Universidad de las Ciencias Informáticas

Facultad 8

Análisis, diseño e implementación del sistema para la planificación del entrenamiento del judo femenino

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores:

Yailín González Rivas

Frank Ismauri Pérez Rodríguez

Tutores:

Ing. Aliuska Sánchez Ibarria

Ing. Karenia Donatien Goliath

Ciudad de La Habana, Junio del 2009

"Año 50 del triunfo de la Revolución"

Declaración de autoría

Declaramos ser autores del trabajo de diploma Análisis, diseño e implementación del sistema para la planificación del entrenamiento del judo femenino, y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los _____ días del mes de _____ del año _____.

_____	_____
Firma del autor	Firma del autor
Yailín González Rivas	Frank Ismauri Pérez Rodríguez
_____	_____
Firma del tutor	Firma del tutor
Ing. Aliuska Sánchez Ibarria	Ing. Karenia Donatien Goliath

Agradecimientos

Agradecemos a todas las personas que de una forma u otra nos ayudaron a seguir adelante, especialmente a:

Nuestras tutoras Karenia Donatien y Aliuska Sánchez por su apoyo incondicional, y sus valiosas opiniones y comentarios, además de toda la ayuda brindada.

A todo el tribunal por su apoyo.

A todos los profesores que nos han enseñado con dedicación a lo largo de estos años.

A nuestros compañeros durante los cinco años de carrera.

A toda nuestra familia por estar siempre a nuestro lado, en especial a nuestros padres por su amor y apoyo.

Y a la Revolución por habernos dado la posibilidad de estudiar en esta universidad, en especial a nuestro Comandante Fidel Castro y a Raúl.

Muchas Gracias

Dedicatoria

A toda mi familia por la ayuda y el apoyo que me dieron, en especial a mis padres Ana Gloria y Alberto por su amor y confianza.

A mi novio y compañero de tesis Frank por estar a mi lado en todo momento ya que sin él no hubiera sido posible la realización de este trabajo.

A Reyna y Francisco por su apoyo y confianza.

A todos aquellos que de una forma u otra me ayudaron.

Yailín

A mis excelentes padres y a mi hermano que los quiero mucho por tener de ellos todo el apoyo, amor y confianza en todos estos años.

A mi novia y compañera de tesis Yailín por ser la fuente de mi inspiración, cariño, amor y confianza.

A mis amigos, que me dieron su apoyo en todo momento.

A todos aquellos que de una forma u otra me ayudaron a ser lo que soy.

Frank

Resumen

El Instituto Nacional de Deportes, Educación Física y Recreación (INDER) no cuenta actualmente con un sistema que permita gestionar la planificación del entrenamiento para el judo femenino. Por lo que dicha planificación se les hace muy difícil a los entrenadores, esto trajo consigo que manifestaran la necesidad de crear un sistema que les facilitara la realización de este proceso. El presente trabajo propone la elaboración de un sistema que gestione la planificación del entrenamiento, dando solución de esta manera a los problemas que existen. El sistema está compuesto por cuatro módulos. El módulo administración es donde se gestionarán los usuarios. El módulo plan gráfico permitirá realizar la planificación de las fechas de las pruebas médicas, competencias, campos de entrenamiento y test pedagógicos. Además estará dividido por macrosistemas, mesosistemas y microsistemas para una mayor organización. En el módulo test pedagógico se evaluarán pruebas de fuerza, resistencia, velocidad, flexibilidad, prueba teórica y técnica. En el módulo atleta se gestionarán los datos de todas las atletas. Para la realización de la propuesta se utilizó PHP como lenguaje de programación por parte del servidor, HTML y CSS por parte del cliente, RUP como metodología de desarrollo, UML como lenguaje de modelado, *Visual Paradigm* como herramienta de modelado, *JavaScript* para las validaciones del lado del cliente y como IDE el *Eclipse*, permitiendo de esta manera la creación de un sistema que posibilitará a los entrenadores almacenar toda la información que antes no podían guardar, haciendo posible su consulta en cualquier momento.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Cuba es conocida mundialmente por la buena calidad de sus atletas y sus victorias en muchas disciplinas deportivas. Esto se debe al gran entrenamiento y preparación que poseen los deportistas y a la entrega incondicional de los excelentes entrenadores con los que cuenta el Instituto Nacional de Deporte, Educación Física y Recreación (INDER), los cuales han desempeñado un magnífico papel en el rendimiento, preparación y logros de los atletas cubanos. Logros que cada año tratan de mejorar, para ello muchas personas han puesto todo su empeño en el desarrollo de nuevas técnicas y herramientas que faciliten la labor de entrenadores, especialistas de deporte y deportistas.

La planificación del entrenamiento deportivo representa el objetivo de acción que se realiza con el desarrollo del entrenamiento de un atleta, cuya meta es lograr la forma óptima del deportista (Manso 2009). Un plan de entrenamiento deportivo es un proceso previsto, organizado, metódico, sistemático, científico, encargado de ordenar, sincronizar e integrar racionalmente a corto y/o largo plazo el contenido y estructura del entrenamiento deportivo y de todas las medidas necesarias y medios disponibles que conducen a la realización efectiva de un entrenamiento y al desarrollo óptimo del rendimiento deportivo (Corsino 2000). Este plan constituye un proceso muy engorroso para los entrenadores de los diferentes deportes, pues para su elaboración es necesario realizar una serie de cálculos y búsquedas de informaciones las cuales muchas veces se encuentran almacenadas en tablas dentro de documentos que no están digitalizados. Además el plan gráfico y los test pedagógicos que se encuentran dentro del plan de entrenamiento se caracterizan por ser muy complejos, lo que hace muy difícil su realización de forma manual, provocando la pérdida de datos importantes para la planificación.

Teniendo en cuenta la situación actual surge el **problema científico**: ¿Cómo facilitar el proceso de gestión de la planificación del entrenamiento deportivo para el judo femenino?

De lo planteado anteriormente se deriva como **objeto de estudio** el proceso de informatización de la planificación del entrenamiento deportivo.

Enfocando la investigación hacia el **campo de acción** de planificación del entrenamiento deportivo en la Escuela Cubana de Judo Femenino.

El **objetivo general** de este trabajo consiste en crear un software que permita la gestión de la planificación del entrenamiento deportivo para el judo femenino.

Teniendo en cuenta los aspectos mencionados anteriormente la **idea a defender** es: si se desarrolla un software que cumpla con los requisitos funcionales y no funcionales definidos para el mismo, que automatice todo el proceso de planificación del entrenamiento deportivo, entonces se facilitará el trabajo de los entrenadores de la Escuela Cubana de Judo Femenino.

