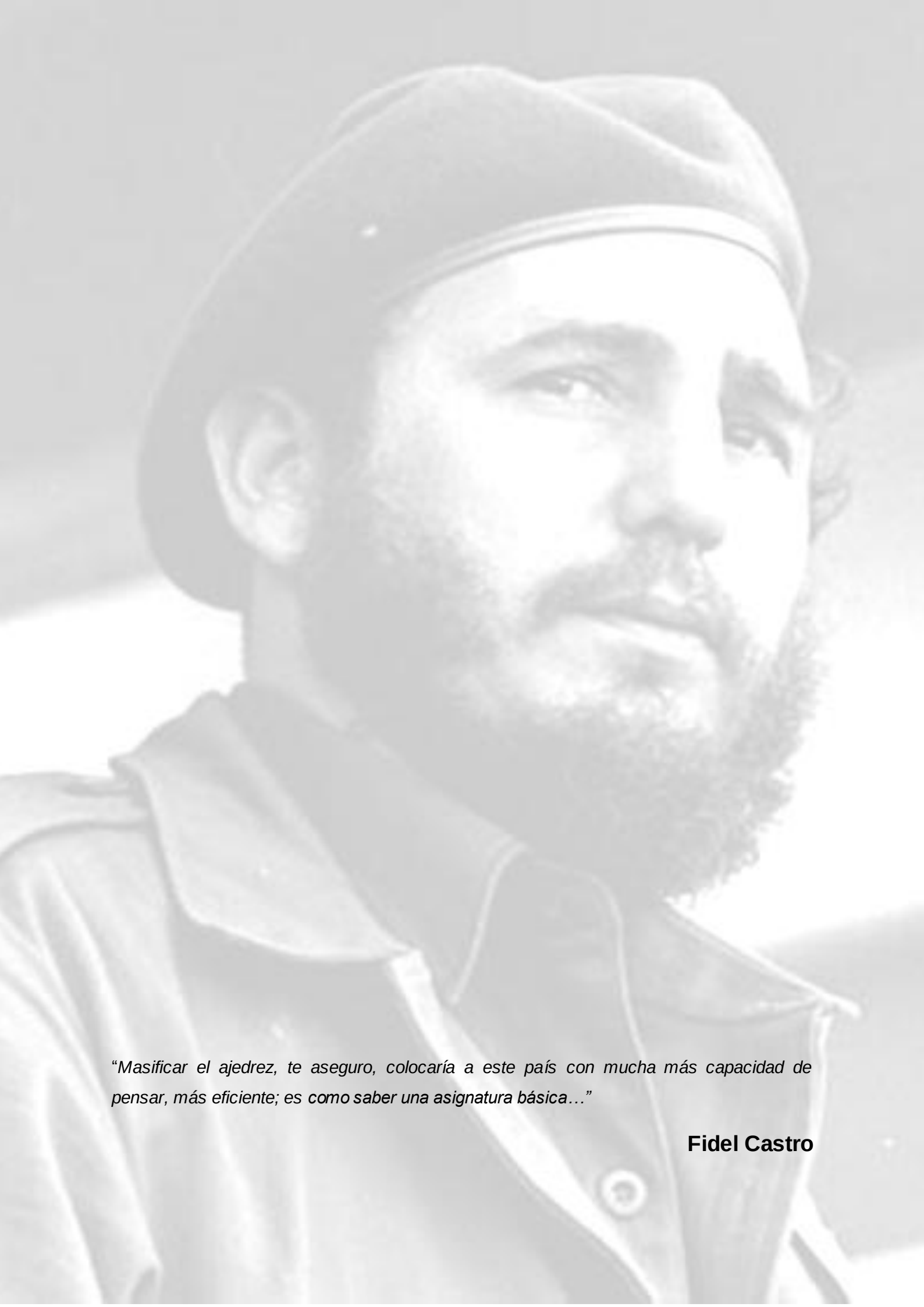
DECLARACIÓN DE AUDITORIA

Declaro que soy el único autor del trabajo "Ingeniería de Requerimientos aplicados a la plataforma Infodrez 2.0" y autorizo a la Facultad 8 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Autor: Tutor:

_____ _____

Ivette Catalá Matienzo Ing. Alison Muñoz Capote



"*Masificar el ajedrez, te aseguro, colocaría a este país con mucha más capacidad de pensar, más eficiente; es como saber una asignatura básica…*"

Fidel Castro

AGRADECIMIENTOS

A mi mamá, pues a ella le debo "todo", por su apoyo, su sacrificio, por ser un ejemplo a seguir, además le agradezco por buscarme el mejor padre del mundo.

A mi papá por estar ahí siempre para mí.

A mi hermano que es lo más grande que tengo en la vida.

En fin agradezco a mis padres y hermano por brindarme un hogar feliz, por ser mi sostén, por confiar y tener fe en mi, por su paciencia, dedicación y por quererme más de lo que se pueda escribir en estas páginas.

A mi segunda madre Celeida, por apoyarme y lograr que me sintiera parte de su familia.

A toda mi familia en general, a mis abuelos, a mis tíos, y primos, ¿Qué sería yo sin ellos?

A mi tutor Alison por ser paciente conmigo, aunque muchas veces tuve que ser muy paciente con él, pero sin su apoyo esta tesis no hubiera llegado a su fin.

A mi grupo de primer año ellos fueron los primeros en alentarme a seguir cuando pensé que no podría continuar.

A todas mis amistades y compañeros que sin proponérselos estuvieron ahí para mí.

A los muchachos del ajedrez pues con ellos pasé muy buenos momentos.

A los profesores que de una forma u otra contribuyeron en mi formación profesional.

A Reimis, Neima, Yanay, Yarsel, Maritza(la bb), Katy (pinkgirl), la vieja Mayde, Ridercito, Yunior, Iralis, Lili, Vivi (la foca), Yoney, Martha, Camilo, Inés, Lucia, a todos ellos gracias por soportarme tanto tiempo, por alarme las orejas cuando lo merecía, por ayudarme a salir de algún problema cuando lo necesitaba, por darme muy buenos consejos, en fin por ser mis amigos.

A la Revolución y la universidad por brindarme esta oportunidad.

A todos muchas gracias……..……

Resumen

Infodrez ha jugado un papel muy importante en la informatización del ajedrez en la Universidad de las Ciencias Informáticas (UCI). Ha desarrollado productos en las principales áreas de este deporte que satisfacen las necesidades de los especialistas de la cátedra Remberto A. Fernández de la UCI. Estos productos presentan algunas deficiencias, aunque se considera que la mayor de ellas es el hecho de estar separados y se dificulta la consulta de toda la información simultáneamente. En el presente trabajo se dio solución a esta problemática, con el diseño de la segunda versión de la plataforma Infodrez. Esta propuesta abarca principalmente funcionalidades presentes en los módulos de Juego Online y Ajedrez por Correspondencia. En el desarrollo de esta propuesta se utilizó una de las metodologías tradicionales más usadas actualmente en el mundo, el Proceso Unificado de Desarrollo (RUP) unido al lenguaje de modelado UML. Durante el levantamiento de requisitos se definen algunas técnicas para la obtención y especificación de los requerimientos, continuando con su validación a través de prototipos presentados al usuario antes de entrar a modelar el sistema. Durante el análisis y el diseño se aplicaron los patrones de asignación de responsabilidades GRASP así como el patrón arquitectónico MVC. En la modelación de los artefactos generados durante el paso por estos flujos de trabajo se utiliza la herramienta Visual Paradigm, propuesta por el proyecto.

Introducción

Cuba se caracteriza por ser una potencia deportiva y el ajedrez no es la excepción. Es un deporte muy seguido por la afición cubana y su calidad lo demuestran los resultados alcanzados en Olimpiadas y de forma individual por algunos de sus jugadores. La informática ha tenido su impacto positivo en el desarrollo de este deporte, pero no ha tenido una total influencia debido a factores como son la privatización de los software, además de que los servicios los brindan aplicaciones publicadas en Internet y los usuarios en nuestro país no tienen casi o ningún acceso a ella desde la red nacional.

Motivado por esta problemática, la UCI, a través del proyecto Infodrez, se trazó como meta la informatización del ajedrez cubano, a partir de la creación de aplicaciones que se utilizarán en las diversas áreas de este deporte y enfocada a las problemáticas reales de este deporte en nuestro país. Las propuestas de productos identificados en el curso 2007-2008 fueron:

- Desarrollo de la versión 2.0 del módulo Juego Online de Infodrez.
- Concepción del módulo de Ajedrez por Correspondencia de Infodrez
- Confección y desarrollo del módulo Gestor de Estadísticas del proyecto Infodrez.
- Gestionar el Componente del Rating ELO.

Después de haber efectuado revisiones en cada una de las propuestas de productos del proyecto Infodrez, se identificaron las siguientes problemáticas en cada uno de los módulos:

- Réplica de reportes.
- Información no centralizada.
- Información no estandarizada.
- Réplica de código.

Teniendo en cuenta estas deficiencias y en vista a que el proyecto desea entrar en competencia en el mercado internacional, se hace necesario crear un producto que cubra las expectativas de los usuarios y sus funcionalidades tengan la eficiencia de los más utilizados en la actualidad. Para ello el proyecto identifica la inexistencia de un sistema que integre la gestión de las principales áreas de ajedrez. El presente trabajo abordará las áreas referentes a Juego Online y Ajedrez por Correspondencia los cuales no fueron abarcados en la primera propuesta de este tipo.

Por tal razón el **Problema Científico** de la investigación se puede formular de la siguiente manera: ¿Cómo realizar la modelación para integrar a la plataforma de Infodrez los módulos Juego Online y Ajedrez por Correspondencia aplicando la Ingeniería de Requerimientos definida para el proyecto Infodrez?

Como **Objetivo General** la realización del análisis y diseño de un sistema integrado con los módulos Juego Online y Ajedrez por Correspondencia.

Considerando como **Objeto de Estudio** el proceso de análisis y diseño de software de gestión y tomando como **Campo de Acción** la modelación de los módulos Juego Online y Ajedrez por Correspondencia del proyecto Infodrez.

Se plantean como **Objetivos específicos:**

1. Analizar el estado de los módulos Juego Online y Ajedrez por Correspondencia.
2. Identificar las deficiencias que presenta cada uno por separado.
3. Identificar ventajas que debe tener el sistema propuesto sobre los sistemas existentes.
4. Elaborar una propuesta de sistema de la segunda versión de la plataforma Infodrez con los módulos unificados.

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

1. Análisis de los modelos propuestos por las tesis anteriores.
2. Análisis de los sistemas similares existentes.
3. Entrevistas a especialistas de los sistemas existentes en el proyecto.
4. Refinación de los requisitos de cada uno de los sistemas existentes en el proyecto.
5. Elaboración del modelo propuesto.

Como **Idea a defender** se puede modelar un sistema integrado, para crear una herramienta tal que permita la gestión de los módulos del proyecto Infodrez.

Los **Resultados esperados** son: El análisis y diseño de la versión 2.0 de la plataforma de Infodrez incorporando los módulos de Juego Online y Ajedrez por Correspondencia.

En la elaboración del presente trabajo se tuvo en cuenta como método principal la entrevista la cual es una conversación planificada con un especialista para obtener información. Se usó para el conocimiento de los fenómenos características del posible sistema a desarrollar. De igual forma el empleo del método histórico-lógico permitió acceder al conocimiento del

estado actual de la problemática, sus fuentes, en tanto aseguró, la formulación más adecuada de los conceptos teóricos que sustentan la investigación .El analítico-sintético posibilitó el procesamiento y decantación de la información encontrada en las fuentes consultadas. Es preciso señalar que como método empírico más empleado resultó específicamente la observación, el que facilitó un entendimiento cabal de la situación existente así como la formulación de las distintas vías y variables de solución.

Capítulo 1. Fundamentación Teórica

1.1 Introducción.

A lo largo de este capítulo se realizó una descripción de los aspectos que se tienen en cuenta para modelar el sistema, se efectúo además un análisis de la metodología de desarrollo de software a utilizar, así como los lenguajes de modelado, herramientas CASE, técnicas de captura requisitos, además de los gestores de Base Datos. Se realizó una búsqueda en internet de sistemas similares existentes y se consultaron especialistas del tema y no se encontró evidencia de ninguno.

1.2 Desarrollo

1.2.1 Metodologías de desarrollo.

Hoy en día el desarrollo del software requiere mayor esfuerzo debido a que los clientes piden productos más grandes y con mayor calidad, por lo que se hace necesario el uso de una metodología de desarrollo que guíe este proceso y garantice la satisfacción del cliente. No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc.).

La metodología de desarrollo de software en Ingeniería de Software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Es definido por Mario Piattini Velthuis como un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software (Piattini, 1996). Piattini expresa que "una metodología representa el camino para desarrollar software de una manera sistemática".

Estas se pueden enmarcar en dos grupos: tradicionales (pesadas) y ágiles (ligeras). Las metodologías tradicionales se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir así como las herramientas y notaciones que se usarán. Las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Las metodologías de desarrollo más conocidas en la actualidad son: Programación Extrema (XP), Proceso Unificado de Desarrollo de Software (RUP) y Scrum.

1.2.1.1 Metodología RUP

El Proceso Unificado de Desarrollo de Software (*Rational Unified Process)* en inglés, conocido como RUP, es el resultado de varios años de desarrollo y uso práctico, unifica los mejores elementos de las metodologías anteriores. Es un proceso de desarrollo de software que unido al Lenguaje Unificado de Modelado UML, constituye una de las metodologías más utilizadas para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP se divide en cuatro fases:

- Fase de Inicio: se define la idea, la visión del producto, como se enmarca en el negocio y el alcance del proyecto.
- Fase de Elaboración: se planifican las actividades necesarias y los recursos requeridos, especificando las características y el diseño de la arquitectura.
- Fase de Construcción: se construye el producto, la arquitectura y los planes, hasta que el producto está listo para ser enviado a la comunidad de usuarios.
- Fase de Transición*:* se realiza la transición del producto a los usuarios, lo cual incluye: manufactura, envío, entrenamiento, soporte y mantenimiento del producto, hasta que el cliente esté satisfecho.

El proceso propuesto por RUP posee dos dimensiones: la primera, representa el aspecto dinámico del proceso, y está expresado en términos de ciclos, fases, iteraciones e hitos, la segunda, representa el aspecto estático, que se describe en términos de componentes, actividades, flujos de trabajo, artefactos y actores como se muestra en la figura 1.1.

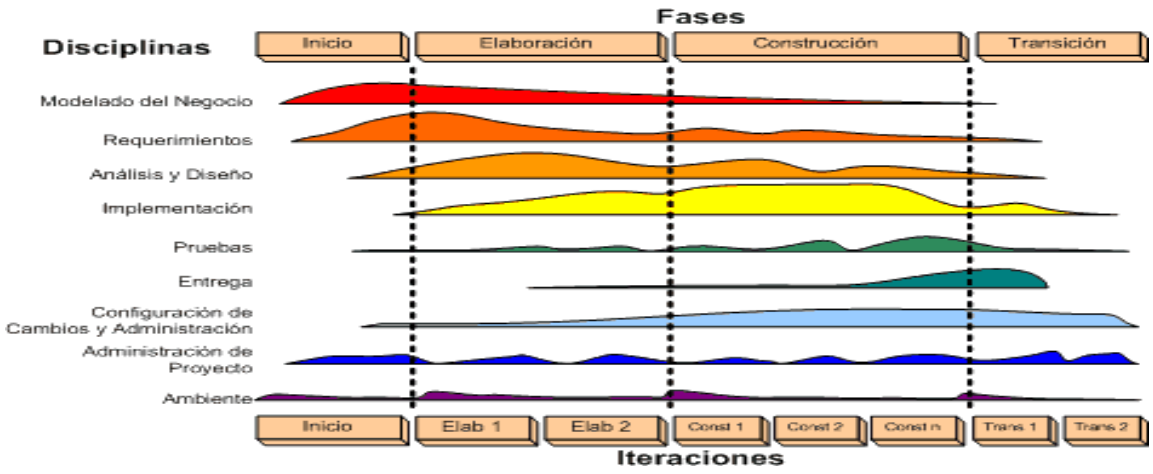


Figura1.1: RUP en dos dimensiones

La realización de las cuatro fases de RUP produce una generación del producto. Cada fase tiene una o más iteraciones de todos los flujos y finaliza en un hito. Al finalizar cada fase, en estos hitos ha de cumplirse que:

- Fase de Inicio: Objetivos (Visión).
- Fase de Elaboración: Arquitectura.
- Fase de Construcción: Capacidad Operacional Inicial.
- Fase de Transición: Liberación del Producto.

RUP cuenta con 9 flujos de trabajo, los 6 primeros de ingeniería y los 3 últimos de apoyo:

- 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, la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba (Testeo): Busca los defectos a lo largo del ciclo de vida.
- Instalación: Produce liberación 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.

RUP en su modelación define como sus principales elementos:

- Trabajadores (“quién”): Define 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.

El ciclo de vida de RUP se caracteriza por:

- Dirigido por casos de uso**:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso
- Centrado en la arquitectura**:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Iterativo e Incremental**:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

En resumen, las metas de RUP según Kruchten son:

- Asegurar la producción de un software de alta calidad que reúna las necesidades de los usuarios finales dentro de un plan y un presupuesto predecible.
- Proveer un enfoque disciplinado para asignar tareas y responsabilidades dentro del desarrollo del sistema.
- Proveer un camino metódico, sistemático para desarrollar, diseñar y validar una arquitectura.

- Reducir en gran medida el riesgo que representa la construcción de sistemas complejos, porque evoluciona de forma incremental partiendo de sistemas más pequeños en los que ya se tiene confianza.

1.2.1.2 Metodología a utilizar.

Se decidió aplicar la metodología RUP *(Proceso Unificado de Desarrollo de Software)* ya que esta metodología es la que mejor se adapta al proceso, debido a que como metodología tradicional realiza una fuerte planificación durante todo el proceso de desarrollo del software realizando una intensa etapa de análisis y diseño antes de la construcción del sistema. Además que en la universidad se encuentran grandes volúmenes de información y experiencia acerca de la misma. Esta metodología fue escogida de antemano por el proyecto, la justificación de la misma se puede encontrar en el documento Plan de Desarrollo de Software del proyecto Infodrez (1).

1.2.2 Lenguajes de Modelado.

Un modelo es una descripción de (parte de) un sistema, descrito en un lenguaje bien definido. Un lenguaje bien definido es un lenguaje con una sintaxis y semántica precisa, y que puede ser interpretado y manipulado por un ordenador.

Los modelos proporcionan un mayor nivel de abstracción, permitiendo trabajar con sistemas mayores y más complejos, y facilitando el proceso de codificación e implementación del sistema de forma distribuida y en distintas plataformas.

1.2.2.1 Lenguaje UML 2.0

El desarrollo de las metodologías de análisis y diseño orientados a objetos trajo consigo la existencia de diversas notaciones y propuestas, entre las que cabe destacar la metodología de Booch, la Técnica de Modela de Objetos (OMT) de Rumbaugh y Software Orientados a Objetos de Ingeniería (OOSE) de Ivar Jacobson. Ante el problema de la proliferación de notaciones distintas, los autores más relevantes unieron esfuerzos para producir una notación unificada, y el resultado de esto fue el Lenguaje Unificado de Modelado o Unified Modeling Language, UML. Entre los lenguajes de modelado que define OMG (Object Management Group) el más conocido y usado es el UML.

Esta no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas orientada a objetos. Los diagramas se realizan con la finalidad de presentar diversas perspectivas de un

sistema, a los que se les denominan modelos, estos describen lo que supuestamente hará un sistema, pero no dicen como implementarlo.

UML posibilita el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Se ha convertido en un estándar de facto en la industria, ya que permite visualizar, especificar, construir y documentar los elementos que forman un sistema de software orientado a objetos (Jacobson, y otros, 2000). A continuación se definen cada una de las operaciones que permite UML.

- Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado, sirviendo para su futura revisión.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Está formado por tres clases de bloques de construcción. El primero de ellos son los Elementos, es decir, las abstracciones de cosas reales o ficticias (como objetos y acciones); luego se encuentran las Relaciones, esto es la relación entre los elementos (la cual puede ser de dependencia, asociación, generalización, etc.); finalmente, se encuentran los Diagramas, que son colecciones de elementos con sus relaciones.

1.2.2.2 Selección del Lenguaje de Modelado.

Se decide utilizar como lenguaje de modelado el UML, debido a que este es uno de los más usados en la modelación de los artefactos que se van generando en el proceso de desarrollo del software. Al ser creado por los autores de la metodología RUP, este lenguaje da soporte a la misma, por lo que cubre todas las necesidades de esta metodología formando un buen complemento a la hora de desarrollar un producto. Este lenguaje fue escogido de antemano por el proyecto, la justificación del mismo se puede encontrar en el documento Plan de Desarrollo de Software del proyecto Infodrez (1).

1.2.3 Herramientas CASE

Las herramientas CASE (**C**omputer **A**ided **S**oftware **E**ngineering, Ingeniería de Software Asistida por Ordenador) sirven de ayuda a lo largo de todo el ciclo de vida de desarrollo del software, permitiendo realizar el diseño del proyecto, la compilación automática, documentación o detección de errores entre otras funcionalidades.

Estas herramientas aumentan la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Cuentan también con una credibilidad y exactitud que tienen un reconocimiento universal, siendo usadas por cualquier desarrollador y /o programador que busca un resultado óptimo y eficiente, pero sobre todo, que busca esa minuciosidad necesaria de los procesos y entre los procesos. Algunos ejemplos de aplicaciones CASE son: Visual Paradigm por UML, Rational ClearCASE, Rational Rose.

1.2.3.1 Herramientas CASE, Visual Paradigm 6.0.

Visual Paradigm es una herramienta CASE que utiliza el UML como lenguaje de modelado. Cubre el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, realizar ingeniería tanto directa como inversa y genera la documentación del proyecto automáticamente en varios formatos como PDF, HTML y MS Word.

Es una herramienta colaborativa, soporta múltiples usuarios trabajando sobre el mismo proyecto. Proporciona abundantes tutoriales, demostraciones interactivas y proyectos de UML. Combina el modelado UML, código de ingeniería avanzada y una excelente interoperabilidad en una única plataforma de desarrollo.

VP-UML (Visual Paradigm por UML) ayuda a construir aplicaciones en menor tiempo, de una mejor forma y más económica. Se debe destacar su robustez, usabilidad y portabilidad.

Características de Visual Paradigm.

- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Interoperabilidad con modelos UML2 (meta-modelos UML 2.x para plataforma Eclipse) a través de XML.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.

- Diagramas EJB - Visualización de sistemas Enterprise JavaBeans (EJB).
- Diagramas de flujo de datos.
- Soporte ORM - Generación de objetos Java desde la base de datos
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XMI.
- Soporta múltiples plataformas.

1.2.3.2 Selección de Herramientas CASE

Visual Paradigm: se decidió seleccionar esta herramienta debido a que se pueden realizar las modelaciones del sistema que se propondrá de manera cómoda y es soportada por la distribución Debian del sistema operativo Linux, que es la plataforma que soporta el ambiente de desarrollo del proyecto Infodrez. Esta herramienta fue escogida de antemano por el proyecto, la justificación de la misma se puede encontrar en el documento Plan de Desarrollo de Software del proyecto Infodrez (1).

1.2.4 Técnicas para la recopilación de requisitos

El uso de las herramientas para la recopilación de requisitos es alentado tanto para mejorar la calidad como la productividad en un proceso de desarrollo de software. Aproximadamente el 60-70% de los proyectos informáticos fallan por la pobre recopilación, análisis y gestión de requisitos.

Una herramienta para la recopilación de requisitos debe ser capaz de realizar algunas funciones como el registro, edición y rastreo de requisitos, así como la generación de informes.

Entre las técnicas de recopilación de requisitos más utilizadas se encuentran las mencionadas a continuación:

1.2.4.1 Tormenta de Ideas.

Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Puede ayudar a generar una gran variedad de vistas de problemas y a formularlo de diferentes formas, sobre todo al comienzo de captura. Se requiere participación intensiva del analista.

1.2.4.2 Técnicas para Facilitar las Especificaciones de una Aplicación (TFEA)

Estas son una variación de las entrevistas buscando identificar el problema, proponer elementos de solución, negociar diferentes enfoques y especificar un conjunto preliminar de requisitos de la solución. A pesar de ir un paso más allá de las entrevistas convencionales, precisan aún de una alta participación del analista.

1.2.4.3 Entrevistas

Es la más tradicional de las técnicas de obtención y consiste en reuniones analista-interesado en las cuales se suceden preguntas y respuestas para extraer el dominio de la aplicación (Goguen y Linde, 1993). En Pressman (2005) se presentan conjuntos de preguntas que se pueden utilizar en el desarrollo de esta técnica, que tiene una alta participación del analista y se realiza en conjunto con otras técnicas.

1.2.4.4 Despliegue de la Función de Calidad (DFC)

Esta técnica incluye entrevistas y documentación de la organización con las cuales se construye la tabla de opinión del interesado. Esta tabla se analiza con diagramas, matrices y métodos de evaluación para extraer los requisitos esperados e intentar obtener requisitos innovadores (Pressman, 2005). También esta técnica requiere una alta interacción con el analista.

1.2.4.5 Juego de roles

En su forma más simple, consiste en que el desarrollador, el analista y cada uno de los miembros del equipo de desarrollo del software toman el lugar del interesado y ejecutan la actividad de trabajo que éste desempeña. Ellos experimentan las inexactitudes y problemas ligados con el sistema que se está especificando. Se busca suministrarle al analista una perspectiva nueva del problema que le permita la obtención de los requisitos del sistema por construir (Raghavan *et al.*, 1994). Esta técnica también presenta alta participación de los involucrados.

1.2.4.6 Introspección

Esta técnica recomienda que el analista se ponga en el lugar del interesado y trate de imaginar cómo desearía éste la aplicación de software. Basado en estas suposiciones, el analista entrega recomendaciones al interesado sobre la funcionalidad que debería tener dicha aplicación (Goguen y Linde, 1993). El problema radica en que un analista no es un tipo normal de interesado, pues posee un conocimiento técnico más elevado; por ello, es posible que entre las recomendaciones haya cosas que el interesado aún no necesita o que

incluso no sabe que necesitará en un futuro. En este caso, el discurso se refiere más a la solución que al dominio del problema.

1.2.4.7 Casos de uso o escenarios

Son descripciones que incluyen actores, eventos, operaciones y objetivos de esas operaciones, generalmente ligados con el funcionamiento de una solución informática (Leffingwell y Widrig, 1999; Raghavan *et al.,* 1994; Pressman, 2005; Sommerville 2001). Los escenarios como técnica de obtención poseen dos limitaciones: exigen una alta participación del interesado en su elaboración y necesitan que él realice una concepción completa de la solución informática, que sólo sería posible al final del proceso de obtención de requisitos.

1.2.4.8 Talleres

Los requisitos tienen a menudo implicaciones cruzadas desconocidas para las personas implicadas individuales y que a menudo no se descubren en las entrevistas o quedan incompletamente definidas durante la misma. Estas implicaciones cruzadas pueden descubrirse realizando en un ambiente controlado, talleres facilitados por un analista del negocio, en donde las personas implicadas participan en discusiones para descubrir requisitos, analizan sus detalles y las implicaciones cruzadas. A menudo es útil la selección de un secretario dedicado a la documentación de la discusión, liberando al analista del negocio para centrarse en el proceso de la definición de los requisitos y para dirigir la discusión.

1.2.4.9 Cuestionarios

Es una técnica para recopilar información por medio de preguntas escritas, que se responden en ausencia del analista. No es muy confiable, ya que las respuestas obtenidas no pueden ser fácilmente verificadas o pueden estar "elaboradas por motivos propios o de grupo".

Se recomienda su uso:

- Cuando las personas que posean la información se encuentren dispersas geográficamente.
- Cuando se requiere obtener información al mismo tiempo de numerosas personas.
- Cuando la información necesaria está conformada por tablas o datos que pueden ser verificados por otra vía.

1.2.4.10 Selección de las técnicas de captura de requisitos.

Se decidió utilizar para la captura de requisitos las propuestas de Tormenta de Ideas por ser este un sistema nuevo y se necesita tener el criterio colectivo de la mayor cantidad de especialistas del tema. Posteriormente se realizan las Técnicas para Facilitar las Especificaciones de una Aplicación (TFEA) para tener una mejor comprensión y validación de los requisitos.

1.2.5 Lenguaje del lado del servidor

Son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Estos son independientes del cliente y es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo.

1.2.5.1 PHP 5.0

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas. Se muestra como código impregnado dentro de una página HTML. El modo de operación del PHP es el siguiente:

- El Navegador realiza una petición al servidor (se escribe la URL).
- Después el servidor ejecuta el código PHP solicitado y retorna el código HTML generado al Navegador.
- Por último el Navegador muestra la respuesta del servidor.

Características de PHP:

- **Velocidad:** Posee gran velocidad y no crea demoras en la máquina, por esta razón no requiere demasiados recursos del sistema.
- **Estabilidad:** Utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- **Seguridad:** El sistema debe poseer protecciones contra ataques, por lo que PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde el archivo *.ini.
- **Simplicidad:** Fácil de aprender sin dificultades en especial a los programadores con experiencia en los lenguajes de programación C y C++.

Ventajas de PHP:

- Este lenguaje es capaz de correr en la mayoría de las plataformas utilizando el mismo código fuente, ya sean plataformas Unix o Windows.
- Se puede ejecutar bajo diferentes servidores de aplicaciones web como son Apache, IIS, AOLServer, Roxen y THTTPD.
- Puede interactuar con muchos motores de bases de datos tales como MYSQL, Oracle, Informix, PostgreSQL, y otros muchos.
- Generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Está completamente escrito en el lenguaje C, así que se ejecuta rápidamente utilizando poca memoria.
- Es Código Abierto (Open Source), lo cual significa que el usuario no depende de una compañía específica para arreglar problemas de funcionamiento, y no está forzado a pagar actualizaciones anuales para tener una versión que funcione.

Se escoge este lenguaje por su facilidad de aprendizaje, su soporte multiplataforma tanto de diversos Sistemas Operativos como servidores HTTP y de bases de datos, además se distribuye de forma gratuita bajo una licencia abierta (GNU).

1.2.6 Lenguaje del lado del cliente.

Las técnicas y tecnologías del lado del cliente son aquellas que se ejecutan en el navegador del usuario que pueden ser: Internet Explorer, Mozilla Firefox, Opera, entre otros. Esto permite que algunos procesos se ejecuten en las máquinas de los clientes y así no se sobrecarga el servidor, además de agilizar las conexiones con los mismos. Los usuarios pueden visualizar toda la información ya que los navegadores son capaces de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en páginas que son el resultado de dicha orden.

En la realización de la propuesta de sistema integrado del proyecto Infodrez se utiliza el JavaScript y el HTML como lenguajes del lado del cliente.

1.2.6.1 HTML 4.0

HTML, siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). Estas constituyen la filosofía de este lenguaje, por medio de ellas se pueden controlar los elementos tipográficos del texto: tipo, color y tamaño de las fuentes, el estilo, así como también la inclusión de tablas, listas, formularios, la inserción de fotos, sonidos, fondos, los enlaces mencionados anteriormente.

Las etiquetas se pueden modificar por medio de sus atributos, éstos son del tipo atributo="valor" y se colocan detrás del nombre de la etiqueta
El nombre de la etiqueta y sus atributos se colocan entre los símbolos **<** y **>** y normalmente se usan dos, una de inicio y otra final, para conseguir el efecto deseado.

Etiquetas fundamentales que se encuentran en HTML:

- <html>: define el inicio del documento HTML, le indica al navegador que lo que viene a continuación debe ser interpretado como código HTML.
- <head>: define la cabecera del documento HTML, esta cabecera suele contener información sobre el documento que no se muestra directamente al usuario.

Dentro de la cabecera <head> podemos encontrar:

- <title>: define el título de la página. Por lo general, el título aparece en la barra de título encima de la ventana.
- <link>: para vincular el sitio a hojas de estilo o íconos
- <style>: para colocar el estilo interno de la página, ya sea usando CSS, JavaScript u otros lenguajes similares. No es necesario colocarlo si se va a vincular a un archivo externo usando la etiqueta <link>
- <body>: define el contenido principal o cuerpo del documento. Esta es la parte del documento html que se muestra en el navegador; dentro de esta etiqueta pueden definirse propiedades comunes a toda la página, como color de fondo y márgenes.

Este lenguaje además de mostrar texto, también permite visualizar gráficos, video, audio, así como vínculos hacia otras partes de la misma página u otros sitios de la red. Es el encargado de decirle al navegador dónde ubicar cada texto, imágenes, videos y la forma que van a tener estos. Entre las muchas ventajas que presenta HTML, se encuentra, que son posibles de crear y visualizar en cualquier sistema operativo.

Un documento HTML puede ser creado con la ayuda de un simple editor de texto, como el Bloc de Notas del sistema operativo Microsoft Windows. Sin embargo, también hay

herramientas profesionales que nos permiten crear las páginas en un entorno gráfico, como son el caso de Dreamweaver, FrontPage.

1.2.6.2 JavaScript 1.8

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Sencillo y pensado para hacer las cosas con rapidez. Personas que no tengan experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.

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

JavaScript es un lenguaje de programación interpretado, es decir, no requiere de compilación. Orientado a objetos, utilizado principalmente en páginas Web y con sintaxis semejante a la del lenguaje Java y el lenguaje C. Compatible con todos los navegadores modernos, por lo que es el lenguaje de programación del lado del cliente más utilizado.

Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- No se tienen en cuenta los espacios en blanco y las nuevas líneas
- Se distinguen las mayúsculas y minúsculas
- No se define el tipo de las variables
- No es necesario terminar cada sentencia con el carácter de punto y coma
- Se pueden incluir comentarios

Ventajas que presenta este lenguaje:

- No requiere tiempo de compilación.
- Los script pueden desarrollarse en un período de tiempo relativamente corto.
- Posee características de interfaz, que son gestionados por el navegador y por el código HTML.
- Los programas JavaScript tienden a ser pequeños y compactos, no requieren mucha memoria ni tiempo adicional de transmisión.
- Es independiente de la plataforma de hardware o sistema operativo, siempre y cuando exista un navegador con soporte JavaScript.

1.2.7 Gestores de Base de Datos.

Los sistemas de gestión de base de datos, conocidos como SGBD, sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, además de permitir crear y mantener una base de datos asegurando su integridad, confidencialidad y seguridad. Tiene como objetivo manejar de forma clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de los datos. Existen gestores de Bases de Datos muy utilizados como el MySQL y el PostgreSQL. Para el desarrollo de la aplicación se utilizará el MySQL con el objetivo de que la misma presente mayor portabilidad sobre la plataforma donde se despliegue.

1.2.7.1 MySQL 5.0.2

Permite la gestión de los datos de una base de datos relacional usando un lenguaje de consulta estructurado (SQL). Mediante estas consultas, MySQL llevará a cabo una determinada acción sobre la base de datos. Es una aplicación de código abierto hasta esta versión, permite redistribuir una aplicación que la contenga y modificar su código para mejorarla o adaptarla a nuestras necesidades. Es un sistema fácil de instalar y configurar ya sea en servidores de Windows o Linux. (Martín, 2006).

Características de MySQL:

- Acceso a las bases de datos de forma simultánea por varios usuarios y/o aplicaciones.
- Seguridad, en forma de permisos y privilegios, determinados usuarios tendrán permiso para consultar o modificar determinadas tablas. Esto permite compartir datos sin que peligre la integridad y la disponibilidad de estos.
- Potencia: SQL es un lenguaje muy potente para consulta de bases de datos.
- Portabilidad: Las consultas hechas usando SQL son fácilmente portables a otros sistemas y plataformas, debido a la estandarización que posee este lenguaje.
- Escalabilidad: 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.
- Conectividad: permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es común que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con sistemas operativos Windows, Linux, Solaris.
- Permite manejar registros de longitud fija o variable.

MySQL es un software de fuente abierta, ligado en gran medida al PHP y muy utilizado en aplicaciones Web, el entorno es intensivo en la lectura de los mismos, lo que hace a MySQL ideal para éste tipo de aplicaciones.

1.2.8 Patrones

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma.

GRASP (General Responsibility Assignment Software Patterns)
Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

En cuanto a las responsabilidades, UML define una responsabilidad como “un contrato u obligación de un clasificador”. Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento.
Estas responsabilidades pueden ser de dos tipos:
Conocer:

- Datos privados encapsulados.
- Objetos relacionados.
- Las cosas que pueden derivar o calcular.

Hacer:

- Hacer algo él mismo, como crear un objeto o hacer un cálculo.
- Iniciar una acción en otros objetos.
- Controlar y coordinar actividades en otros objetos.