Los **objetivos específicos** a cumplir son:

- Diseñar teóricamente la investigación.
- Revisar y estudiar la existencia de otras aplicaciones o soluciones similares aplicadas a diferentes deportes.
- Seleccionar el lenguaje y metodología de desarrollo de software a utilizar.
- Seleccionar las herramientas que se van a utilizar según la metodología.
- Capturar los requisitos del sistema.
- Elaborar el análisis y diseño del sistema según los requerimientos.
- Implementar el software en dependencia de las funcionalidades que debe tener.

Las **tareas** a realizar para dar solución a los objetivos son:

- Realizar la revisión bibliográfica para conformar el estado del arte de la investigación.
- Realizar un estudio sobre los sistemas informáticos existentes que apoyan la planificación del entrenamiento deportivo.
- Realizar un estudio sobre los diferentes lenguajes y metodologías de desarrollo de software que existen para conocer las ventajas que brindan.
- Realizar el estudio de todas las herramientas posibles a utilizar según la metodología de desarrollo de software seleccionada.

- Realizar entrevistas con los clientes que desean automatizar el sistema para conocer como funciona el proceso de planificación del entrenamiento y para definir los requerimientos del sistema.
- Realizar la elaboración del modelo del negocio.
- Realizar los diagrama de clases del análisis y del diseño.
- Realizar la implementación del sistema propuesto.

Los **métodos científicos** utilizados en la investigación fueron:

Teóricos: constituyen el enfoque general para abordar los problemas científicos, de ahí que posibilitan profundizar en las regularidades y cualidades esenciales de los fenómenos.

Empíricos: permiten la obtención y elaboración de los hechos fundamentales que caracterizan a los fenómenos, a la par que facilitan confirmar hipótesis y teorías (Navarro 2005).

Dentro de los métodos teóricos:

Analítico – sintético

Son dos procesos inherentes al pensamiento, operaciones lógicas importantes; que permiten buscar la esencia de los fenómenos, los rasgos que los caracterizan y los distinguen. Su objetivo en una investigación es analizar las teorías, documentos, etc.; permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Análisis histórico-lógico

Para investigar sobre las aplicaciones informáticas de este tipo implementadas en Cuba y el resto del mundo.

Dentro de los empíricos se emplearon:

Entrevistas

Para conocer cuales son las necesidades de los profesionales del INDER y de la Federación Cubana de Judo.

Para mayor facilidad y organización a la hora de realizar el estudio del documento, este trabajo se encuentra dividido en cuatro capítulos.

- Capítulo 1.- Fundamentación teórica: incluye el estado del arte del tema tratado, además de la selección de la metodología, plataforma, lenguaje y herramientas, para desarrollar el sistema.
- Capítulo 2.- Características del sistema: se modela el negocio y se hace el levantamiento de requisitos, obteniendo como artefacto fundamental el diagrama de casos de uso del sistema.
- Capítulo 3.- Análisis y diseño del sistema: se comienza la construcción de la propuesta, determinando y describiendo las clases del diseño.
- Capítulo 4.- Implementación: se hace referencia al modelo de implementación mediante el diagrama de despliegue y de componentes.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En este capítulo se expone la base teórica sobre la que se fundamenta el sistema para la planificación del entrenamiento del judo femenino. También se realizó un estudio de los sistemas similares que existen tanto a nivel nacional como internacional, no solo dentro del deporte de judo, sino también dentro de otras disciplinas deportivas. Además de un estudio de las metodologías, estándares para el desarrollo de software, tecnologías actuales, lenguajes de desarrollo y herramientas, así como la selección de las adecuadas para usarlas en el desarrollo del sistema que dará solución a la propuesta.

1.2 Fundamentación del tema

1.2.1 Sistemas de planificación del entrenamiento deportivo

Según el Diccionario de Ciencias de la Educación planificar es: el proceso que debemos seguir para alcanzar objetivos concretos en unos plazos terminados y en etapas definidas, partiendo del conocimiento y de la evaluación científica de la situación de origen y utilizando de modo racional los medios naturales y los recursos humanos disponibles. La planificación del entrenamiento deportivo es un proceso sistemático, que se realiza de forma organizada, registrando todo el contenido del entrenamiento, con el objetivo de que los atletas alcancen la forma deportivo antes de presentarse a una competencia fundamental. La Escuela Cubana de Judo Femenino debido a que no tiene ningún sistema que gestione esta planificación, la realiza de forma manual. Este proceso es bastante complejo debido a que para su elaboración hay que realizar muchos cálculos, y al realizarlos manualmente provoca que se pierdan datos importantes, además de que pueden ocurrir errores de cálculos, y al ser un proceso largo consume mucho tiempo de los entrenadores, que lo podrían emplear para preparar a los atletas.

Con este trabajo se pretende automatizar el proceso de planificación del entrenamiento para el judo femenino. El objetivo es crear una aplicación capaz de gestionar todo lo relacionado con la planificación del entrenamiento, para que de esta manera los entrenadores no tengan que realizar este proceso manualmente.

1.3 Sistemas similares

En la actualidad existen múltiples sistemas que facilitan la planificación del entrenamiento deportivo. Estos varían en dependencia de la disciplina deportiva, ya que no todos los deportes entrenan de la misma forma ni con la misma intensidad. Actualmente existen sistemas que han colaborado con la planificación del entrenamiento del judo y otros deportes en diversos países.

1.3.1 Sistemas similares a nivel internacional

En el mundo existen diversos sistemas que se encargan de la planificación del entrenamiento deportivo para diferentes deportes, un ejemplo de ello es:

***X-MEDALIST**:* Es un software que se encarga de planificar y controlar el entrenamiento para deportes individuales. Además de incorporar herramientas nuevas aplicadas al entrenamiento, permite trabajar de forma rápida y coordinada, organizando toda la actividad laboral. Este programa está orientado a entrenadores de cualquier deporte individual. Brinda al entrenador la posibilidad de ingresar a todos sus atletas. Además permite llevar un registro y control de lesiones, historia clínica y deportiva, y del análisis de la técnica individual y el biorritmo personal. También permite revisar toda la información que se encuentra en las planificaciones en forma de lista simple o bien combinada entre actividades o deportistas. Dispone de una herramienta que le permite comparar volúmenes de diferentes áreas entre dos o más deportistas y brinda la posibilidad de exportar todos los resultados a formato *Excel* (R) (Palomeque 2009).

X-Training FUSSION: Es una versión de un programa de computación para la planificación, periodización y control del entrenamiento deportivo. Este programa está pensado y diseñado para cubrir todas las expectativas de los entrenadores personales, preparadores físicos, técnicos de diferentes deportes, clubes, instituciones deportivas, laboratorios de evaluaciones deportivas, etc. Además brinda la posibilidad de confeccionar planificaciones para un deportista individual, o bien para un equipo de cualquier deporte. Puede almacenar simultáneamente infinita cantidad de planificaciones, para uno o más equipos o individuos, y para todas las categorías que desee. Tiene un generador de evaluaciones que brinda la posibilidad a los entrenadores de definir cualquier tipo de evaluación, y controlar hasta 10 variables diferentes de forma simultánea. Además, puede calcular resultados adicionales utilizando fórmulas que combinen todos los resultados obtenidos.

Para ahorrarle tiempo y esfuerzo, tiene implementado un sistema que permite exportar las planillas de evaluaciones a *Excel* para poder cargar los resultados de las evaluaciones directamente en el campo de entrenamiento, incorporando luego de forma automática el traspaso de información hacia su base de datos para que pueda analizar los resultados y obtener todo tipo de comparaciones y datos estadísticos (Palomeque 2009).

Espacio virtual para profesores y entrenadores de judo: Es un sitio que puede ser usado como guía por los entrenadores para la realización de diversas actividades entre las que se pueden citar: impartir cursos de judo, explicar las didácticas del judo, así como dar a conocer los parámetros a tener en cuenta para que un judoca alcance la forma óptima. Es simplemente una Web de información al entrenador (Sariola 2008).

Elatleta.com: La Web oficial del club deportivo *Páris*, cuyo objetivo es informar de todo lo relacionado con el deporte de atletismo. Este plan de entrenamiento pretende fortalecer el organismo de las personas que practican el atletismo, para ello se centra en mejorar la capacidad aeróbica y la musculatura; especialmente las piernas y todos los músculos que protegen la columna. Es una aplicación Web de estructura dinámica que brinda mucha información a los entrenadores de atletismo (Seco 2000).

1.3.2 Sistemas similares a nivel nacional

Automatización de la planificación del entrenamiento deportivo en diferentes deportes.

Es una planificación realizada por la Facultad de Cultura Física de Villa Clara para el entrenamiento deportivo de diferentes deportes. Como resultado de la investigación se obtuvo el diseño y programación de sistemas automatizados para la planificación del atletismo, las pesas como deporte, las pesas como deporte auxiliar y la esgrima. Para la elaboración de cada una de las planificaciones se utilizó *Microsoft Access* (Ortega 2005).

Planificación del entrenamiento para el deporte de atletismo: Para este software se tuvieron en cuenta los elementos teóricos de la planificación en este deporte, los cuales permitieron definir los macrociclos, períodos y etapas que se encuentran dentro del plan de entrenamiento requerido por los especialistas.

Planificación del entrenamiento para el deporte pesas: Para el software de pesas se tuvieron en cuenta los elementos teóricos y los pasos para realizar la misma de forma manual. El software tiene la

ventaja de haber sido confeccionado para el equipamiento más moderno disponible en el país y facilita las entradas de datos utilizando las posibilidades de los cuadros combinados para la selección de textos, en lugar de tener que teclearlos, y da la posibilidad a los usuarios de seleccionar las variantes de semanas y días más usados y de incluir otras variantes o modificar las ya existentes.

Planificación del entrenamiento en las pesas como deporte auxiliar: En este software se tuvo en cuenta que los entrenadores de todas las disciplinas utilizan las pesas como deporte auxiliar, pero ello requiere de su planificación, la cual si se hace con todo el rigor necesario, trae aparejada la realización de una gran cantidad de cálculos engorrosos, que le restan tiempo de su labor fundamental: la atención de las diferencias individuales de los atletas. El software permite obtener el plan de entrenamiento de todos los microciclos del mesociclo planificado (utilizando las pesas) por plano muscular y zona de intensidad en cada día de entrenamiento.

Planificación del entrenamiento deportivo en esgrima: Para el software de esgrima se tuvo en cuenta el criterio de los entrenadores, los cuales solicitaron el modelo de salida requerido. Basándose en este y en las transformaciones necesarias de los datos se programó el sistema. El software facilita las entradas de datos utilizando las posibilidades de los cuadros combinados para la selección de textos en lugar de tener que teclearlos.

Todos estos sistemas son de gran ayuda para la realización del software deseado, ya que tienen en común automatizar el proceso de planificación, ya sea para atletismo, pesas, esgrima, o cualquier otro deporte. Con el fin de que la persona que trabaje directamente con la aplicación se ahorre el tedioso trabajo de realizar grandes cantidades de cálculos, y así agilizar el proceso de forma organizada y sistemática. Actualmente la Escuela Cubana de Judo Femenino no cuenta con un software que facilite la planificación del entrenamiento. Por lo que con el apoyo de los sistemas estudiados anteriormente y la experiencia de los entrenadores en la realización de estas planificaciones, se va a desarrollar un software que permita registrar diariamente los ejercicios que van a realizar las atletas, además de la intensidad con que van a entrenar, también se tendrán registrados en que microsistemas se le aplicarán los test pedagógicos, campos de entrenamiento, las pruebas médicas y las competencias en las que participarán, entre otras funcionalidades.

1.4 Estudio de las metodologías y estándares para el desarrollo de software

Rational Unified Process (RUP)

Es una metodología de desarrollo pesada para la ingeniería de software que va más allá del análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. Además de que unifica completamente a un equipo de desarrollo y optimiza la productividad de cada uno de los miembros del mismo, ayudando a los líderes de proyecto a incrementar su experiencia en el desarrollo de proyectos.

Los verdaderos aspectos definitorios del proceso unificado se resumen en tres frases claves, dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental. Esto es lo que hace único al proceso unificado (I. Jacobson 2000a).

Iterativo e incremental: donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo.

Dirigido por casos de uso: que orienta el proyecto a la importancia para el usuario y lo que este necesita y desea.

Centrado en la arquitectura: relaciona la toma de decisiones que indica como tiene que ser construido el sistema y en qué orden (Gallego 2007).

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

- Trabajadores (quién): son las personas involucradas en cada proceso.
- Artefactos (qué): puede ser un documento, un modelo o un elemento de un modelo.
- Actividades (cómo): son los procesos que se llegan a determinar en cada iteración.
- Flujo de actividades (cuándo): son los flujos donde se va a realizar cada actividad.

Un proyecto que se desarrolle utilizando la metodología RUP está dividido en 4 fases ,6 flujos de ingeniería y 3 de apoyo (Mellon 2009).

Fases:

- Inicio: Alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo de vida para el proyecto, generando el ámbito del proyecto, el caso de negocio, síntesis de arquitectura posible y el alcance del proyecto.
- Elaboración: Establecimiento de la línea base para la arquitectura del sistema y proporcionar una base estable para el diseño y el esfuerzo de implementación de la siguiente fase, mitigando la mayoría de los riesgos tecnológicos.
- Construcción: Completar el desarrollo del sistema basado en la línea base de la arquitectura.
- Transición: Garantizar que el software esté listo para entregarlo a los usuarios.