Se pueden destacar cinco patrones principales que son:

1.2.8.1 Experto:

Asignar una responsabilidad al experto en información: la clase que tiene la información necesaria para la realización de la asignación.

1.2.8.2 Creador:

Asignar a la clase B la responsabilidad de crear una instancia de la clase A si se cumple que uno o más de los casos siguientes.

1. B agrega objetos de A.
2. B contienes objetos de A.
3. B registra instancias de objetos de A.
4. B utiliza más estrechamente objetos de A.
5. B contiene datos de inicialización que se pasarán a un objeto A cuando sea creado (por tanto B, es un Experto con respecto a la creación de A).
6. B es un creador de los objetos de A.

1.2.8.3 Bajo Acoplamiento

Asignar responsabilidad de manera que el acoplamiento permanezca bajo. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras.

1.2.8.4 Alta Cohesión

Asigna una responsabilidad de modo que la cohesión siga siendo alta. En la perspectiva del diseño orientado a objetos, la cohesión (ó, más exactamente, la cohesión funcional) es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

1.2.8.5 Controlador

Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- El “sistema” global (controlador de fachada).
- La empresa u organización global (controlador fachada).
- Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “Manejador<NombreCasoUso>” (controlador de casos de uso).

En el presente trabajo se utilizó el patrón experto, este se evidencia en las clases controladoras de los distintos diagramas del diseño puesto que estas poseen la mayor

información necesaria para resolver las funcionalidades requeridas. Se usó además el bajo acoplamiento, una vez que las relaciones entre las clases que participan en el desarrollo de los distintos casos de usos son mínimas asegura así que exista entre ellas una alta cohesión. Se utiliza también el patrón creador con el objetivo de asignar la responsabilidad de crear instancias de otras clases a la clase adecuada para este fin.

1.3 Conclusiones

En este capítulo se hizo un análisis de las herramientas que van a permitir la modelación del sistema propuesto, así como las funcionalidades de cada una de ellas. Como se pretende que el sistema cubra las expectativas de los usuarios, se utilizará como metodología de desarrollo RUP, unido al lenguaje de modelado UML 2.0 y como herramienta CASE el Visual Paradigm 6.0. Entre las técnicas de capturas de requisitos tenemos a utilizar la Tormentas de Ideas y las técnicas para facilitar las especificaciones de una aplicación (TFEA). Además de la utilización del JavaScript 1.8 y HTML 4.0 como lenguajes del lado del cliente, PHP 5.0 del lado del servidor y MySQL 5.0.2 como gestor de Base de Datos.

Capítulo 2. Propuesta del sistema

2.1 Introducción

En este capítulo se abordará temas referentes al sistema propuesto, profundizando en sus características y en la situación problemática existente. Se expondrán los dos primeros flujos de trabajo de la metodología RUP, específicamente “Modelo de negocio” y “Requerimientos”. El primer flujo es donde se genera todo lo referente a la dinámica del negocio, se obtienen los diagramas Casos de Uso del Negocio, los diagramas de actividades de cada Caso de Uso y el Modelo de Objetos. Por otra parte en Requerimientos se genera el diagrama de Caso de Uso del Sistema, las descripciones de cada uno de ellos y los Requisitos Funcionales y no Funcionales, así como las técnicas de validación de estos requisitos.

2.2 Desarrollo

2.2.1 Fundamentación de la situación problémica

La Cátedra Honorífica de Ajedrez “Remberto Fernández” perteneciente a la Universidad de las Ciencias Informáticas unido al proyecto Infodrez, juegan un papel fundamental en la informatización del ajedrez cubano. Actualmente este proyecto tiene concebido herramientas que gestionan información relacionada con las principales áreas de este deporte: arbitraje, estadística, torneos y juegos en tiempo real.

El área de Arbitraje permite informatizar la labor de un árbitro en un evento, a partir de la creación de torneos, el control de las rondas que lo componen además de los reportes que genera que ofrecen información de cada jugador del torneo. La estadística posibilita a jugadores y entrenadores hacer un estudio del rendimiento en un período de tiempo determinado, además de obtener las características de sus partidas. Mientras que el Visor de Torneos facilita la gestión de la información relacionada con los torneos de ajedrez, para que toda la información se encuentre disponible en un mismo lugar y pueda ser consultada de forma eficiente y rápida.

Según lo expuesto anteriormente en algunos de los módulos se gestiona u ofrece la misma información, lo que provoca una serie de problemas que traen consigo serias consecuencias para el cliente, además de no ser una solución informática factible. Estos problemas son:

Réplica de código y réplica de reportes

Visor de torneos y arbitraje trabajan con información referente a los torneos, en común tienen código sobre los reportes que generan los resultados de cada una de las rondas, la variación de ELO y Rating Performance de cada jugador durante el evento. Los reportes que se generan en cada uno de estos módulos no son estándares entre sí, aunque hacen referencia al mismo tipo de información.

Ambos sistemas trabajaban la gestión del torneo presentando las siguientes funcionalidades:

- Creación de un torneo (características, definición de grupo y asignación de jugadores).
- Generación del emparejamiento de cada ronda según el sistema de juego.
- Sistema de clasificación del torneo.

Información no centralizada

Los módulos de Visor de Torneos, Arbitraje y Estadística, trabajan en principio con la información que se genera a partir del desempeño de un jugador en un torneo. Existe inconsistencia entre esta información debido a la no centralización de la misma. Los principales problemas identificados han sido:

- Nombres de jugadores duplicados y escritos de formas diferentes.
- Nombres de eventos duplicados y escritos de formas diferentes.
- No correspondencia entre partidas jugadas en un torneo.

Información no estandarizada

Las informaciones y reportes que se generan en visor de torneos, arbitraje y estadística son similares sin embargo la manera de mostrarse a los usuarios son diversas.

2.2.2 Propuesta del sistema

Se pretende diseñar una plataforma integrada usando software libre que abarque todos los ámbitos de la cultura ajedrecística, configurable según las necesidades de los usuarios y reutilizable para una amplia gama de manifestaciones deportivo-recreativas de similar carácter.

Con el diseño de esta plataforma se proporcionará una aplicación que podrá ser utilizada para brindar características de cualquier jugador, ver torneos que estén registrados o efectuándose en tiempo real en la misma. El sistema debe satisfacer los requisitos funcionales y las necesidades de los usuarios a los cuales va enfocado el producto.

2.2.3 Flujo de procesos que se realizan en el Ajedrez por Correspondencia.

- Solicitar jugar torneo: todo jugador que haya solicitado su inscripción en los eventos que organiza la FECAP, será ubicado en algún torneo que esté acorde con su rating, y tiene la responsabilidad de jugar el mismo. Esta actividad la realiza la federación (2).

- Solicitar anular participación en un torneo: el jugador puede gestionar ante la Comisión de Eventos su anulación de un torneo, con el árbitro de la competencia. El árbitro enviará al jugador su decisión de otorgar o no la anulación. Si la decisión es negativa lo hará por vía certificada. Si el árbitro otorga la anulación, el solicitante guardará la notificación hasta pasados 6 meses de concluido el torneo. Si no es otorgada la anulación el jugador deberá comenzar a jugar tan pronto reciba la respuesta, aun cuando vaya a hacer uso de su derecho de apelación (2).

- Entregar documentación del torneo: la Comisión de Eventos deberá entregar al jugador la documentación del torneo solicitado o la comunicación de la imposibilidad, hasta ese momento, de completar un grupo acorde a su solicitud. Esta documentación estará conformada por la nómina del torneo, que a su vez contendrá el número del torneo, nombre y dirección del árbitro, rating y categoría de cada participante, así como el grupo de jugadores con el que llevará piezas blancas. Este documento contendrá además las bases del torneo con la dirección de la Comisión de Eventos que organiza el evento para cualquier reclamación, forma en que se controlará el tiempo de reflexión, si el torneo otorga normas y la puntación necesaria para obtenerlas (2).

- Iniciar partida de torneo: el jugador podrá comenzar a jugar en cualquier momento entre el instante que recibe la documentación del torneo y la fecha señalada para el inicio. Para ello enviará su primera jugada a los oponentes con los que lleva piezas blancas y aguardará por los envíos de aquéllos con quienes lleva piezas negras (2).

- Enviar jugadas: al enviar sus jugadas, los jugadores deberán consignar los siguientes datos: número del torneo, la última jugada del contrario .En el caso de haber aceptado una proposición, es obligatorio añadir a la última jugada del rival todos los movimientos aprobados (propios y del adversario). La jugada propia. Fecha en que su rival envió la jugada, que en el caso de los torneos postales, será la fecha del matasellos del correo expedidor y en el caso de los torneos por correo electrónico, la fecha reflejada por el servidor del jugador que la envió (2).

- Conclusión del torneo: se da como torneo concluido cuando el número de partidas inconclusas no exceda la cantidad de 6 y en cada una de estas se hayan realizado como

mínimo 36 jugadas. A partir de ese momento, los jugadores dispondrán de 30 días para realizar el análisis de sus partidas inconclusas y enviarlos al árbitro por correo certificado (1).

- Informe final del torneo: una vez que el árbitro recibe los resultados del proceso de adjudicación, enviará a cada jugador el cuadro sinóptico del torneo y la fundamentación de sus partidas adjudicadas. El jugador dispondrá de 7 días a partir de la fecha en que lo recibe para solicitar se subsane cualquier error que detecte. El árbitro tendrá hasta 60 días para cerrar el torneo (1).

2.2.4 Flujo de procesos que se realizan en el Juego Online.

- Organizar y crear torneos que son actividades generalmente realizadas por el organizador o director del evento, donde se acuerdan las bases y los objetivos del torneo, los árbitros que serán invitados y los jugadores que estarán presentes en el mismo.

- Realizar pareos en las rondas que es dirigido por los árbitros auxiliares o adjuntos que estarán desarrollando el torneo, a grandes rasgos los pareos son la acción de decidir cuáles serán las parejas de jugadores a enfrentarse en cada ronda teniendo en cuenta algunos parámetros necesarios.

- Jugar las partidas de cada ronda es una actividad realizada por cada jugador que se enfrenta en el torneo, llegando a un resultado final, victoria, derrota o tabla.

- Informar los resultados después de cada ronda y el resultado final del torneo es tarea de los árbitros involucrados, basado en esa información de siguen haciendo las rondas restantes.

2.2.5 Modelo del Negocio

El modelo de negocio es un artefacto de la disciplina de Ingeniería del Software que tiene como objetivo conocer la estructura y dinámica de la organización en el cual se va a implantar el sistema, comprender los problemas actuales e identificar las posibles mejoras. Además asegurar que el equipo de proyecto, los usuarios finales, tengan un entendimiento común de la organización; da una visión de qué es lo necesario hacer para satisfacer las exigencias de los usuarios. Teniendo en cuenta que en el flujo de trabajo de modelo de negocio se generan artefactos, a continuación se hace referencia a los mismos.

2.2.5.1 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, organización, máquina o sistema de información externos con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio obteniendo un resultado que lo beneficia (Jacobson, 2004).

Nombre	**Descripción**
Organizador	Organiza el torneo, define sus bases y objetivos, gestiona a los jugadores y árbitros que participarán.
Jugador	Es quien juega las partidas, ya sea en tiempo real o por correspondencia, es el protagonista principal.
Federación	Es la rectora del Ajedrez por Correspondencia para el desarrollo de las competencias oficiales. Es la encargada de gestionar los torneos a los jugadores por su rating.
Observador	Es aquel participante del torneo ya sea jugador o no que observa las partidas que se están jugando.

2.2.5.2 Trabajadores del negocio

Un trabajador del negocio representa una persona o un sistema automatizado (software) que actúa en el negocio realizando una o varias actividades comprendidas dentro del caso de uso, interactuando con los trabajadores del negocio y manipulando entidades del mismo (Jacobson, 2004).

Nombre	**Descripción**
Árbitro	Realiza un monitoreo constante de todas las partidas, atento a que no se cometan violaciones, en caso de materializarse alguna debe expulsar al infractor del evento. Es el que realiza el pareo para comenzar la ronda. El árbitro en ocasiones puede funcionar como observador. En el caso del Ajedrez Postal es el encargado de hacer que se cumplan las bases del torneo como por ejemplo, verificar que un jugador realice su jugada en el tiempo correspondido.
Director del Torneo (Árbitro	Lleva el control del listado de todos los participantes en el torneo. Informa los resultados cada vez que termina una ronda y tiene el

Principal)	conocimiento de todo el proceso de pareo. Es la persona encargada de dirigir el torneo, así como conceder licencias solicitadas por los jugadores, actualizar la tabla de resultado del torneo. Confecciona listado de ganadores y notifica las posiciones del torneo. También decide partidas no concluidas.
Encargado del Rating	Es el encargado de calcular y actualizar el rating del torneo. Realiza un informe con el rating actualizado de cada jugador y envía un informe actualizado con el rating a los jugadores del torneo.
Encargado de las solicitudes de Torneos	Es el encargado de analizar los motivos de la licencia solicitada por el jugador y así tiene la autoridad de concedérsela o no.

2.2.5.3 Diagrama de Casos de Uso del Negocio

Un caso de uso del negocio representa un proceso dentro del negocio que se estudia, por lo que se corresponde con una secuencia de acciones con un orden lógico y que producen un resultado observable para ciertos actores del negocio (Jacobson, 2004). A continuación se muestra el diagrama de Casos de Uso del Negocio, donde se representa una vista general de la dinámica del negocio a través de relaciones entre Casos de uso y actores.

Diagrama de Casos de Uso de Negocio

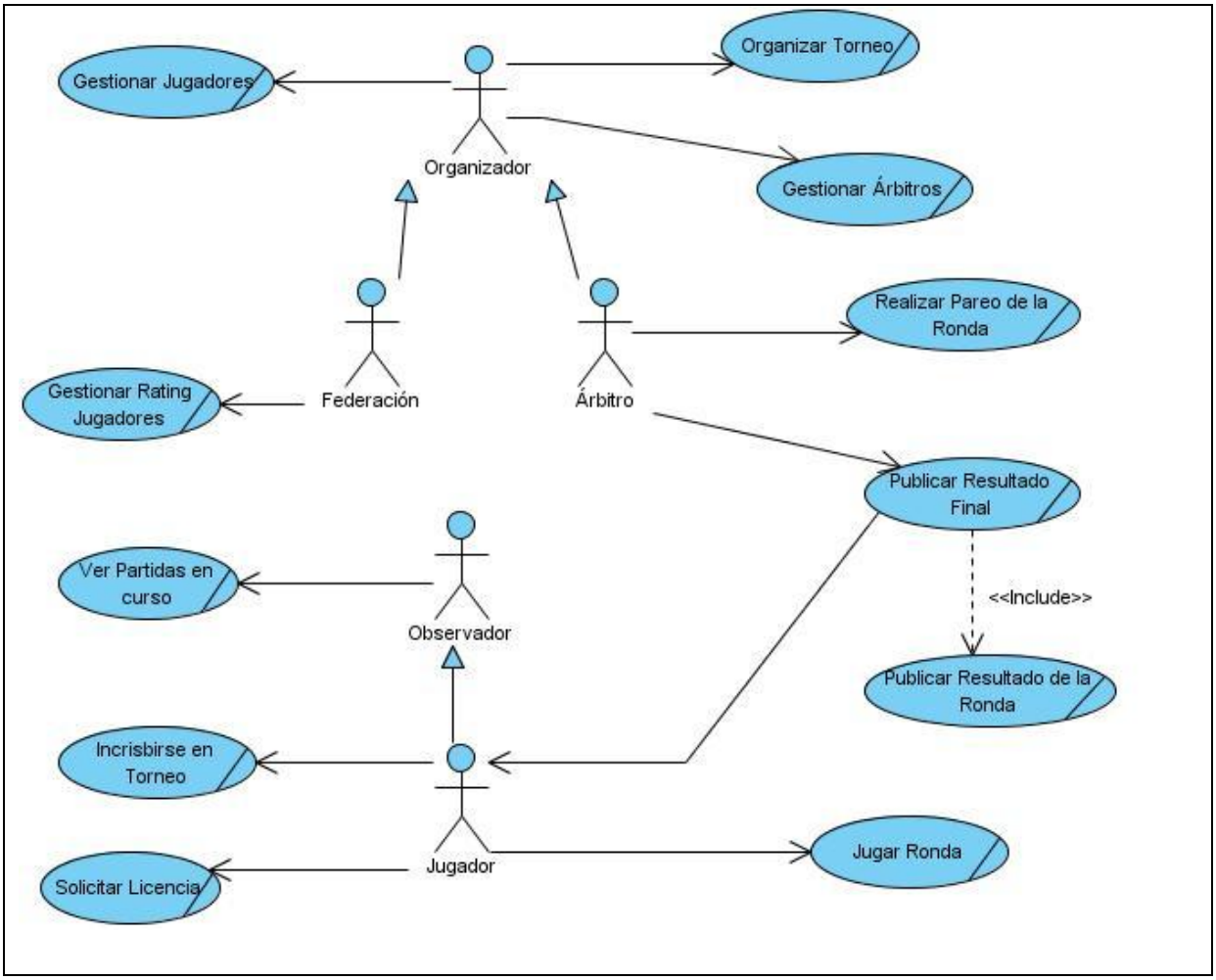


Figura 2.1: Diagrama Casos de Uso del negocio.

2.2.5.4 Reglas del Negocio

Para un mejor entendimiento del negocio seguidamente se establecen las reglas del mismo:

1- Un usuario que juegue en ajedrez por correspondencia puede jugar varias partidas a la vez.

2- Si el torneo otorga normas, se debe mostrar la puntuación necesaria para obtenerlas, (la que puede ser modificada si en el transcurso de la competencia cambian las condiciones iniciales).

3- El jugador debe saber la fecha de inicio del evento, nombre, rating y categoría de cada participante.

4- Cualquier información que varíe el comportamiento habitual de las competencias, como por ejemplo, las jugadas obligatorias a realizar en los torneos temáticos, características propias de los torneos por equipos, deben hacerlas los árbitros.

5- Potestad del árbitro de establecer la obligatoriedad de la realización de seis jugadas trimestrales mínimas en el evento que serán comprobadas periódicamente a fin de asegurar el desarrollo adecuado del mismo.

6- El organizador y el árbitro principal son los únicos encargados de hacer la gestión de los árbitros adjuntos o auxiliares.

8- Es obligatorio saber nombre y dirección del árbitro que atenderá el evento para cualquier reclamación.