Flujos de ingeniería:

- Modelo de negocio
- Requerimiento
- Análisis y diseño
- Implementación
- Prueba
- Despliegue

Flujos de apoyo:

- Gestión de configuración y cambios
- Gestión de proyecto
- Ambiente o entorno

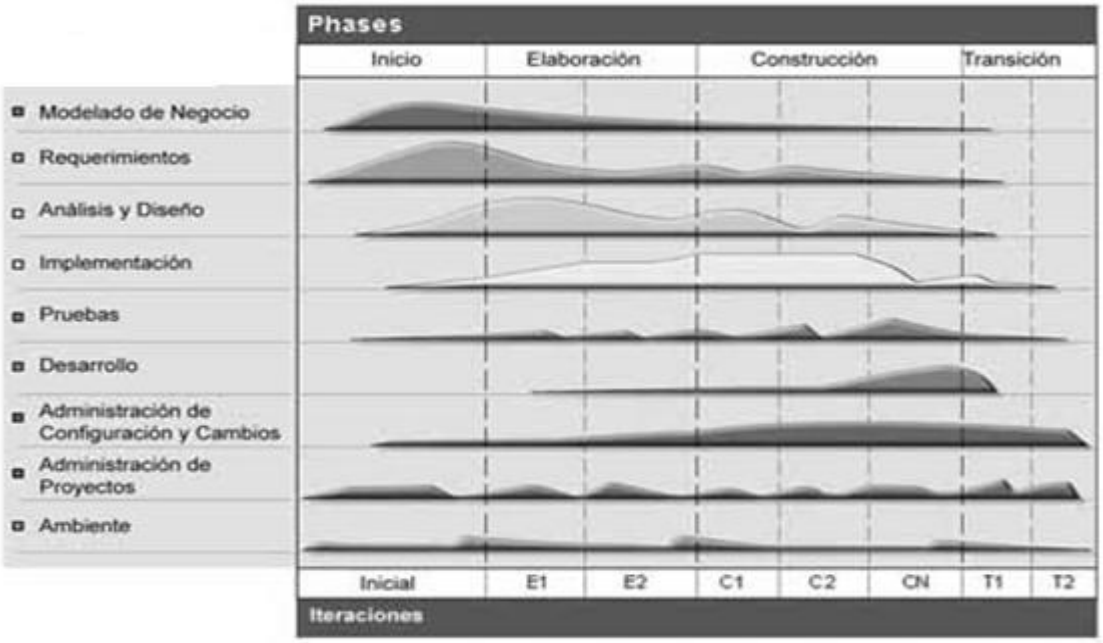


Figura 1: Fases y flujos de trabajo de RUP.

Extreme Programming (XP)

La programación extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

XP se caracteriza por ser una metodología ágil, la cual ofrece al cliente el software que necesita y cuándo lo necesita. Combina las mejores prácticas para desarrollar software, y las lleva al extremo.

Es el más destacado de los procesos ágiles de desarrollo de software, se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

XP se basa en la simplicidad, la comunicación y el reciclado continuo de código, es decir no es más que aplicar una pura lógica. Tiene como meta la disminución de costes y su objetivo principal es satisfacer a los clientes y potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los desarrolladores y el cliente son parte del equipo de desarrollo y se encuentran involucrados en el desarrollo del software.

Los desarrolladores deben saber que en el ciclo de vida de desarrollo de un proyecto de software los cambios van a aparecer, cambiarán los requisitos, las reglas de negocio, el personal, la tecnología, todo

va a cambiar. Por tanto el problema no es el cambio, ya que este va a suceder, sino la incapacidad de adaptarnos a estos cambios.

En un proyecto desarrollado con la metodología XP se definen 4 variables (Escribano 2002):

- Coste
- Tiempo
- Calidad
- Ámbito

Además esta metodología trata de minimizar la complejidad de un proyecto tratando de enfocarse directamente hacia el objetivo, haciendo uso de las relaciones interpersonales y la rapidez de reacción. Se caracteriza por llevar a cabo **pruebas unitarias**, basadas en pruebas hechas a los procesos de mayor importancia, de esta forma es como si pudiese obtener los posibles errores. Permite además **refabricación** que no es más que la reutilización de código, siendo el cambio más flexible. Una de las características interesantes de esta metodología es que permite la **programación en pares**, la cual consiste en que dos programadores trabajen una misma estación de trabajo, haciendo más eficiente el trabajo, ya que un programador complementa la tarea del otro y viceversa.

El ciclo de vida ideal de XP consta de seis fases:

- Exploración
- Planificación de la entrega
- Iteraciones
- Producción
- Mantenimiento
- Muerte del proyecto

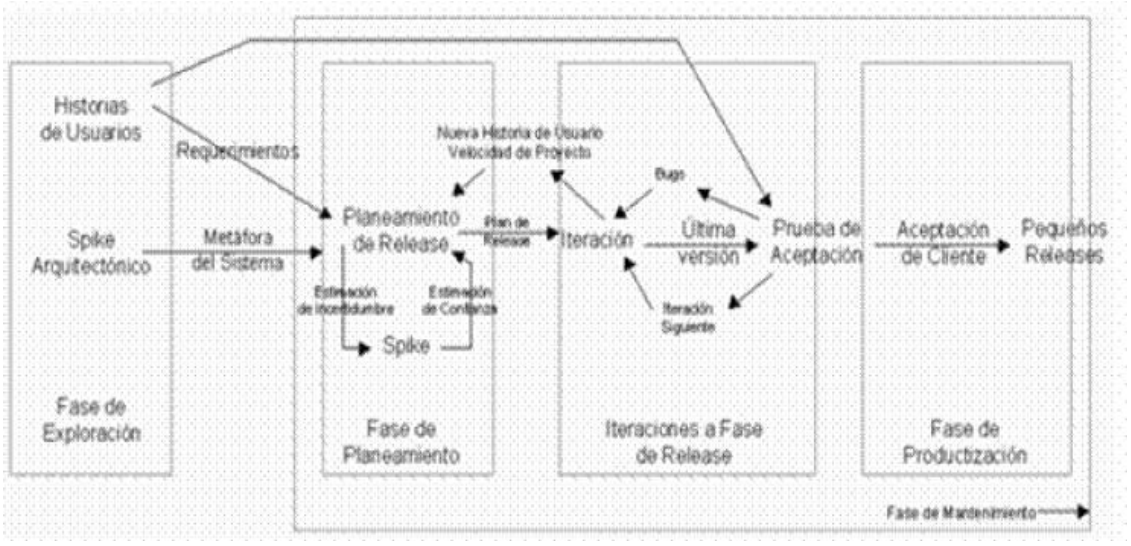


Figura 2: Ciclo de desarrollo de XP.

Selección de la metodología para dar solución al problema

XP es una metodología que tiene como característica que tanto los jefes de proyecto, los desarrolladores y el cliente, son parte del equipo de trabajo y se encuentran involucrados en el desarrollo del software. En este caso el cliente no es parte del equipo de desarrollo, por lo que esta metodología no contribuye lo suficiente como para desarrollar un software con calidad. Sin embargo RUP mantiene al equipo enfocado en producir incrementalmente software operativo a tiempo, con las características y calidad requerida. Además las mejores prácticas probadas en la industria, contenidas en RUP, incorporan las lecciones aprendidas de cientos de líderes de la industria y miles de proyectos.

Por todas las características vistas anteriormente se ha decido usar RUP como metodología, ya que basa su trabajo principalmente en la documentación del software y expone un conjunto de actividades que están orientadas a visualizar, especificar, construir y documentar los artefactos necesarios para el desarrollo de software con calidad, presentando una exhaustiva definición de artefactos para ello. También hace uso del Lenguaje Unificado de Modelado (UML), cuya utilización de diagramas y gráficos brindan una mejor perspectiva de lo que se quiere.

Lenguaje Unificado de Modelado (UML)

UML es un lenguaje de modelado visual que se usa para visualizar, especificar, construir y documentar artefactos de un sistema de software (I. Jacobson 2000b).

Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software (Lafuente 2000). Este está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que UML es un lenguaje de modelado, cuenta con reglas para combinar tales elementos. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

El Lenguaje Unificado de Modelado proporciona a los desarrolladores un vocabulario que incluye tres categorías: elementos, relaciones y diagramas (I. Jacobson 2000a).

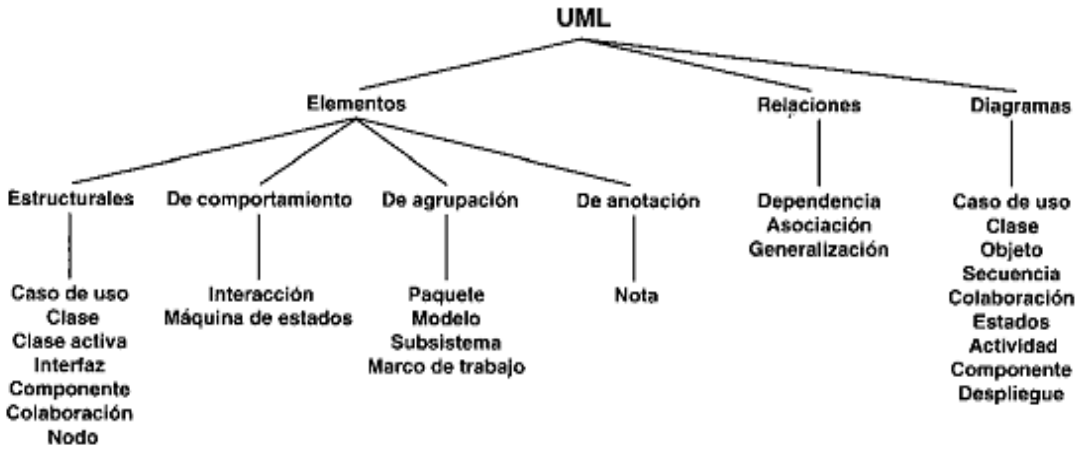


Figura 3: El vocabulario de UML.

Es importante destacar que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso, sino un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Además describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

1.5 Tendencias y tecnologías actuales. Selección de las herramientas y lenguajes de desarrollo.

Se propone utilizar tecnología Web para resolver el problema planteado anteriormente, debido a que esta tecnología posee muchas ventajas como:

- Se necesita solamente un navegador Web y conexión al sistema.
- Se pueden ejecutar varios clientes simultáneamente y en diferentes estaciones de trabajo.
- Se necesitan pocos recursos para que la aplicación trabaje correctamente.
- Toda la información se encontrará de forma centralizada en el servidor, al igual que la administración.
- No hay generalmente discriminación del sistema operativo por parte del usuario, aunque exista por parte del servidor.

1.5.1 Lenguajes de programación

El sistema que se implementará será una aplicación Web, por tanto usará arquitectura cliente-servidor. En este caso el cliente solo estará relacionado con la interfaz externa del servidor, no con su lógica interna. Además el cliente puede trabajar sin importar donde se encuentre el servidor, ni que sistema operativo tenga, solamente necesita conexión hacia él.

El servidor puede estar sujeto a cambios que deben influir poco o en nada en el cliente, también puede estar en diferente plataforma a la del servidor que no afectará el funcionamiento del sistema. Algo importante es que el servidor puede brindar servicios múltiples y simultáneos a clientes en cualquier lugar.

Para llevar a cabo la implementación del sistema es necesario saber cual lenguaje de programación usar.

Un lenguaje de programación es un conjunto de símbolos y palabras que permiten al usuario de una computadora darle instrucciones y órdenes para que la computadora las ejecute (Miguel Ángel Manzanedo del Campo 1999). Es utilizado para controlar el comportamiento físico y lógico de un ordenador.

Para desarrollar un sistema Web los lenguajes de programación se pueden dividir en:

- Lenguaje del lado del cliente.
- Lenguaje del lado del servidor.

Dentro de este último caso (lenguaje del lado del servidor), se pueden apreciar lenguajes como PHP, ASP, JSP, *Java* y otros más. Dentro de los lenguajes del lado del cliente se encuentran HTML, CSS, *JavaScript* entre otros.

***Hypertext Pre-Processor* (PHP)**

Es un lenguaje *script* interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas (Hinostroza 2005). Principalmente es usado en la interpretación del lado del servidor (*server-side scripting*), pero puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

PHP es un lenguaje sencillo, de sintaxis cómoda y similar a la de otros lenguajes como C o C++, es rápido a pesar de ser interpretado, multiplataforma y dispone de una gran cantidad de librerías que facilitan muchísimo el desarrollo de las aplicaciones (Castillo 2007).

Además tiene muchísimas más características, como son:

- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados extensiones).
- Posee una amplia documentación entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones (desde PHP5).

Active Server Pages (ASP)

Es una tecnología del lado del servidor desarrollada por *Microsoft* para la realización de sitios Web dinámicos. Las páginas Web desarrolladas bajo este lenguaje necesitan tener instalado *Internet Information Server* (IIS).

ASP no necesita ser compilado para ejecutarse. Existen varios lenguajes que se pueden utilizar para crear páginas ASP. El más utilizado es *VBScript*, nativo de *Microsoft*. El código ASP puede ser insertado junto con el código HTML. Los archivos cuentan con la extensión (asp).

Características:

- Usa *Visual Basic Script*, siendo fácil para los usuarios.
- Comunicación óptima con SQL *Server*.
- Soporta el lenguaje *Jscript* (*JavaScript* de *Microsoft*).
- Se necesita escribir mucho código para realizar funciones sencillas.
- Tecnología propietaria.
- Hospedaje de sitios Web costosos.

Java Server Pages (JSP)

Es un lenguaje multiplataforma, para la creación de sitios Web dinámicos, acrónimo de *Java Server Pages*. Está orientado a desarrollar páginas Web en *Java* y creado para ejecutarse del lado del servidor.