2.2.5.5 Diagrama de Clases del Modelo Objetos del Negocio

Este diagrama describe cómo colaboran los trabajadores y las entidades del negocio, y proporciona un acercamiento a la identificación de los futuros actores y entidades del sistema.

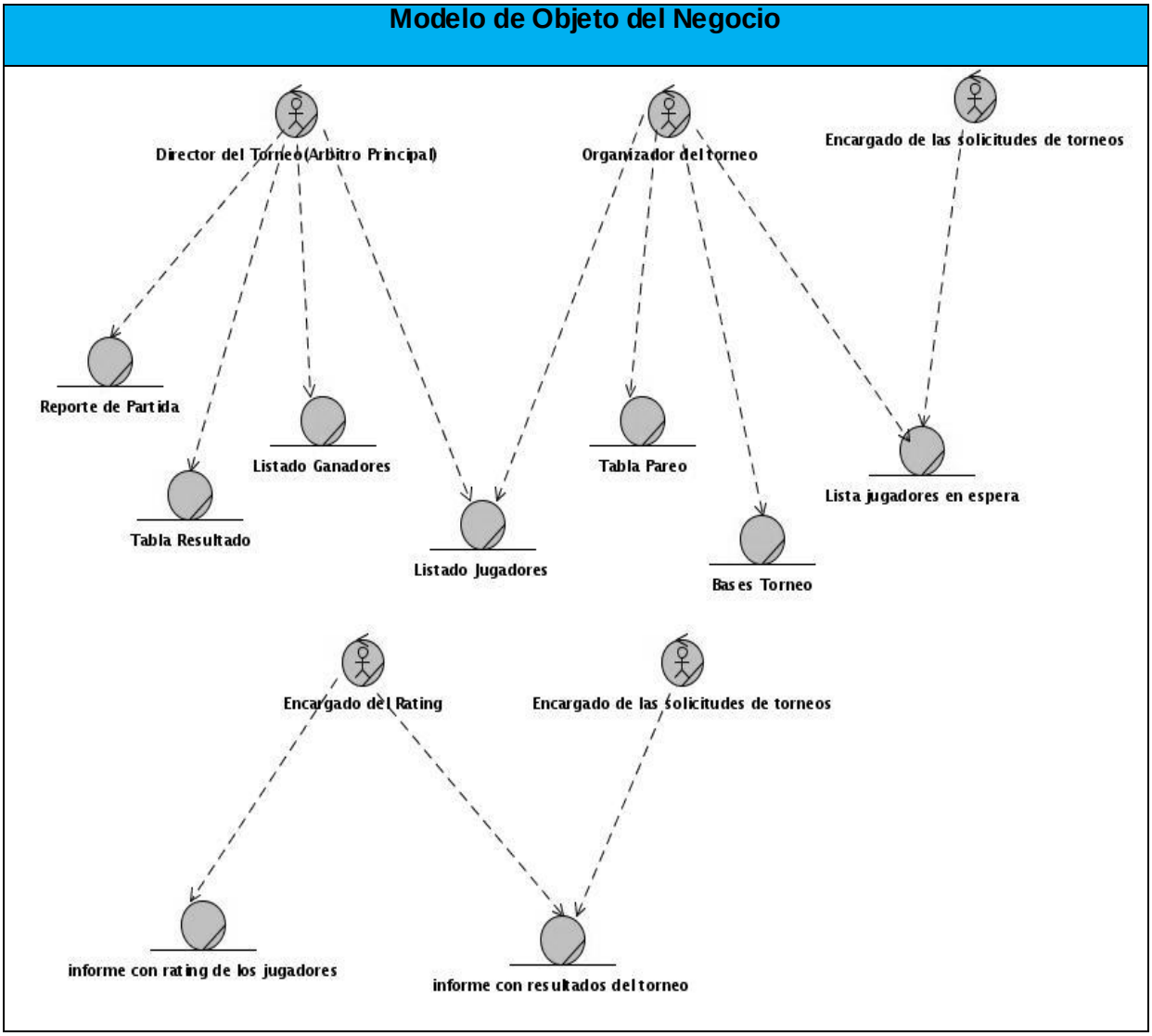


Figura 2.2: Diagrama Modelo de Objetos del Negocio

2.2.6 Requerimientos

En el flujo de trabajo de Requerimientos se define que es lo que debe hacer el sistema, siendo necesaria la captura de requisitos para identificar las funcionalidades requeridas y las restricciones impuestas. El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), define requerimiento como:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Características que deben tener los requerimientos:

- Especificados por escrito
- Posibles de probar o verificar
- Descritos como una característica del sistema a entregar
- Lo más abstracto y conciso posible. Para evitar malas interpretaciones

Los requisitos se pueden clasificar en 2 categorías:

- Requisitos funcionales
- Requisitos no funcionales

2.2.6.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. En el negocio existen actividades que serán posibles automatizaciones, estas no constituyen exactamente requisitos funcionales, pero son el inicio para identificar qué debe hacer el sistema.

Funcionalidades que debe cumplir el sistema:

RF 1: Autenticar usuario

RF 2 Gestionar usuario

2.1 Adicionar Usuario

2.2 Ver Usuario

2.3 Modificar Usuario

2.4 Eliminar Usuario

RF 3: Gestionar Jugador

3.1 Adicionar Jugador

3.2 Ver Jugador

3.3 Modificar Jugador

3.4 Eliminar Jugador

RF 4 Gestionar Torneo

4.1 Adicionar Torneo

4.2 Ver Torneo

4.3 Modificar Torneo

4.4 Eliminar Torneo

4.5 Listar Torneo

RF 5 Gestionar jugadores del torneo

5.1 Adicionar jugadores a un torneo

5.2 Eliminar jugadores de un torneo

RF 6 Buscar jugador

6.1 Listar jugadores por criterios de búsqueda

RF 7 Gestionar partida

7.1 Registrar Partida

7.2 Ver Partida

7.3 Eliminar Partida

RF 8 Permitir observar a otros jugadores las partidas en curso.

8.1 Permitir a jugadores que están observando partidas realizar comentarios de misma

RF 9 Gestionar Título del Jugador

9.1 Adicionar Título del Jugador

9.2 Modificar Título del Jugador

9.3 Eliminar Título del Jugador

RF 10: Mostrar Ritmo de Juego

RF 11 Gestionar Árbitro

11.1 Adicionar Árbitro

11.2 Ver Árbitro

11.3 Modificar Árbitro

11.4 Eliminar Árbitro

RF 12 Notificar por Correo

12.1 Mostrar Interfaz de Correo

RF 13 Mostrar Partidas

13.1 Mostrar partidas creadas

RF 14 Mostrar Datos de Partidas

14.1 Mostrar campos de la partida seleccionada

RF 15 Conceder Licencia

15.1 Actualizar tabla del jugador

RF 16 Solicitar Licencia

RF 17 Definir Partidas no Concluidas

RF 18 Permitir unirse a Torneo

18.1 Mostrar Torneos Disponibles

18.2 Mostrar Datos de Inscripción

2.2.6.2 Requisitos no funcionales

1- Requerimiento de Software:

Para la instalación del servidor

- Servidor Apache versión 2.0 superior.
- PHP 4 o superior con módulo LDAP para autenticar usuarios de Directorio Activo.
- Servidor de bases de datos MySQL versión 4.0 superior.
- Sistemas Operativos GNU/Linux (recomendado) o Microsoft Windows.

Para la interpretación por clientes (Navegadores Web):

- Internet Explorer v6 o superior.
- Mozilla Firefox 2.0 superior (recomendado).

2- Requerimiento Hardware:

- Mínimo Requerido: RAM 256 Mb, Disco duro 3 Gb.
- Conexión de red por MODEM o fibra óptica.
- Recomendado: RAM 512 Mb, Disco duro 10Gb.

3- Requerimiento de Usabilidad:

Interfaz web fácil de usar para personas que no son especialistas en informática, permitiendo un mejor uso e interacción con los usuarios y mayor facilidad de intercambio de información.

4- Requerimiento de Diseño e Implementación

El sistema se recomienda que funcione sobre una plataforma de sistema operativo GNU/Linux a modo texto, basado en el modelo cliente-servidor. Utilizar además para su desarrollo herramientas como el Quanta, Bluefish, ZendStudio, phpmyadmin y Visual Paradigm. Portabilidad: Es un sistema multiplataforma.

5- Requerimiento de Seguridad:

Verificar autenticidad y niveles de privilegio del usuario antes de hacer cualquier operación dentro del sistema. Verificar que las funcionalidades sean mostradas en correspondencia con los niveles de privilegios. Protección contra operaciones no autorizadas que en algún momento puedan afectar la integridad de los datos. La autenticación se realiza contra un LDAP que solo permite al acceso de usuarios del dominio.

Confidencialidad: Solo se permitirán conexiones mediante usuarios del directorio activo.

6- Político – Culturales:

Este módulo junto a Infodrez fueron creados inicialmente como una necesidad en la Universidad, por tener probabilidades de llegar a ser un producto nacional y de competir en el mercado internacional, debe ajustarse a las características de otros software con este fin que están actualmente predominando en el ajedrez mundial.

7- Requerimiento de Soporte:

Legales:

La plataforma está basada en la licencia GNU/GPL.

Confiabilidad:

Realiza control de errores y recuperación ante fallos.

2.2.6.3 Actores del sistema

Un actor del sistema está definido como el individuo, grupo de individuos, sistema automatizado o máquina que interactúa con el mismo intercambiando información. Los trabajadores del negocio que tienen actividades a automatizar son candidatos a actores del sistema. Si algún actor del negocio continúa interactuando con el sistema, entonces también será actor del sistema.

Nombre	**Justificación**
Usuario	Debe registrarse si es la primera vez que entra al sistema, y en caso de que no sea así, debe autenticarse. El usuario es un autor genérico, una vez que entra al sistema pasa a desempeñar su rol, que puede ser jugador, árbitro u organizador del torneo.
Jugador	Es el actor principal, es quien juega las partidas dentro del sistema.
Árbitro	Es el encargado de monitorear que no se cometa ninguna infracción en el transcurso de las partidas. Es quién expulsa a los jugadores si cometen alguna indisciplina.
Organizador del torneo	Organiza los torneos que se juegan en el sistema, puede funcionar como árbitro.

2.2.6.4 Definición de Casos de Uso del Sistema

Los casos de uso son artefactos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

Casos de Uso

CUS-1	Registrarse
Actor	Usuario
Resumen	Cuando el usuario desea entrar al sistema por primera vez, debe llenar unos datos para que el sistema lo registre. Su correo electrónico, nombre y apellidos, su nick, contraseña y sexo.

CUS-2	Autenticarse
Actor	Usuario
Resumen	Cuando el usuario desea entrar al sistema debe escribir su correo electrónico y su contraseña.

CUS-3	Conceder Licencia
Actor	Árbitro
Resumen	El árbitro es quien decide si se da licencia o no a un jugador, en cualquiera de los casos debe comunicarlo vía electrónica.

CUS-4	Definir Partidas no Concluidas
Actor	Árbitro
Resumen	Es el encargado de decidir las partidas que no han concluido, pasado el tiempo reglamentado.

CUS-5	Observar Mensajes Privados
Actor	Árbitro
Resumen	El árbitro observa los mensajes privados de todos los usuarios que están online. Con el objetivo de no permitir indisciplinas.
CUS-6	Gestionar Invitación Privada
Actor	Jugador
Resumen	El jugador podrá hacer una invitación privada a otro jugador, el cual puede aceptar o no el reto.

CUS-7	Observar Juego
Actor	Jugador
Resumen	El jugador puede seleccionar una partida de las que se están jugando para observarlo, solo puede ser una a la vez.

CUS-8	Solicitar Torneos Disponibles
Actor	Jugador
Resumen	Puede seleccionar algún torneo, dependiendo siempre de su rating con el que pide el torneo. Luego de solicitar el torneo se lo notificará por correo su inscripción.

CUS-9	Realizar Jugada
Actor	Jugador
Resumen	El jugador una vez que esté dentro de una partida creada, podrá realizar una jugada y espera a que su oponente juegue para volver a hacerlo. Cuando el oponente haga su jugada si no está dentro del sistema se le notificará por correo

CUS-10	Gestionar Invitación pública
Actor	Jugador
Resumen	El jugador podrá o no aceptar alguna partida pública creada, o podrá crearla.

CUS-11	Solicitar Licencia
Actor	Jugador

Resumen	Si el jugador está imposibilitado de jugar debe pedir licencia al árbitro del torneo, en caso de que sea anulada su petición. Debe jugar.

CUS-12	Gestionar Jugadores
Actor	Organizador del torneo
Resumen	Una vez que el organizador cree un torneo, él es el encargado de aceptar o rechazar los jugadores que solicitaron registrarse en el torneo, en dependencia del rating que estos tengan. Debe enviar las notificaciones al correo.

El diagrama de casos de uso del sistema describe la relación entre actores y casos de usos. Es un artefacto que modela las funciones deseadas para el sistema y su entorno.

Diagrama Casos de Uso del Sistema

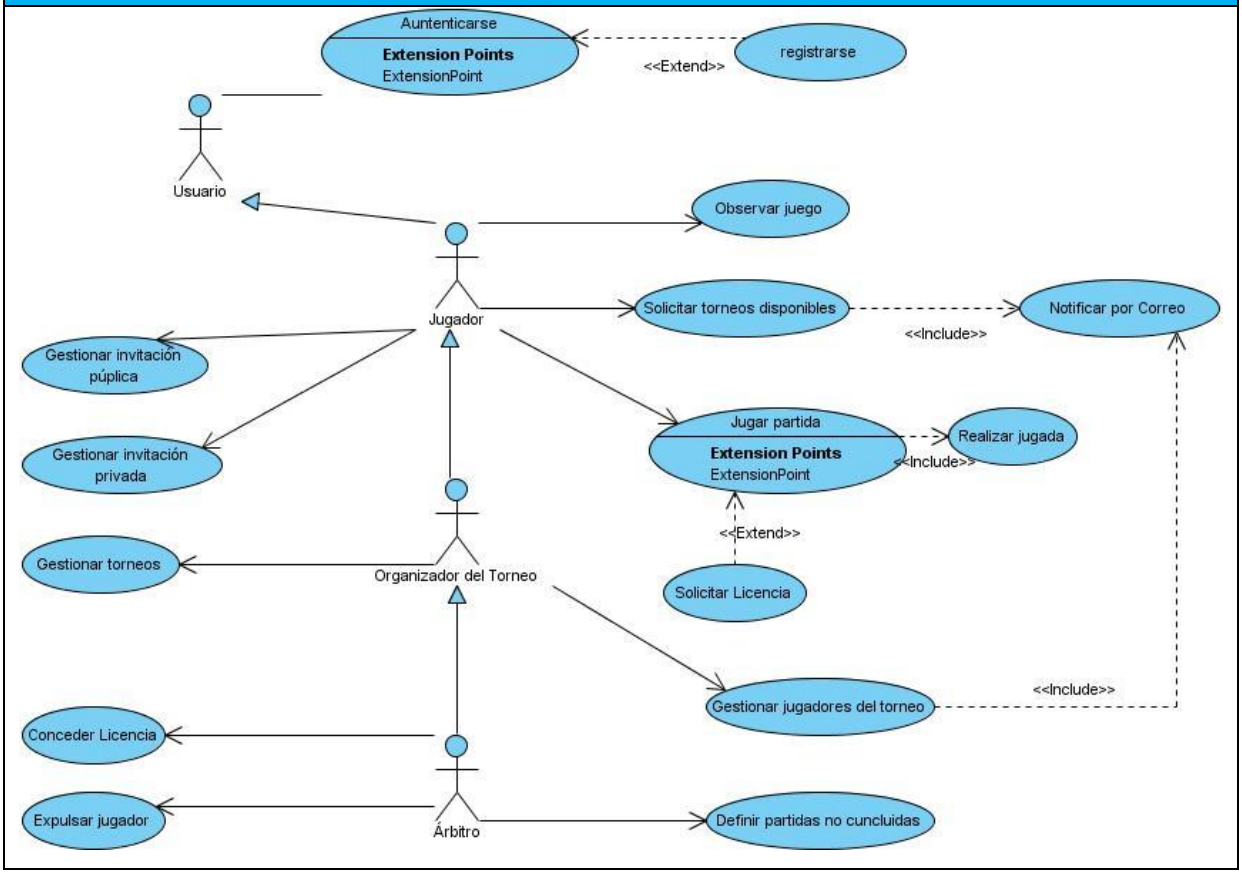


Figura 2.4: Diagrama Casos de Uso del Sistema

2.2.6.5 Descripción de los Casos de Uso del Sistema

Descripción del Caso de Uso Autenticarse.

Caso de Uso	**Autenticarse**
Actores	Jugador, Organizador del Torneo, Árbitro
Propósito	Entrar al sistema para desempeñar el rol que le corresponde.
Resumen	El CUS se inicia cuando el Jugador, Organizador del torneo o el director del torneo deciden autenticarse en el sistema, el sistema verifica si la autenticación es correcta y estos usuarios pasan a desempeñar su rol.
Referencias	RF 1, RF 2.1
Precondiciones	Usuario sin autenticar.
Pos_condiciones	Usuario autenticado y dentro del sistema.

Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Jugador, Organizador del Torneo, Árbitro: Accede al sistema vía Web. 2- Jugador, Organizador del Torneo, Árbitro: Introduce usuario y contraseña	3- Verifica si el usuario y la contraseña son correctos. 4- Si son correctos entran al sistema.
Curso alterno de los eventos	
6- Jugador, Director del Torneo Árbitro: Corrige los datos incorrectos e intenta de	5- Si los datos son erróneos le muestra un mensaje de error informándole al usuario que verifique si se autentico correctamente.

nuevo entrar al sistema hasta introducir el usuario y la contraseña correctos.	6- Se ejecuta la actividad 4 del curso normal.
Prioridad	Crítico

Descripción del Caso de Uso Jugar Partida

Caso de Uso	**Jugar Partida**
Actores	Jugador
Propósito	Jugar una partida mediante el sistema
Resumen	El CUS se inicia cuando el Jugador solicita al sistema jugar una partida, el sistema muestra al usuario una interfaz dándole la posibilidad de seleccionar la partida que desee jugar así como la modalidad (Juego Online o Ajedrez por Correspondencia). El jugador selecciona la partida que desea jugar.
Referencias	RF 1, RF 7.1
Precondiciones	Debe existir varios jugadores conectados en el sistema
Pos_condiciones	Se realiza las jugadas correspondientes a dicha partida.

Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Solicita la opción de jugar partida en el sistema.	2- Muestra al usuario una interfaz dándole la posibilidad de seleccionar la partida que desee jugar.
3- Selecciona la partida que desea jugar	4-Muestra una interfaz gráfica dándole la posibilidad de empezar la partida

5- Juega la partida completa	6- Termina caso de uso
Curso alterno de los eventos	
5- No juega la partida completa por abandono y la partida queda sin concluir. 6- El caso de uso finaliza.	
Prioridad	Crítico

Descripción del Caso de Uso Gestionar Torneos

Caso de Uso	**Gestionar Torneo**
Actores	Organizador de Torneo
Propósito	El Organizador del torneo tenga la posibilidad de Crear, Eliminar, Ver, Listar y Modificar torneos en el sistema
Resumen	El CUS se inicia cuando el Organizador del Torneo selecciona la opción de Gestionar Torneo, luego selecciona el tipo de gestión, introduce los datos necesarios y el sistema realiza la acción seleccionada por el Organizador del Torneo.
Referencias	RF 4 , RF 4.1, RF 4.2, RF 4.3, RF 4.4, RF 4.5
Precondiciones	Existe una necesidad de gestionar un torneo.
Pos_condiciones	Se gestiona la acción seleccionada por el Organizador.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**

1- El caso de uso inicia cuando el Organizador del torneo accede al vínculo Gestionar torneo. 3- Selecciona una de las opciones.	2- El sistema muestra las opciones de crear, eliminar, ver, listar y modificar torneos. 4- El sistema muestra la sección solicitada. 5- Termina caso de uso.
Sección : Crear Torneo	
4- Solicita la opción de Crear Torneo. Introduce los datos necesarios teniendo en cuenta la información necesaria, que son los objetivos y las bases. Para los objetivos es necesario conocer: - El árbitro del torneo. - El Rango de Elo que participará. - Categoría del torneo. - Si habrá o no aumento de categoría para los jugadores participantes. Para las bases se debe saber: - Cantidad de participantes. - Calidad o nivel del torneo (se mide por el Elo promedio de los jugadores). - Fecha de inicio y fin del torneo. - Nombre del torneo. 3- Introduce los datos que requiere el formulario.	2-Muestra una interfaz donde le muestra al Organizador del torneo los campos que debe llenar para Crear un torneo. 4-Envía los datos del torneo creado a la tabla de torneo en la base de datos. 5- Termina caso de uso

Sección: Modificar Torneo	
1- Solicita modificar Torneo	
	2-Realiza una consulta a la base de datos, a la tabla de torneos y solicita torneo creado que aun no ha empezado para modificarlos.
	3-Muestra una interfaz con todos los torneos.
4- Selecciona el torneo que desea modificar.	
	5-Muestra una interfaz con los datos del Torneo.
6-Modifica los datos que desee del Torneo.	
	7-Actualiza los campos modificados en la base de datos.
	8- Termina el caso de uso
Curso alterno de los eventos	
	3- Si no existen torneos creados se envía un mensaje informando que no existen torneos en el sistema. 4-El caso de uso finaliza.
Sección : Eliminar Torneo	
1- Solicita la opción de eliminar un torneo del sistema	
	2-Realiza una consulta a la base de datos, a la tabla de torneos y solicita todos los torneos creados hasta el momento.
	3-Muestra una interfaz con todos los torneos existentes.
4- Selecciona el torneo que desea eliminar.	
	5-Elimina el torneo seleccionado de la base de datos.

	6- Termina caso de uso
Curso alterno de los eventos	
	3- Si no existen torneos creados se envía un mensaje informando que no existen torneos en el sistema. 4-El caso de uso finaliza.
Prioridad	Crítico

Descripción del Caso de Uso Gestionar Torneos

Caso de Uso:	Gestionar Jugador
Actores:	Organizador del Torneo
Propósito:	Permite al árbitro gestionar toda la información referente a los jugadores: adicionar, ver, modificar y eliminar un jugador.
Resumen:	El CUS se inicia cuando el Organizador del Torneo selecciona la opción de Gestionar Jugador, luego selecciona el tipo de gestión, introduce los datos necesarios.
Precondiciones:	Existe la necesidad de gestionar un jugador de un torneo.
Pos_condiciones	Se Gestiona la acción seleccionada por el Organizador del Torneo.
Referencias	RF 5, RF 5.1, RF 5.2
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Organizador del Torneo selecciona dentro de la página principal la opción Gestionar Jugador del Menú Torneo.	2- El sistema brinda las opciones de: - Adicionar Jugador

	-Ver Jugador -Modificar Jugador -Eliminar Jugador
3 -El Organizador del Torneo selecciona una de las opciones.	4- Si selecciona “Adicionar Jugador” ver sección **Adicionar Jugador** - Si selecciona “Ver Jugador” ver sección **Ver Jugador** -Si selecciona “Modificar Jugador” ver sección **Modificar Jugador** -Si selecciona “Eliminar Jugador” ver sección **Eliminar Jugador**
Sección: “Adicionar Jugador”	
5- El Organizador del Torneo selecciona la opción de Adicionar Jugador	6- El sistema muestra el formulario con todos las opciones para que el Organizador del Torneo entre todos los datos que se necesitan para adicionar un jugador al sistema. -Nombre -Apellido -Sexo -Título -Código FIDE -Federación -Fecha de nacimiento -Elo -K

5- El Organizador del Torneo introduce los datos necesarios para adicionar un nuevo jugador y pulsa el botón Aceptar.	6-El sistema verifica que los datos insertados son correctos.
	7-El sistema verifica que el jugador no se encuentre registrado en la base de datos, en caso afirmativo lo adiciona a la base de datos y muestra un mensaje informando que el registro se realizó correctamente, concluyendo con esto el caso de uso del sistema.
Flujo Alterno de Eventos	
	7-Si los datos introducidos son incorrectos el sistema muestra un mensaje indicando en que elemento del formulario fue donde estuvo el error. Se procede al flujo normal de eventos desde el paso 6.
	8-Si el jugador ya existe el sistema emite un mensaje informando la existencia del mismo.
Sección: "Ver Jugador"	
5-El Organizador del Torneo selecciona el jugador a consultar datos	6-El sistema muestra los datos del jugador seleccionado, termina así el caso de uso
Sección: "Modificar Jugador "	
5- El Organizador del Torneo selecciona el jugador que desea modificar y marca la opción de Modificar Jugador	6- El sistema verifica que el jugador no pertenezca a un torneo que no haya finalizado, en caso positivo muestra los datos del jugador a ser modificados
7- El Organizador del Torneo modifica los datos y pulsa el botón Aceptar	8- El sistema verifica que los datos modificados son válidos, en caso positivo estos se actualizan en la base de datos,

	concluyendo así el caso de uso del sistema.
Flujo Alterno de Eventos	
	8- Si el jugador pertenece a un torneo que no haya finalizado se muestra un mensaje informando sobre la situación
	9- Si los datos que se modifican son incorrectos el sistema mostrará un mensaje señalando el campo donde los datos no son válidos.
Sección: "Eliminar Jugador "	
5- El Organizador del Torneo selecciona el jugador que desea eliminar y marca la opción de Eliminar Jugador	6- El sistema verifica que el jugador no pertenezca a un torneo que no haya finalizado, en caso positivo muestra un mensaje de confirmación para verificar que realmente se quiere eliminar el jugador.
7- El Organizador del Torneo pulsa el Aceptar del mensaje de confirmación.	8- El sistema elimina el jugador de la base de datos y muestra un mensaje de que ha sido eliminado, terminando así el caso de uso.
Flujo Alterno de Eventos	
	6- Si el jugador pertenece a un torneo que no haya finalizado se muestra un mensaje informando sobre la situación
7-El Organizador del Torneo pulsa el Cancelar del mensaje de confirmación.	

Capítulo 3. Análisis y Diseño

3.1 Introducción

El desarrollo de este capítulo se enmarca en las actividades que propone la metodología RUP para la fase de elaboración, fundamentalmente el flujo de trabajo de Análisis y Diseño, modelándose una serie de artefactos que contribuyen a la implementación de la propuesta del sistema. Se realizan los diagramas de clases del análisis y del diseño, así como los diagramas de colaboración para el análisis.

Durante el diseño se aplican los patrones GRASP, para la asignación de responsabilidades a las clases, además del patrón de arquitectura Modelo Vista Controlador, con el objetivo de separar en componentes diferentes los datos de la aplicación, la interfaz de usuario y la lógica de control.

3.2 Desarrollo

3.2.1 Arquitectura y estilos arquitectónicos

La arquitectura de software está formada por la estructura de los elementos de un programa o sistema, sus interrelaciones, los principios y reglas que gobierna su diseño y evolución a lo largo del tiempo. También es definida por David Garlan y Mary Shaw como un nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema".

Para desarrollar una arquitectura potente se utilizan estilos arquitectónicos, éstos indican los tipos de componentes y conectores involucrados, así como patrones y restricciones de interconexión o composición entre ellos.

En el desarrollo de la propuesta de sistema integrado de los módulos del proyecto Infodrez se utiliza el patrón arquitectónico Modelo – Vista – Controlador asociado al Estilo N Capas.

3.2.1.1 Modelo Vista Controlador.

El patrón Modelo – Vista – Controlador conocido como MVC, divide un componente o un subsistema en tres partes lógicas: modelo, vista y controlador, facilitando la modificación o personalización de cada parte. Se utiliza frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo

es el Sistema de Gestión de Base de Datos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista.

La separación de vistas, controladores y modelos permite realizar las siguientes labores de forma más sencilla:

- Agregar nuevas vistas.
- Agregar nuevas formas de recolectar las órdenes del usuario (interpretar sus modelos mentales).
- Modificar los objetos de negocios bien sea para mejorar el performance o para migrar a otra tecnología.
- Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas. Las correcciones se deben hacer en un solo lugar y no en varios como sucedería si tuviésemos una mezcla de presentación e implementación de la lógica del negocio.
- Las vistas también son susceptibles de modificación sin necesidad de provocar que todo el sistema se detenga.

3.2.1.2 Estilo N Capas

El Estilo N Capas se utiliza para descomponer un sistema en capas (Presentación, Lógica de Negocio y Acceso a Datos), posibilitando el manejo de su complejidad. Cada una de estas debe ocuparse de un nivel específico del problema y debe tener poca cohesión con las demás. Este estilo arquitectónico presenta algunas ventajas como la reutilización de las capas, además de que el cambio en una de ellas no afectaría en gran medida las demás capas.

3.2.2 Modelo de análisis

El modelo de análisis es la primera representación técnica del sistema, por lo que puede considerarse como una primera aproximación al modelo de diseño, y es por tanto una entrada fundamental para las actividades de diseño e implementación subsiguientes. A partir de su realización se refinan y estructuran los requisitos obtenidos con anterioridad proporcionando una visión general del sistema.

El modelo de análisis debe lograr tres objetivos primarios:

- Describir lo que requiere el cliente.
- Definir un conjunto de requisitos que se pueda validar una vez que se construye el software.

- Establecer una base para la creación de un diseño de software.

3.2.2.1 Diagrama de clases del análisis

El diagrama de clases de análisis es un artefacto en el que se representa los conceptos fundamentales en un dominio del problema. Para una mejor comprensión de esta parte, se mostrarán cada uno de los diagramas generados dentro del análisis, específicamente los diagramas de clases y los diagramas de colaboración, compuestos por clases interfaz, clases controladoras y clases entidades.

Clases interfaz: Modelan la interacción entre el sistema y sus actores.
Clases controladoras: Coordinan la realización de varios casos de uso, controlando las actividades de los objetos que implementan la funcionalidad del caso de uso y la interacción entre las clases interfaz y las clases entidades.
Clases entidades: Modelan información que posee larga vida y que es a menudo persistente
(Jacobson, 2004).

Diagrama de Clases de Análisis CUS Autenticar Usuario

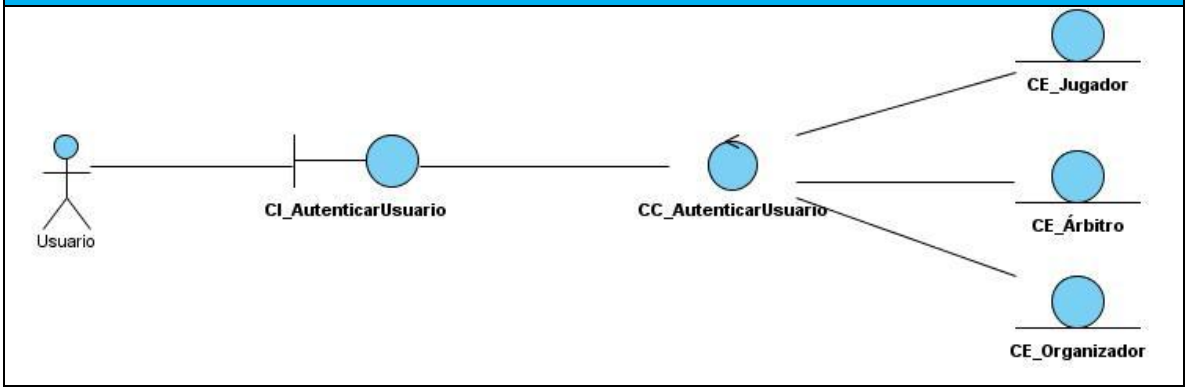


Figura 3.1 Diagrama Autenticar Usuario

Diagrama de Clases de Análisis CUS Expulsar Jugador

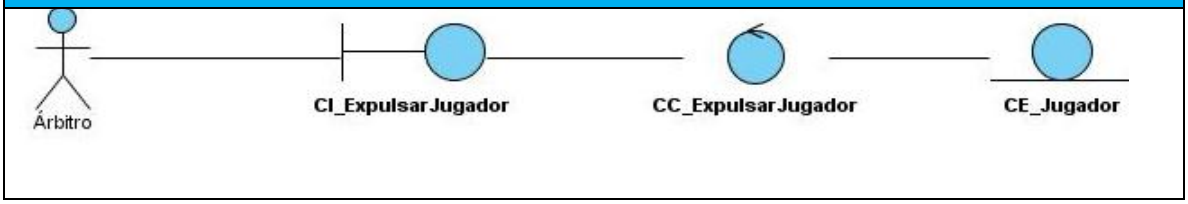


Figura 3.3 Expulsar Jugador

Diagrama de Clases de Análisis CUS Gestionar Torneo

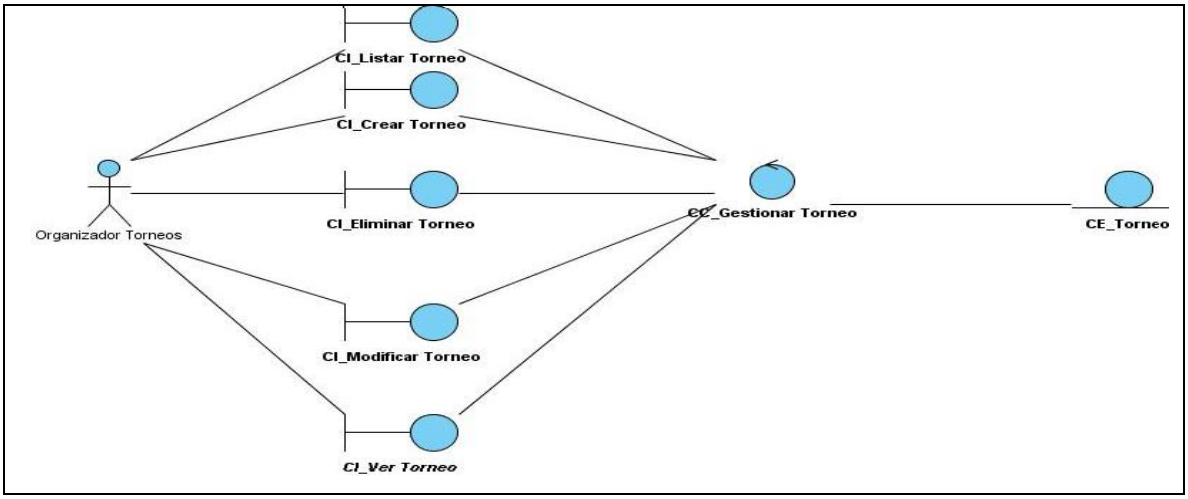


Figura 3.2 Diagrama Gestionar Torneo

Diagrama de Clases de Análisis CUS Gestionar Jugador

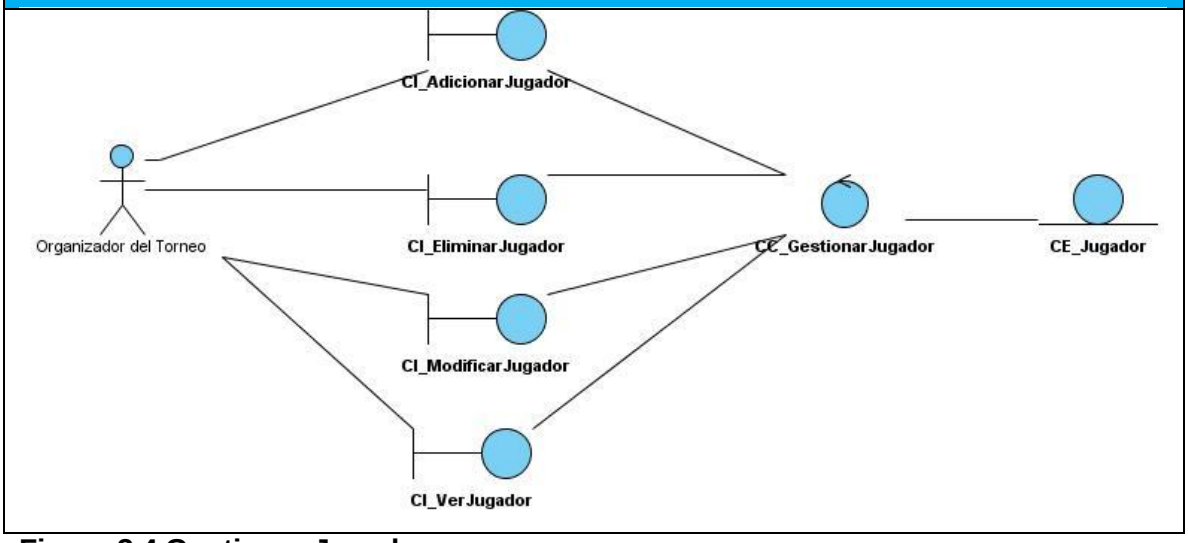


Figura 3.4 Gestionar Jugador

3.2.2.2 Realización de los casos de uso del análisis

La realización de los casos de uso del análisis describe como se lleva a cabo y se ejecuta un caso de uso en término de las clases del análisis y de sus objetos en interacción. Se realiza a través de dos tipos de diagramas: diagrama de secuencia y de colaboración.