Con JSP se pueden crear aplicaciones Web que se ejecuten en variados servidores, de múltiples plataformas, ya que *Java* es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar *scripts* de servidor en sintaxis *Java* (Alvarez 2002).

JSP fue desarrollado por *Sun Microsystems*. Comparte ventajas similares a las de ASP.NET, desarrollado para la creación de aplicaciones Web potentes. Posee un motor de páginas basado en los *servlets* de *Java*. Para su funcionamiento se necesita tener instalado un servidor *Tomcat*.

Características:

- Código separado de la lógica del programa.
- Las páginas son compiladas en la primera petición.
- Permite separar la parte dinámica de la estática en las páginas Web.
- Los archivos se encuentran con la extensión (jsp).
- El código JSP puede ser incrustado en código HTML.
- Complejidad de aprendizaje.

Java

Es un lenguaje de programación orientado a objetos. Tiene un modelo de objetos simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Características:

Orientado a Objetos: método de programación y diseño del lenguaje. Aunque hay muchas interpretaciones, una primera idea es diseñar el software de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones.

Independencia de la plataforma: significa que programas escritos en el lenguaje *Java* pueden ejecutarse igualmente en cualquier tipo de *hardware*. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo.

Posee también recolector automático de basura lo cual independiza al programador de tener que administrar memoria solicitada dinámicamente de forma manual.

Multihebra: Hoy en día se ven terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. *Java* soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario, mientras otro presenta una animación en pantalla y otro realiza cálculos (Marañón 1999).

HTML

HTML, *Hyper Text Markup Language* (Lenguaje de Marcación de Hipertexto) es el lenguaje de marcas de texto utilizado normalmente en la WWW (*World Wide* Web). Fue creado en 1986 por el físico nuclear Tim Berners-Lee; el cual tomo dos herramientas preexistentes: El concepto de hipertexto (conocido también como *link* o ancla) el cual permite conectar dos elementos entre si y el SGML (Lenguaje Estándar de Marcación General) el cual se utiliza para colocar etiquetas o marcas en un texto que indique como debe verse.

HTML no es propiamente un lenguaje de programación como C++, *Visual Basic*, etc., sino un sistema de etiquetas. No presenta ningún compilador, por lo tanto algún error de sintaxis que se presente éste no lo detectará y se visualizará en la forma como éste lo entienda. Estos documentos pueden ser mostrados por los visores o "*browsers*" de páginas Web en internet, como *Netscape Navigator, Mosaic*, *Opera* y *Microsoft Internet Explorer*. También existe el HTML Dinámico (DHTML), que es una mejora de *Microsoft* de la versión 4.0 de HTML que le permite crear efectos especiales como, por ejemplo, texto que vuela desde la página palabra por palabra o efectos de transición al estilo de anuncio publicitario giratorio entre página y página.

CSS

(*Cascading Style Sheets u* Hojas de Estilo en Cascada) es la tecnología desarrollada por el *World Wide* Web *Consortium (W3C)* con el fin de separar la estructura de la presentación. A pesar de que la recomendación oficial del grupo de trabajo de la W3C ya había alcanzado la estabilidad requerida para que fuera soportada por los principales navegadores comerciales, como *Netscape e Internet Explorer*.

La potencia de la tecnología CSS permite aplicar al documento formato de modo mucho más exacto. Si antes el HTML se quedaba corto para maquetar las páginas y había que utilizar trucos para conseguir los efectos deseados, ahora existen muchas más herramientas que permiten definir esta forma:

- Se puede definir la distancia entre líneas del documento.
- Colocar elementos en la página con mayor precisión, y sin dar lugar a errores.
- Definir la visibilidad de los elementos, márgenes, subrayados, tachados, etc.

Esta tecnología es bastante nueva, por lo que no todos los navegadores la soportan. En concreto, sólo los navegadores de *Netscape* versiones de la 4 en adelante y de *Microsoft* a partir de la versión 3 son capaces de comprender los estilos en sintaxis CSS (Alvarez 2001).

JavaScript

JavaScript es muy fácil de aprender para quien ya conoce lenguajes similares como el C++ o *Java*, pero, dada su simplicidad sintáctica y su manejabilidad, no es tampoco difícil para quien se acerca por primera vez a este lenguaje. Sin embargo, esto puede ser un arma de doble filo porque la simplicidad se basa en una disponibilidad de objetos limitada, por lo que algunos procedimientos, aparentemente muy sencillos, requieren *script* bastante complejos (Valdelli 2006).

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, ya que el lenguaje funciona del lado del cliente y los navegadores son los encargados de interpretar estos códigos, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. Además *JavaScript* tiene la ventaja de ser incorporado en cualquier página Web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. Es un lenguaje interpretado basado en objetos. No es un lenguaje de sintaxis estricta en lo que a tipos de datos se refiere: las variables no necesariamente deben ser declaradas. La verificación de objetos se lleva acabo en *run-time* (conocido como *dynamic binding*). Es soportado por la mayoría de los navegadores como *Internet Explorer*, *Netscape, Opera, Mozilla Firefox*, entre otros.

Tecnología *Ajax*

Con el surgimiento de lenguajes como PHP del lado del servidor y *JavaScript* del lado del cliente, surgió *Ajax* en acrónimo de (*Asynchronous JavaScript And XML*). El mismo es una técnica para crear aplicaciones Web interactivas. Las aplicaciones Web proliferan debido a su simplicidad, pero ofrecen una menor interactividad y usabilidad en comparación con las aplicaciones de escritorio, debido a que la interacción del usuario con una aplicación Web se interrumpe cada vez que se necesita algo del servidor. Varias tecnologías han sido diseñadas para resolver este problema, *Ajax* es una nueva solución que no requiere *plugins* o capacidades específicas de ciertos navegadores. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el

servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. En *JavaScript* se efectúan las funciones de llamada de *Ajax* mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. Además *Ajax* es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como *JavaScript* y *Document Object Model* (DOM).

Selección del lenguaje de desarrollo

Después de haber realizado un estudio se llega a la conclusión de que PHP es el lenguaje adecuado para desarrollar la aplicación del lado del servidor, ya que es un poderoso lenguaje e intérprete, ya sea incluido como parte de un servidor Web en forma de módulo o ejecutado como un binario, es capaz de acceder a archivos, ejecutar comandos y abrir conexiones de red en el servidor. Además la aplicación que se va a desarrollar no es grande y PHP es un lenguaje adecuado para pequeñas aplicaciones Web dinámicas. También es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo Web y puede ser incluido dentro del código HTML. PHP es independiente de plataforma, puesto que existe un módulo para casi cualquier servidor Web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante, ya que permite portar el sitio desarrollado en PHP de un sistema a otro sin prácticamente ningún trabajo y en el caso de estar montado sobre un servidor *Linux* o *Unix*, es más rápido que ASP, dado que se ejecuta en un único espacio de memoria.