En el presente trabajo utilizaremos el Diagrama de Colaboración, este ofrece una mejor visión espacial exponiendo los enlaces de comunicación entre objetos, muestra las relaciones entre objetos y son mejores para comprender todos los efectos que tiene un objeto para el diseño de procedimientos. El diagrama de Colaboración puede obtenerse automáticamente a partir del correspondiente diagrama de Secuencia(o viceversa).

Diagrama de Colaboración del Análisis: Autenticar Usuario

Figura 3.5 Autenticar Usuario

Diagrama de Colaboración del Análisis: Adicionar Jugador

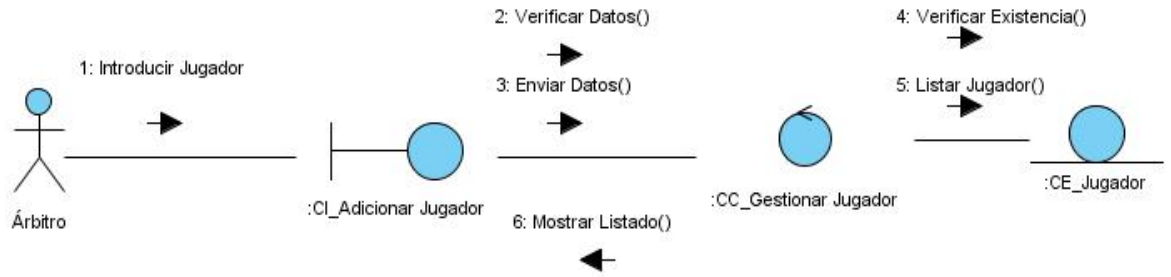


Figura 3.6 Gestionar Jugador: Sección Adicionar Jugador.

Diagrama de Colaboración del Análisis: Modificar Jugador

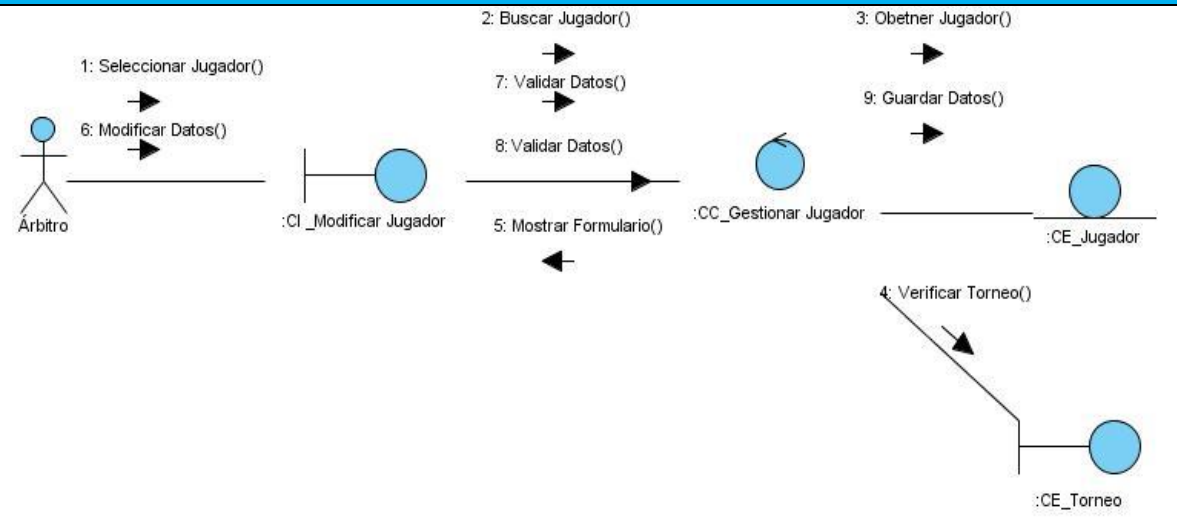


Figura 3.7 Gestionar Jugador: Sección Modificar Jugador

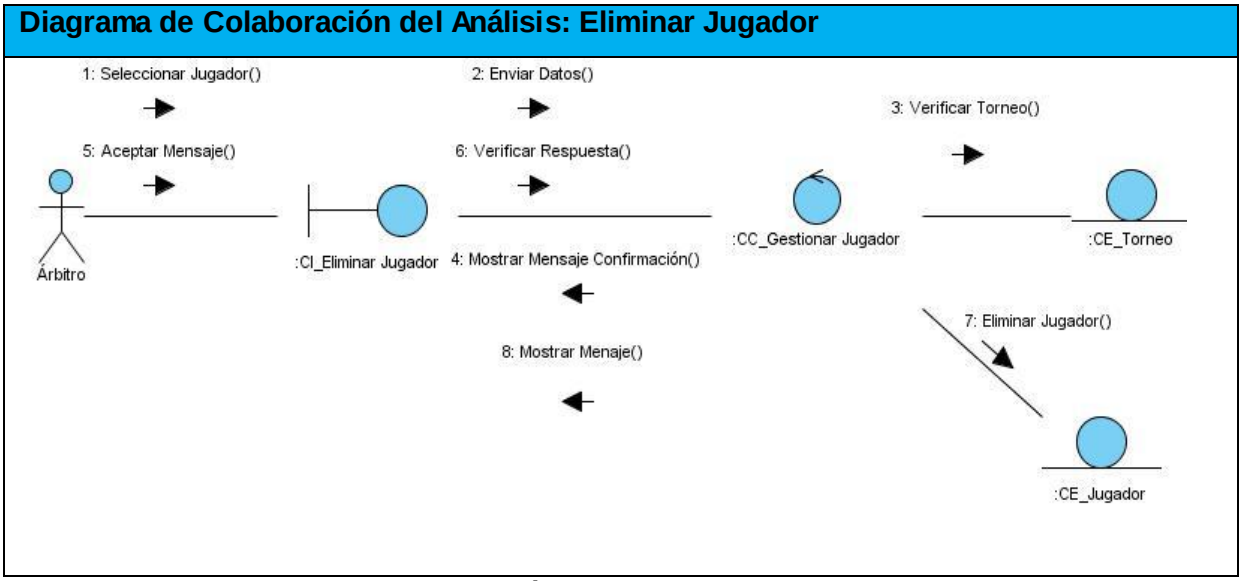


Figura 3.8 Gestionar Jugador: Sección Eliminar Jugador

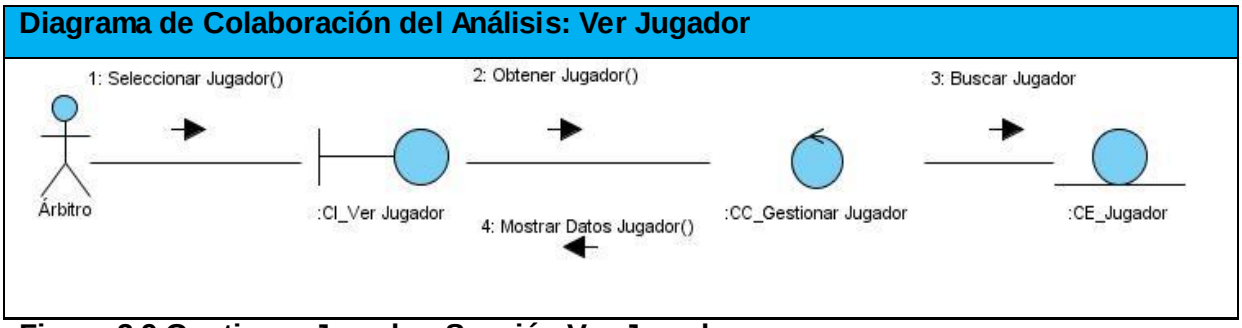


Figura 3.9 Gestionar Jugador: Sección Ver Jugador

3.3.3 Modelo de Diseño

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales. Es el centro de atención final de la fase de elaboración y el comienzo de las iteraciones de construcción, modela el sistema y se le da forma (incluida la arquitectura) para que soporte todos los requisitos y las restricciones que se le suponen. Además impone una estructura que se debe conservar lo más fielmente posible cuando se le dé forma al sistema.

Constituye la entrada fundamental al flujo de Implementación. En el diseño las clases definidas en el análisis se detallan y se añaden nuevas clases para manejar áreas técnicas como base de datos, interfaz del usuario, comunicación, dispositivos, entre otras.

3.3.3.1 Clases del Diseño

Una clase de diseño es una construcción similar en la implementación del sistema. Los diagramas de clases de diseño exponen un conjunto de interfaces, colaboraciones y sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema. Los

diagramas de clases de diseño del sistema están separados en casos de usos, es decir, cada caso de uso cuenta con un diagrama de clases, para hacer más fácil la comprensión.

En la propuesta de sistema integrado de los módulos del proyecto Infodrez se realiza el diagrama de clases del diseño utilizando los estereotipos Web que propone UML: “Server Page”, “Client Page” y “Form”, empleados para el código servidor, código cliente y formularios respectivamente.

Diagrama de Clases del Diseño: Autenticar Usuario

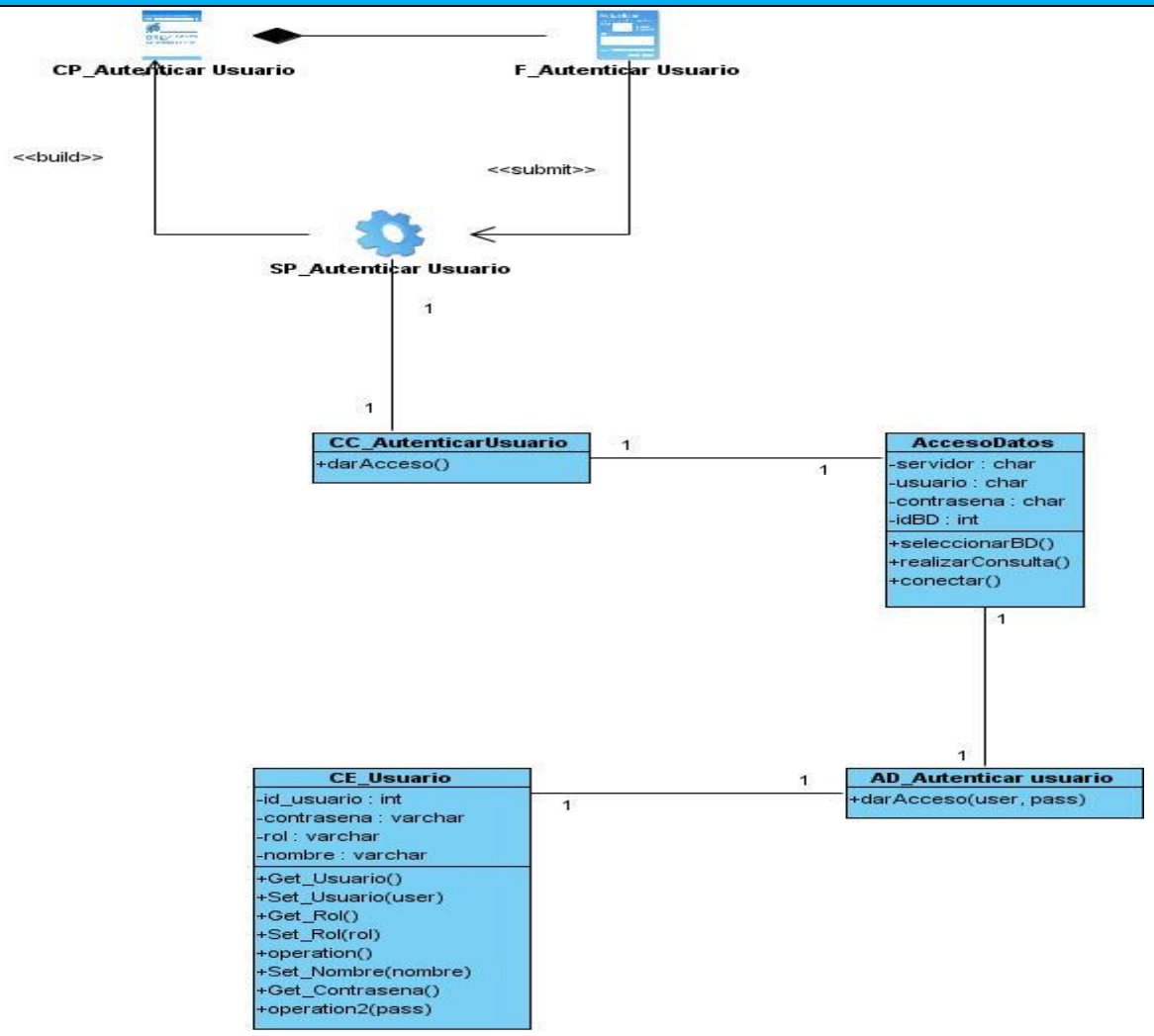


Figura 3.10 Diagrama de Clases del Diseño Autenticar Usuario

Diagrama de Clases del Diseño: Gestionar Torneo

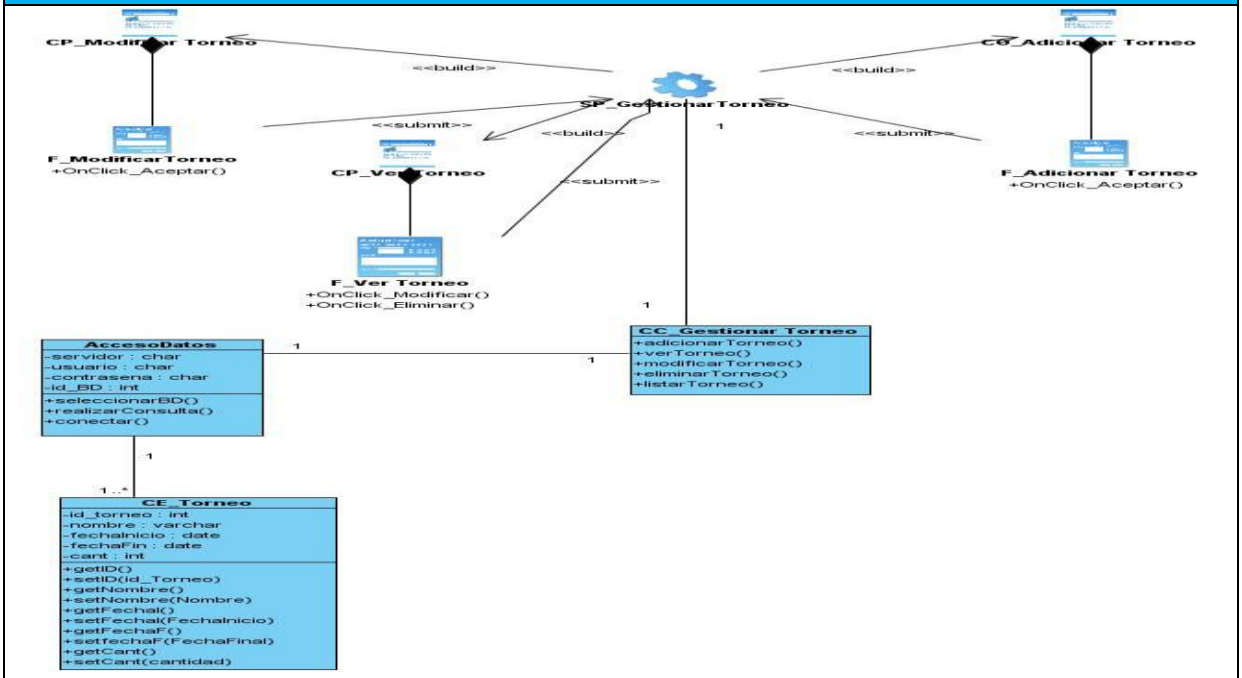


Figura 3.11 Diagrama de Clases del Diseño Gestionar Torneo

Diagrama de Clases del Diseño: Adicionar Ritmo de Juego

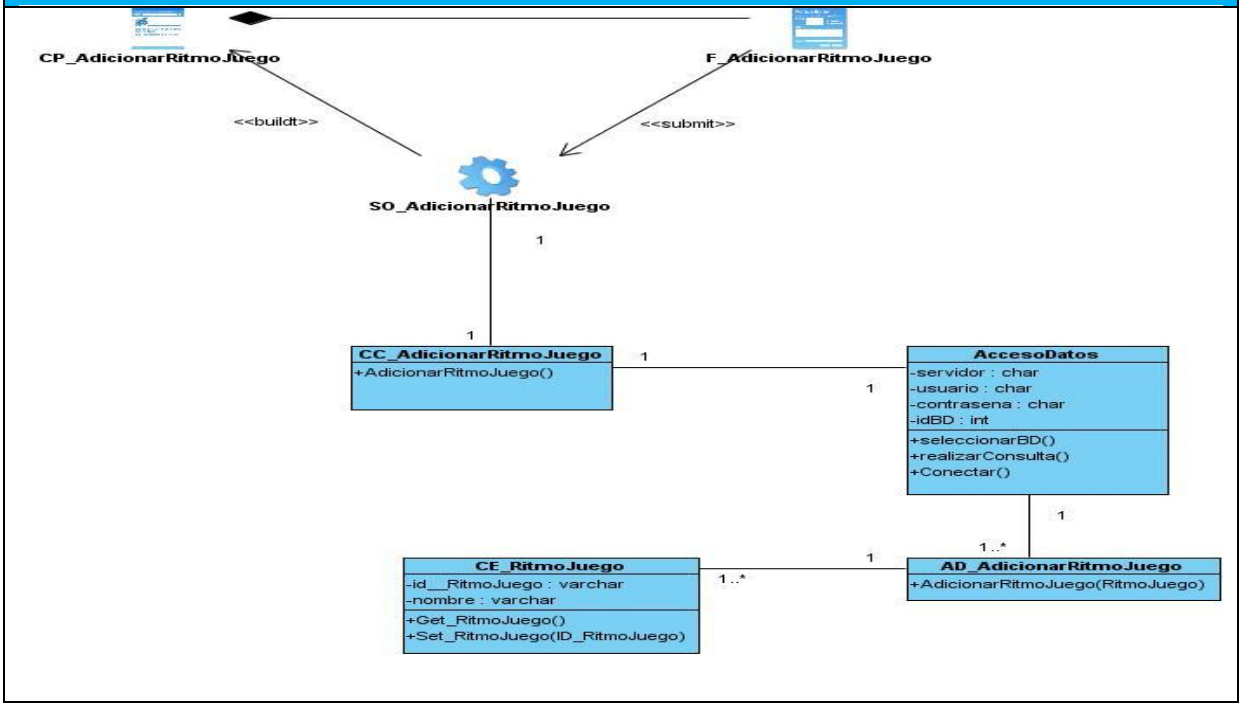


Figura 3.12 Diagrama de Clases del Diseño Adicionar Ritmo de Juego

Diagrama de Clases del Diseño: Gestionar Jugador

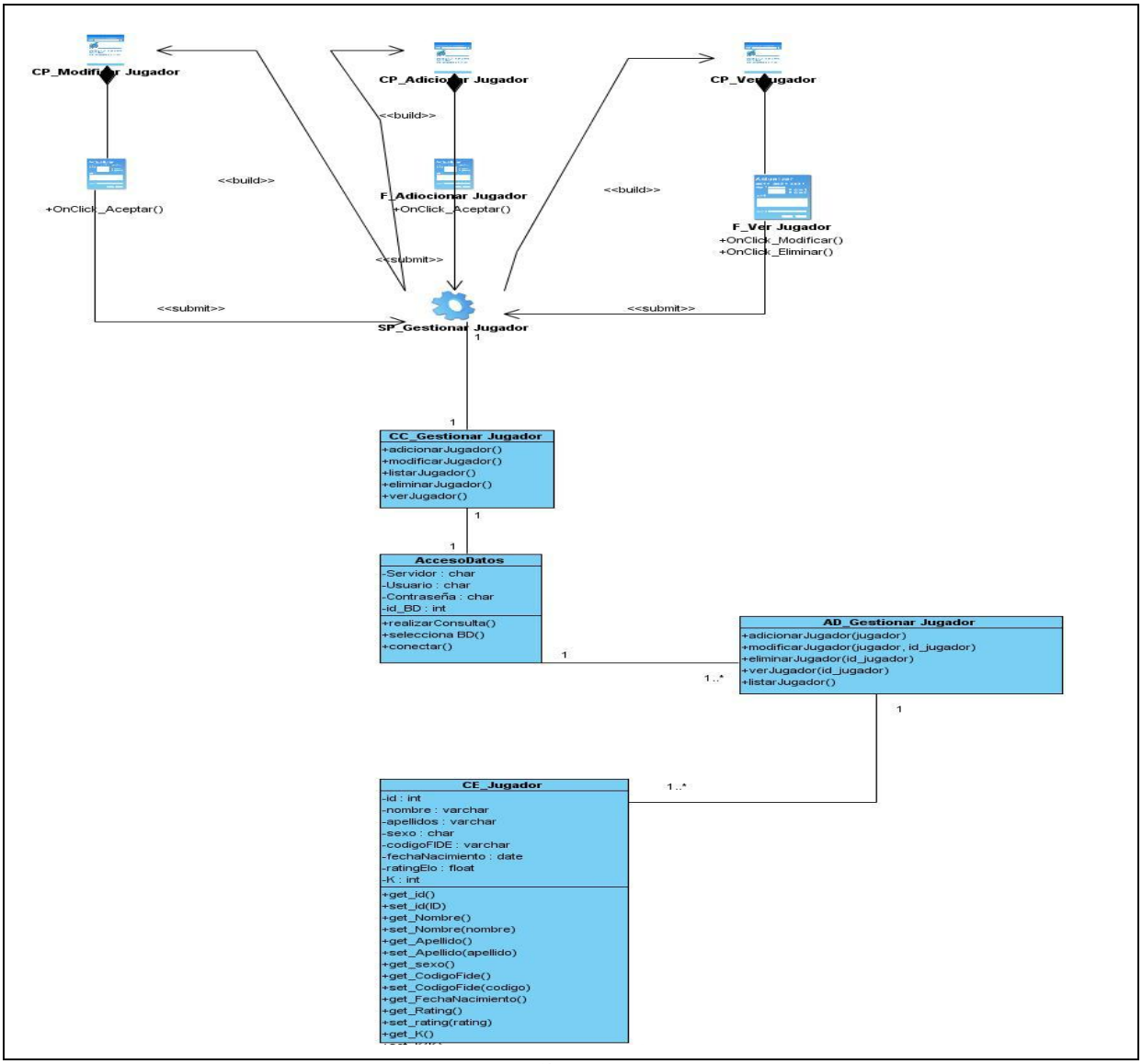


Figura 3.12 Diagrama de Clases del Diseño Gestionar Jugador

3.3.4 Diagrama de clases persistentes

Las clases persistentes son capaces de guardar su estado en un medio permanente, debido a que sus objetos pueden mantener su valor en el espacio y el tiempo, la necesidad de guardar su estado está dada por el almacenamiento físico permanente 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. Deben ser almacenados en algún repositorio como una base de datos.

Diagrama de Clases Persistentes

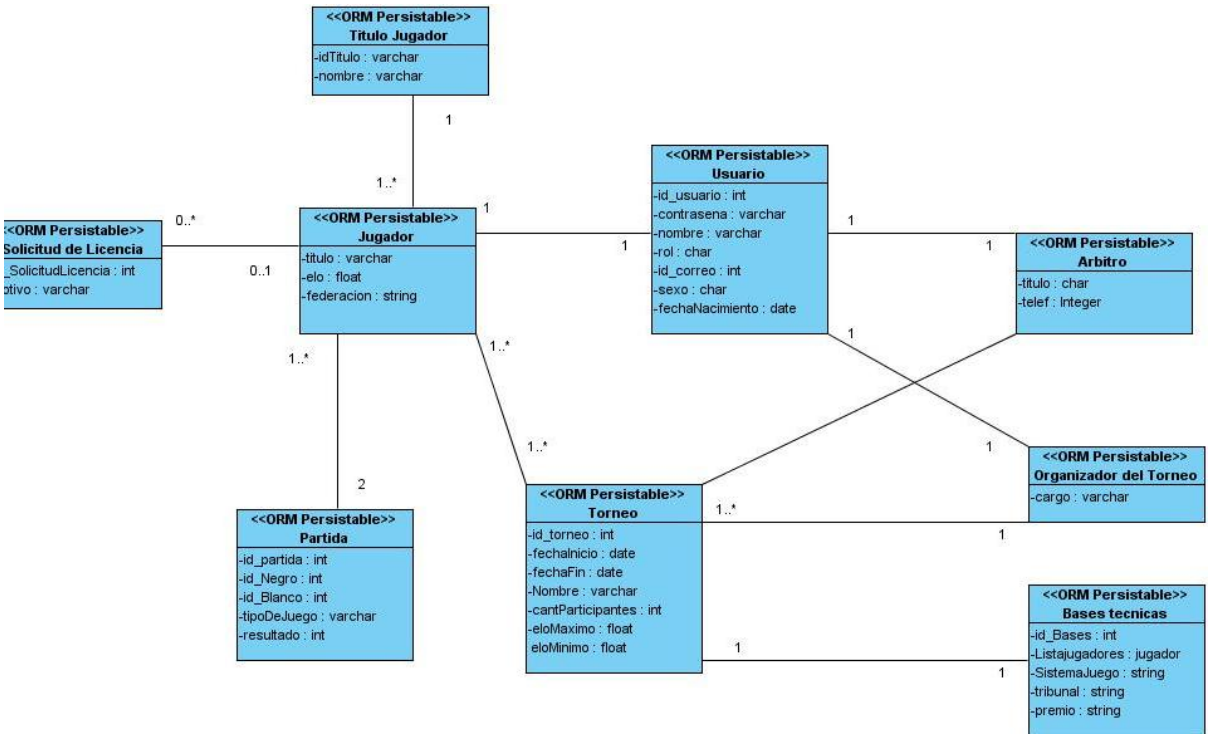


Figura 3. Diagrama de Clases Persistentes

3.3.5 Conclusiones

En el presente capítulo fueron expuestos los distintos elementos que muestran la manera en que se encuentra construido el sistema, en términos de clases del análisis, del diseño y los diagramas de interacción, dando la posibilidad de tener una mejor comprensión de la lógica del mismo. Se construyeron varios artefactos para llevar a cabo el proceso de implementación del sistema. Se utilizaron los patrones GRASP en el diseño para la asignación de responsabilidades a las clases, logrando de ellas una mayor organización y entendimiento, además del patrón arquitectónico Modelo Vista Controlador para lograr la separación de los distintos componentes: los datos de la aplicación, la interfaz de usuario y la lógica de control. Se utilizaron diagramas de clases Web para explicar lógica del negocio del sistema. También fue presentado el diagrama de clases persistentes de la base de datos, que contiene la información que se utilizó para la construcción de la aplicación

Conclusiones Generales

Después de finalizada la propuesta se obtuvo como resultado un sistema integrado con los módulos Juego Online y Ajedrez por Correspondencia del proyecto Infodrez en su segunda versión. Se realizó el modelo de negocio, artefacto que tiene como objetivo conocer la estructura dinámica de la organización en la cual se va a implantar el sistema, además de describirse las reglas del negocio presentes. Se capturaron los requisitos del sistema y se validaron cada uno de ellos a través de diversas técnicas. Se cumplieron cada una de las tareas definidas al comenzar el desarrollo de este trabajo, arribando a las siguientes conclusiones:

- El sistema modelado soluciona los principales problemas que presenta la Cátedra de Ajedrez de la Universidad con la integración Juego Online y el Ajedrez por Correspondencia. El diseño de esta integración constituía una necesidad, debido a que las aplicaciones existentes del proyecto solo gestionaban información referente a las áreas del ajedrez de forma separada.

- Las técnicas de captura de requisitos brindaron la posibilidad de definir de forma clara las funcionalidades del sistema que sería diseñado.

- La metodología de desarrollo utilizada unida al lenguaje de modelado UML y a la herramienta Visual Paradigm ayudó a guiar el proceso de diseño de la propuesta del sistema de los módulos del proyecto Infodrez, además de permitir que el trabajo se desarrollara con mayor eficiencia.

- El diseño realizado permitirá que los desarrolladores de la propuesta de sistema se sientan guiados hacia una implementación sin ambigüedades.

Se puede concluir que se le ha dado cumplimiento al objetivo propuesto ya que se diseñó una aplicación informática para automatizar los procesos de unión del Juego Online y Ajedrez por Correspondencia, facilitando un mejor entendimiento para los diseñadores de base de datos e implementadores.

Recomendaciones

- Mostrar en un gráfico el comportamiento de cada jugador durante un torneo.
- Incorporar la Ingeniería del Módulo Simultánea a la propuesta de sistema.
- Que se implementen en la aplicación los módulos propuestos.

Referencias Bibliográficas

1- **Proyecto Infodrez.** *Documento Plan de Desarrollo de Software***.** 2007

2- **Postal Federación Cubana de Ajedrez.** *Reglamento General FECAP.* 2005.

Bibliográfica

1- **Proyecto Infodrez.** *Documento Plan de Desarrollo de Software.*

2- **Postal Federación Cubana de Ajedrez.** *Reglamento General FECAP.* 2005.

3- **Jacobson, Ivar y Booch, Grady and Rumbaugh, James.** *El proceso unificado de desarrollo de software.* La Habana : Editorial Félix Varela, 2004. Volumen I.

5- **Larman, Craig**; "UML y patrones" Tomo I Capítulo 4, Páginas 185-217.

6- **Torossi, A.U.S Gustavo**. *El Proceso Unificado del Software*. [Online][Citado el 01 abril de 2010] http://antares.itmorelia.edu.mx/~jcolivar/courses/pm10a/rup.pdf.

7- **Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 1999.** *The Unified Software Development Process.* s.l. : Addison-Wesley, 1999. ISBN: 0201571692.

8. **Entorno Virtual de Aprendizaje**. (s.f.). Recuperado el 5 de marzo de 2010, de Fase de inicio. Modelo del Negocio: http://eva.uci.cu/mod/resource/view.php?id=21010**.** http://librosweb.es/javascript/capitulo1.html.

9- **Pressman, Roger S. 2005.** *Ingeniería del Software un enfoque práctico.* La Habana : Felix Varela, 2005. pp. 171-187.

10- **Rational Software, Corporation. 2003.** *Rational Unified Extended Help.* 2003. Rational Unified Process: Overview.

11- **Visual Paradigm for UML**. *Visual Paradigm for UML.* [Online][Citado el 15 de febrero de 2010] Visual Paradigm. http://www.visual-paradigm.com/product/vpuml]

12- **Dondo, Agustín 2002.** PHP en Castellano. [Online][Citado el 15 de febrero de 2010]. http://www.programacion.com/php/articulo/porquephp/

13- **Eguíluz Pérez, Javier.** Librosweb. [Online][Citado el 15 de febrero de 2010]. http://librosweb.es/javascript/capitulo1.html

14- **Innovación y Cualificación, S.L. 2001.** *JavaScript.* Malaga : INNOVACION Y CUALIFICACION, S.L., 2001. ISBN: 84-95733-18-8.

15- **Martín, Eloi de San. 2006.** Programaciónweb. [Online] [Citado 23 de enero 2010]. http://www.programacionweb.net/articulos/articulo/?num=184.

17**. Mendoza Sanchez, María A. 2004.** [Online][Citado 20 enero 2010]. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

18. **Ercoli, Jorge.** *Arquitectura de Sistemas Informáticos*. [Online] 28 de mayo de 2007. [Citado el: 01 de abril de 2010] http://metodologiasdesistemas.blogspot.com/2007/05/diseo-en-3-capas-fisicas-logicas-es.html.

Anexos

Anexo 1. Descripción de casos de Uso

Descripción del caso de uso Mostrar Torneos Disponibles

Caso de Uso	**Mostrar Torneos Disponibles**
Actores	Jugador
Propósito	Mostrar un listado con los torneos disponibles en el Sistema hasta el momento.
Resumen	El CUS se inicia cuando el Jugador solicita al sistema mostrar todos los torneos que se encuentran disponibles, el sistema realiza una consulta a la base de datos en la tabla de torneos y muestra una lista con los torneos que hay disponibles hasta el momento.
Referencias	RF 18, RF 18.1
Precondiciones	Debe existir al menos un torneo creado en el Sistema.
Poscondiciones	Muestra un listado con todos los torneos disponibles y el jugador se inscribe en el de su preferencia.

Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Solicita la opción de Mostrar Torneos Disponibles.	2- Realiza una consulta a la base de datos a la tabla de Torneos. 3- Muestra una lista con los torneos disponibles.
	4- El caso de uso termina.

Curso alterno de los eventos	
Prioridad	Crítico

Descripción del caso de uso Definir Partidas no Concluidas

Caso de Uso	**Definir Partidas no Concluidas**
Actores	Árbitro
Propósito	El árbitro pueda decidir las partidas que se queden aplazadas y se hayan pasado del tiempo límite.
Resumen	El CUS se inicia cuando el Árbitro selecciona la opción de Definir Partidas no Concluidas, el sistema realiza la acción y termina el CUS.
Referencias	RF 17
Precondiciones	Existe la necesidad de definir partidas aplazadas por mucho tiempo.
Poscondiciones	Se definen las partidas aplazadas por el Árbitro.

Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1- Se inicia el caso de uso cuando el Árbitro solicita definir las partidas no concluidas. 3- Selecciona las partidas que han sobrepasado el límite de tiempo	2- El sistema muestra todas las partidas sin definir. 4- El sistema muestra la opción definir.
	5- El caso de uso Termina.

Descripción caso de uso Observar Juego

Caso de Uso	**Observar Juego**
Actores	Jugador
Propósito	Observar partidas de los jugadores
Resumen	El CUS se inicia cuando el Jugador no está jugando ninguna partida y

	desea ver otras partidas online
Referencias	RF 8
Precondiciones	Debe existir partidas creadas
Poscondiciones	Observar las partidas creadas
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
Prioridad	Secundario

Descripción del caso de uso Mostrar Torneos Disponibles

Caso de Uso	**Gestionar Partida**
Actores	Director del Torneo
Propósito	Que el Director del Torneo tenga la posibilidad de Gestionar las partidas en el Sistema ya sea Crearlas o Actualizarlas.
Resumen	El CUS se inicia cuando el Director del Torneo selecciona la opción de Gestionar Partida, luego selecciona el tipo de gestión, introduce los datos necesarios, el sistema realiza la acción seleccionada por el Director del Torneo y termina el CUS.
Referencias	RF 7, RF 7.1, RF 7.2
Precondiciones	Existe una necesidad de gestionar una partida.
Poscondiciones	Se Gestiona la acción seleccionada por el Director del Torneo
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**

1- Se inicia el caso de uso cuando el director del torneo solicita Gestionar una Partida. 3- Selecciona una de las opciones que existen.	2- El sistema muestra dos opciones crear partida y actualizar partida. 4- El sistema muestra según la opción solicitada Crear Partida: Sección Crear Partida, Actualizar Partida: Sección Actualizar Partida.
	5- El caso de uso Termina.
Sección : Crear Partida	
1- Solicita la opción de Crear Partida.	2-Realiza una consulta a la tabla de jugador en la base de datos y solicita los datos del jugador. 3-Realiza el Pareo con los datos del jugador. 4-Registra los datos de la partida según el pareo efectuado creándose la partida, y muestra el pareo.
	5- El caso de uso termina.
Sección: Actualizar Partida	
1- Solicita actualizar una Partida. 4- Selecciona la partida que desea actualizar. 6- Director del Torneo: Actualiza los datos	2-Realiza una consulta a la base de datos, a la tabla de partidas y solicita todos los partidas creados hasta el momento. 3-Muestra una interfaz con todos las partidas. 5-Muestra una interfaz con los datos de la

que desee de la partida.	Partida. 7-Actualiza en la base de datos los campos seleccionados para cambiar.
	8- El caso de uso termina.
Curso alterno de los eventos	
	3- Si no existen torneos creados se envía un mensaje informando que no existen torneos en el sistema. 4-El caso de uso finaliza.
Prioridad	Crítico