Del lado del cliente HTML y CSS son los indicados pues se podrá manejar con más facilidad todo lo referente al diseño, ya que HTML indica al navegador donde colocar cada texto, imagen o video y la forma que tendrán estos al ser colocados en la página. En cuanto a CSS se podrá incluir márgenes, tipos de letra, fondos, colores, incluso definir los estilos en un archivo externo a las páginas; así, si en algún momento se quiere cambiar alguno de ellos, automáticamente se actualizarán todas las páginas vinculadas del sitio.

El uso de *JavaScript* permite hacer la Web más interactiva y posibilita también que muchas validaciones se realicen del lado del cliente, haciendo que el servidor se sobrecargue menos. Además *JavaScript*

permite utilizar la tecnología *Ajax*, tecnología que posibilita la interacción con el servidor en segundo plano, es decir que no modifica la interfaz de la Web cuando esta hace o recibe peticiones por parte del servidor, por lo que el proceso de obtener la información a través de una aplicación de servidor es inevitable, sin embargo, el mecanismo de regenerar el documento que visualiza el usuario en cada petición, puede ser evitado empleando *Ajax*.

Java es un lenguaje para aplicaciones con alta complejidad y seguridad, además una de sus desventajas es que el tiempo en que se accede a las páginas, es lento, en este caso, la solución implica mejorar el equipo con el que se cuenta, (*modem*, RAM, *cache*). ASP es una tecnología propietaria por lo que se decide descartarla, además de que se necesita mucho código para hacer funciones sencillas. En cuanto a JSP es un lenguaje para hacer Web potentes, su entendimiento es complejo y para su funcionamiento se necesita de otras herramientas (*Tomcat*).

1.4.2 Entorno de desarrollo integrado

Un IDE es un entorno de programación que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Estas pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Zend Studio For Eclipse

Zend Studio para *Eclipse* es una tecnología probada que combina el *Zend* PHP y el *Eclipse Tools PDT*, en la creación de un IDE para el desarrollo de aplicaciones Web más potente del mercado, ofreciendo al desarrollador profesional de PHP la potencia de *Zend Studio* y el soporte multilenguaje de *Eclipse* y su enorme conjunto de extensiones (*plugins*) (Carlos David Marrero 2009). El mismo presenta el inconveniente de no ser gratis.

Esta nueva versión combina un IDE versátil y potente con las capacidades de expansión del ecosistema del proyecto *Eclipse*. Entre sus funcionalidades, se destaca la capacidad de refactorización del código fuente, funcionalidad que permite adecuar el comportamiento externo de una función o clase sin cambiar el funcionamiento interno, que junto a los nuevos *wizards* y capacidades de generación de código facilitarán el trabajo de los desarrolladores. Desde el punto de vista de un IDE completo, disponer de un buen *debugger* local con la conexión a los servidores de desarrollo, junto a una política de trabajo en

equipo y un sistema de control de versiones es posible manejar sin problemas proyectos complejos utilizando PHP.

Eclipse

Es un IDE de programación totalmente gratuito, que lleva tiempo demostrando su hegemonía en la programación *Java*, después de posicionarse como un software de desarrollo altamente competitivo entre los paquetes de desarrollo más profesionales está demostrando su calidad en el desarrollo de PHP con el *plugin* de PHP. Este *plugin* además permite que el trabajo con PHP sea muy rápido pues tiene auto completamiento y colorea errores.

Es únicamente un armazón sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los *plugins* adecuados. La arquitectura de *plugins* de *Eclipse* permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otros.

Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de *Java* llamado *Java Development Toolkit* (JDT) y el compilador que se entrega como parte de *Eclipse* y que son usados también para desarrollar el mismo *Eclipse*.

Selección del IDE de desarrollo

El *Eclipse* es el IDE adecuado para desarrollar la propuesta ya que tiene características claves como que es libre. Además de ser un buen entorno de programación que puede competir con cualquier paquete de software comercial. El *Zend Studio For Eclipse* es un entorno muy bueno para el desarrollo de aplicaciones Web, pero requiere licencia de pago, no incluye editor visual HTML, y es un poco complejo.

1.4.3 Herramientas CASE de Modelado con UML

Las herramientas CASE (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*: Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y costo.

Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte

del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm

Es una herramienta CASE que utiliza “UML”: como lenguaje de modelado profesional, la cual soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Permite realizar ingeniería tanto directa como inversa. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o Pdf, y permite control de versiones. Cabe destacar igualmente su robustez, usabilidad y portabilidad.

Rational Rose

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1.

Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Desarrollo Iterativo: utiliza un proceso de desarrollo iterativo controlado donde el desarrollo se lleva a cabo en una secuencia de iteraciones.

Trabajo en Grupo: permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

Ingeniería Inversa: a partir del código de un programa, se puede obtener información sobre su diseño.

Selección de la herramienta CASE para UML

Por las características vistas anteriormente se decide seleccionar *Visual Paradigm*, ya que se integra mejor por su rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

1.5 Conclusiones

RUP es la metodología a utilizar, ya que basa su trabajo principalmente en la documentación del software y expone un conjunto de actividades que están orientadas a visualizar, especificar, construir y documentar los artefactos necesarios para el desarrollo de un software de calidad, presentando una exhaustiva definición de artefactos para ello. También hace uso del lenguaje unificado de modelado (UML), cuya utilización de diagramas y gráficos brinda una mejor perspectiva de lo que se quiere. *Visual Paradigm* es la herramienta de modelado a usar, ya que esta se integra mejor que las otras herramientas estudiadas por su rápida construcción de aplicaciones de calidad, mejores y a un menor costo, además de que permite hacer todo tipos de diagramas y documentación. PHP es el lenguaje a utilizar pues es un poderoso lenguaje e intérprete para desarrollar aplicaciones Web del lado del servidor, además las características de este planteadas anteriormente lo hacen el indicado, y por parte del cliente HTML y CSS abarcarán todo lo referente al diseño de las páginas. El uso de *JavaScript* permite hacer la Web más interactiva y posibilita también que muchas validaciones se realicen del lado del cliente, haciendo que el servidor se sobrecargue menos. Además *JavaScript* permite utilizar la tecnología *Ajax*, tecnología que permite la interacción con el servidor en segundo plano, es decir que no modifica la interfaz de la Web cuando esta hace o recibe peticiones por parte del servidor. El *Eclipse* es el IDE que se va a utilizar para desarrollar la propuesta al problema, ya que tiene como ventaja sobre los demás que es libre y tiene un *plugin* para PHP que facilita el trabajo a la hora de programar en este lenguaje.