Anexo 2 Diagramas de Clases del Diseño

Diagrama de Clases del Diseño: Gestionar Árbitro

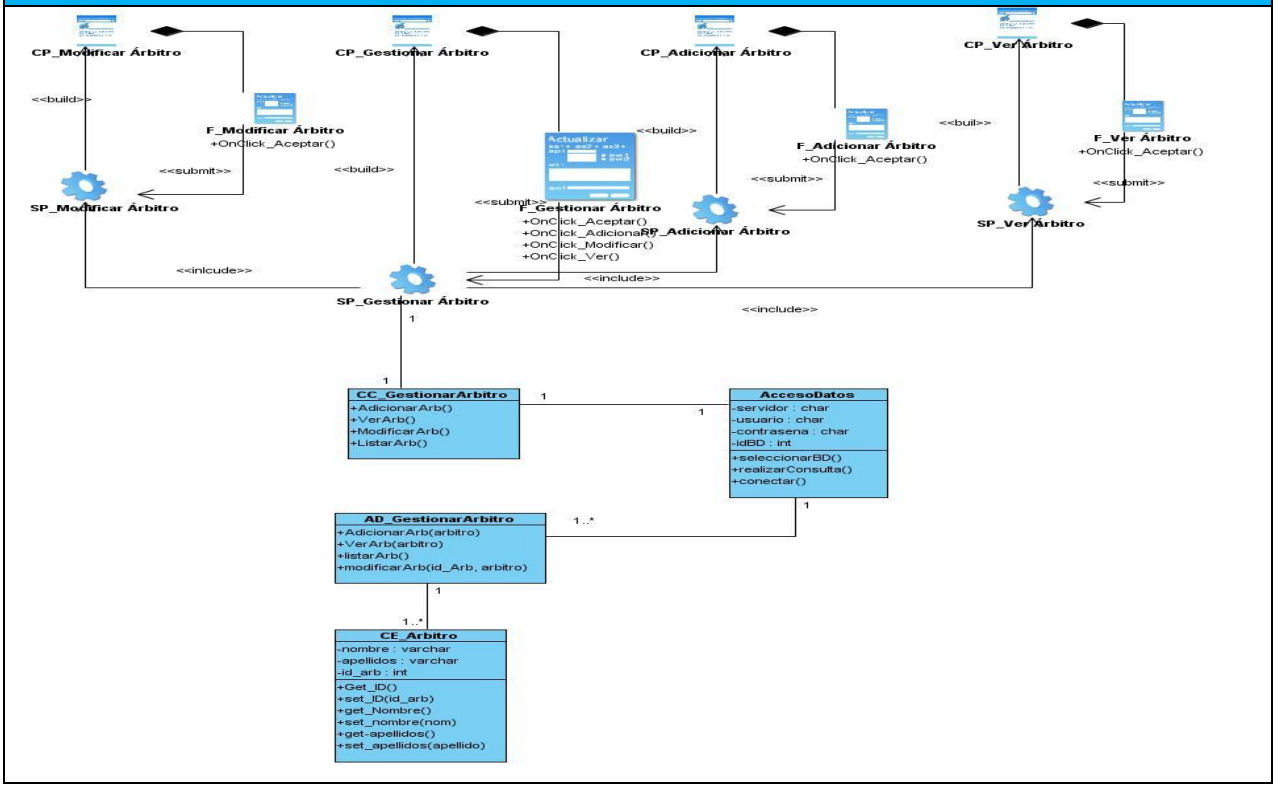


Diagrama de Clases del Diseño: Gestionar Título del Jugador

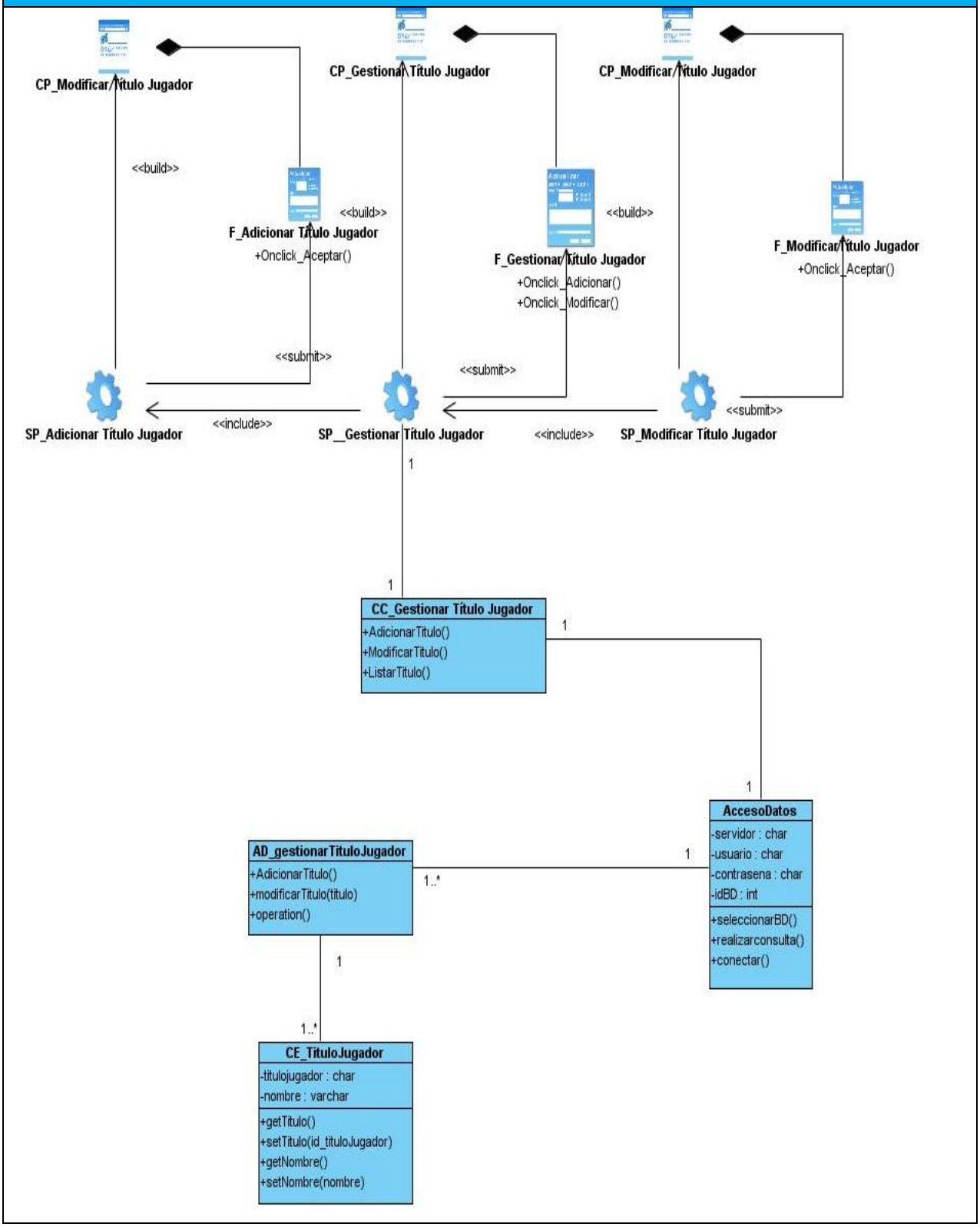
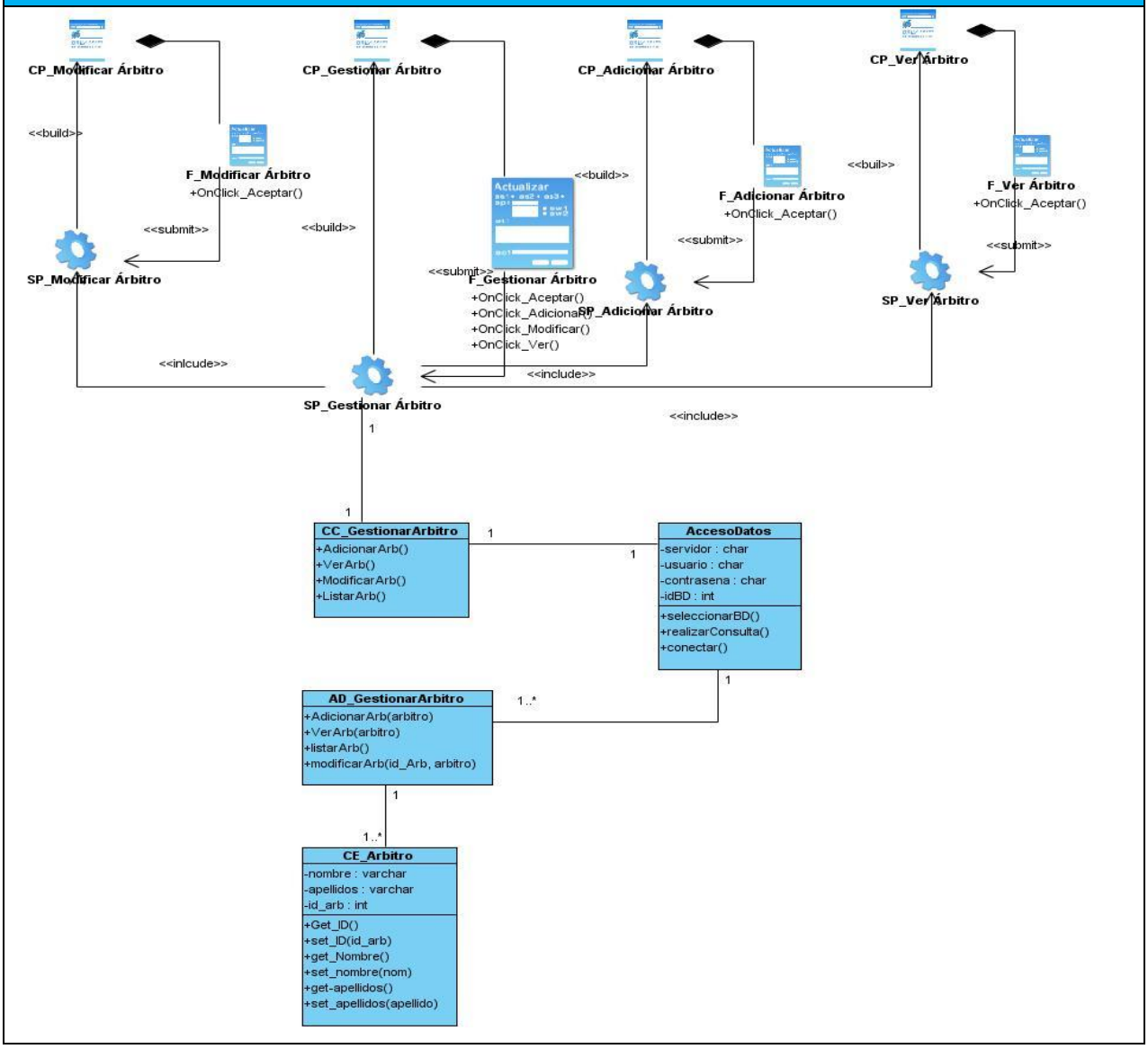
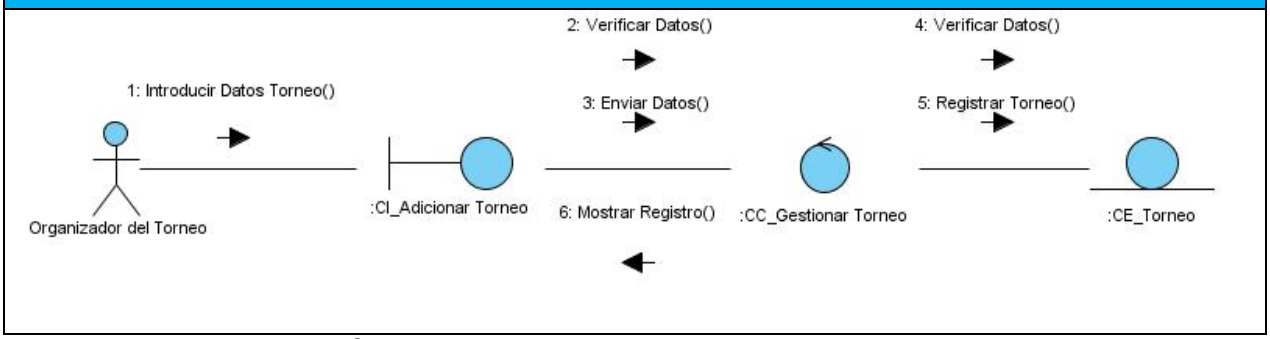


Diagrama de Clases del Diseño: Gestionar Título Árbitro



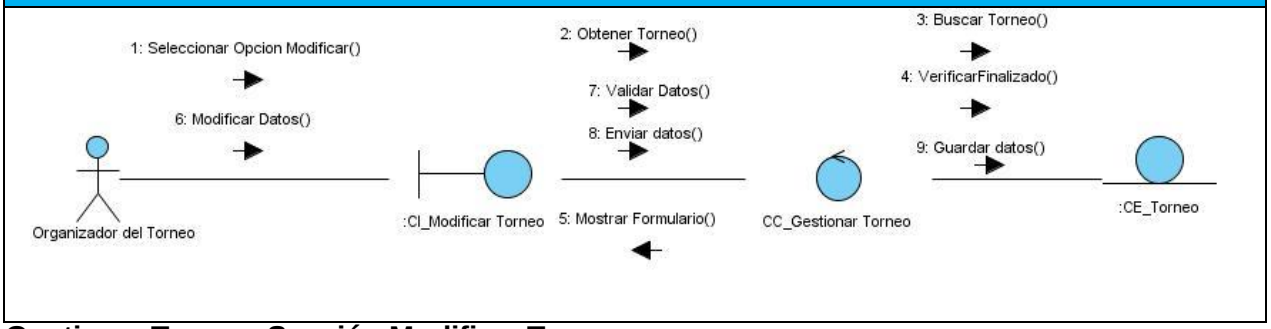
Anexo 3. Diagramas de Colaboración

Diagrama de Colaboración del Análisis Adicionar Torneo



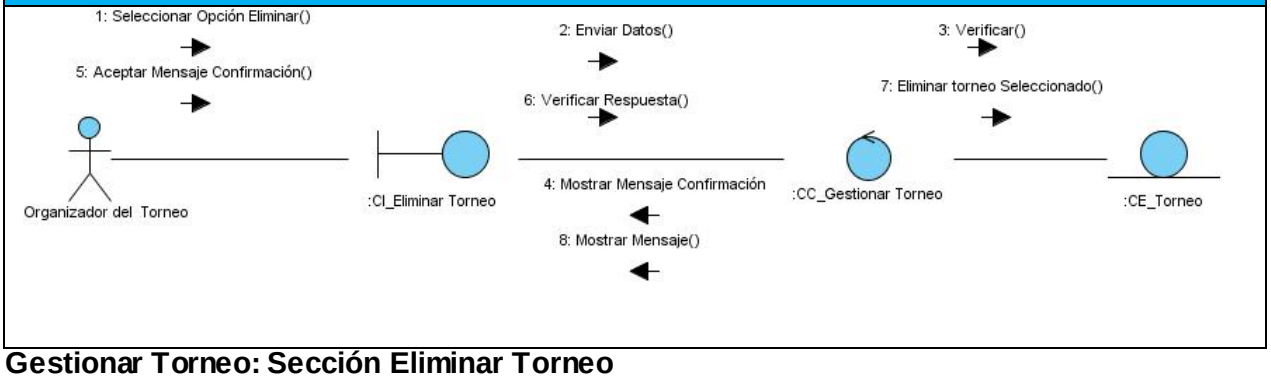
Gestionar Torneo: Sección Adicionar Torneo

Diagrama de Colaboración del Análisis Modificar Torneo



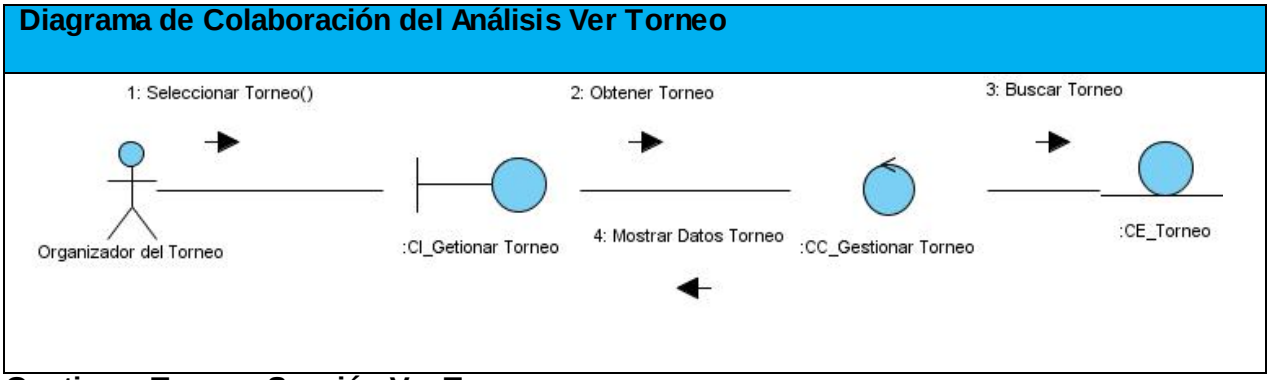
Gestionar Torneo: Sección Modificar Torneo

Diagrama de Colaboración del Análisis Eliminar Torneo



Gestionar Torneo: Sección Eliminar Torneo

Diagrama de Colaboración del Análisis Ver Torneo



Gestionar Torneo: Sección Ver Torneo

Glosario de Términos

Autenticación: en la seguridad de ordenador, la autenticación es el proceso de intento de verificar la identidad digital del remitente de una comunicación como una petición para conectarse. En un web de confianza, "autenticación" es un modo de asegurar que los usuarios son quién ellos dicen que ellos son - que el usuario que intenta realizar funciones en un sistema es de hecho el usuario que tiene la autorización para hacer así.

Debian: es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre, en un formato sencillo en múltiples arquitecturas de computador y en varios núcleos.

EJB (Enterprise JavaBeans): son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Oracle - Sun Microsystems (ahora JEE 5.0).

Elo: 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.

FECAP: Federación Cubana de Ajedrez Postal, organismo rector del juego ciencia a distancia en el país y agrupa en su seno a los amantes de esta disciplina.

HTML: Acrónimo inglés de Hyper Text Markup Language (lenguaje de marcación de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Este lenguaje se basa en tags (instrucciones que le dicen al texto como deben mostrarse) y atributos (parámetros que dan valor al tag). Es el estándar usado en el World Wide Web.

IEEE: corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización

JavaScript; Lenguaje desarrollado por *Netscape*. Aunque es parecido a *Java* se diferencia de él en que los programas están incorporados en el fichero HTML.

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.

RUP: El Proceso Racional Unificado o RUP (Rational Unified Process), es un proceso de desarrollo de software.

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modelling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por la OMG (Grupo dedicado a la promoción de la tecnología orientada a objetos y su estandarización).

.