Capítulo2 Características del sistema

2.1 Introducción

En este capítulo se realiza la descripción de la solución propuesta en el trabajo. Además de los procesos del negocio que tienen que ver con el campo de acción. También se enumeran los requisitos funcionales y no funcionales esperados para el desarrollo del sistema propuesto, así como la identificación de los actores, casos de uso y las relaciones existentes entres ellos a través del modelo de casos de uso del sistema.

2.2 Descripción de los procesos del negocio

El plan de entrenamiento consiste en una planificación de todas las actividades que se van a realizar para preparar al equipo de judo femenino con vista a una competencia. Este está compuesto por planes gráficos , en los cuales los entrenadores identifican la competencia fundamental de un período determinado y dividen el tiempo que hay hasta esa competencia en dos macrosistemas, aunque en casos especiales se puede dividir en tres macrosistemas de cinco mesosistemas cada uno de ellos (mesosistema de preparación física general (MPFG), mesosistema de preparación física especial variada (MPFEV), mesosistema de preparación física especial (MPFE), mesosistema de obtención de la forma deportiva (MOFD) y mesosistema de estabilización de la forma deportiva (MEFD)). Para cada uno de estos mesosistemas definen el volumen, la intensidad, la cantidad de sesiones, horas y minutos que se va a trabajar, los combates que se van a realizar y el rango de tolerancia de peso, además a cada mesosistema le asignan la cantidad de microsistemas deseadas, y planifican en cuales microsistemas van a realizar los test pedagógicos, pruebas médicas, competencias y campos de entrenamiento, con el objetivo de preparar a los atletas para lograr obtener la forma deportiva antes de llegar a la competencia deseada. Otro de los aspectos que se tiene en cuenta dentro del plan de entrenamiento son los test pedagógicos, a través de los cuales se miden la condición física de cada una de las atletas, registrando los resultados obtenidos en las diferentes pruebas realizadas, como son: nivel de resistencia, velocidad, flexibilidad, fuerza, prueba teórica y prueba técnica, para de esta manera preparar a los atletas y lograr que mejoren su condición física.

2.3 Descripción del sistema propuesto

Para dar solución a los objetivos anteriormente planteados, el sistema que se implementará constará de 4 módulos (administración, plan gráfico, test pedagógico y atleta) y 2 roles (administrador y entrenador).

Todos los usuarios que accedan al sistema estarán registrados en la base de datos, permitiendo mayor seguridad al sistema, ya que un usuario no registrado no podrá acceder al sistema. El usuario con rol de administrador será el encargado del registro de los usuarios que accederán al sistema, asignándoles su respectivo rol. El usuario entrenador gestionará todo lo referente al módulo plan gráfico, test pedagógico y atleta.

En el módulo administración es donde se gestionarán los usuarios que accederán al sistema, permitiendo insertar, eliminar y modificar parámetros del usuario una vez que se halla registrado, a este módulo solo accederá el usuario con rol administrador.

El módulo de plan gráfico puede ser accedido solamente por los usuarios entrenadores. Aquí se llevará a cabo la planificación de las fechas de las pruebas médicas, competencias, campos de entrenamiento y test pedagógicos. Además estará dividido por macrosistemas, mesosistemas y microsistemas para una mayor organización.

Al módulo test pedagógico solo podrán acceder los entrenadores, los cuales podrán realizar los test pedagógicos donde se evaluarán pruebas de fuerza, resistencia, velocidad, flexibilidad, prueba teórica y prueba técnica.

En el módulo atleta es donde se gestionarán los atletas que pertenecen al equipo nacional de judo femenino, a éste módulo solo accederá el usuario entrenador.

En resumen, el sistema permitirá a los entrenadores almacenar toda la información que antes no podían guardar, haciendo posible su consulta en cualquier momento. Además el sistema hará posible digitalizar la realización de test pedagógicos, planes gráficos y llevar un registro con los datos de todas las atletas.

2.4 Modelo de negocio

Permite visualizar el alcance de la organización, representando lo que abarca y cuáles son sus límites. Así mismo, modela las actividades y procesos que ejecuta, señala gráficamente las funciones y metas que

persigue el negocio, y también permite identificar cuáles son los entregables y roles dentro de la organización.

2.4.1 Reglas del negocio

Una regla del negocio es la declaración de políticas y restricciones de negocio que el sistema debe cumplir. Consiste en definir una exigencia específica o invariable que debe satisfacer el negocio. Las reglas del negocio pueden aplicarse siempre o sólo bajo una condición específica. Es necesario que la aplicación muestre las restricciones que existen en el negocio, de tal forma que no sea posible realizar acciones no válidas.

- El entrenador es el encargado de realizar los test pedagógicos y el plan gráfico.
- Los test pedagógicos solo se pueden realizar según la planificación que tengan en el plan gráfico.
- Los test pedagógicos solo se pueden realizar después de haber elaborado el plan gráfico.
- Los test pedagógicos que se le realizarán a las atletas solo contendrán las siguientes pruebas:
    - ✓ Prueba Técnica
    - ✓ Prueba Teórica
    - ✓ Flexibilidad
    - ✓ Resistencia
    - ✓ Velocidad
    - ✓ Fuerza
- El nivel otorgado a las atletas después de haber realizado los test pedagógicos será: 1, 2, 3, 4 y 5.
- Cada macrosistema contendrá los siguientes mesosistemas:
    - ✓ Mesosistema de preparación física general (MPFG).
    - ✓ Mesosistema de preparación física especial variada (MPFEV).
    - ✓ Mesosistema de preparación física especial (MPFE).
    - ✓ Mesosistema de obtención de la forma deportiva (MOFD).

- ✓ Mesosistema de estabilización de la forma deportiva (MEFD).

2.4.2 Actores del Negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externo; con los que el negocio interactúa.

Actor	**Descripción**
Dirección de alto rendimiento del INDER.	Es el organismo encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y el Gobierno en cuanto a los programas deportivos, de educación física y recreación. Es el encargado de orientar la elaboración del plan de entrenamiento y quien se beneficia de la realización de los procesos del negocio.

Tabla 1: Actores del negocio.

2.4.3 Trabajadores del Negocio

Define el comportamiento y las responsabilidades de un individuo. Representa a un ser humano, software o *hardware* que desempeña un rol dentro de las realizaciones de un caso de uso del negocio.

Trabajador	Descripción
Entrenador	Es la figura central de un equipo deportivo. Trabaja en la elaboración del plan de entrenamiento; específicamente en la realización de los test pedagógicos y el plan gráfico.

Tabla 2: Trabajadores del negocio.

2.4.4 Descripción textual de los casos de uso del negocio

La descripción textual de los casos de uso del negocio, ayuda a comprender la lógica del mismo y como ocurre la relación actor-negocio.

Caso de Uso:	Realizar Plan Gráfico
Actores:	Dirección de alto rendimiento del INDER.
Trabajadores:	Entrenador
Resumen:	El caso de uso inicia cuando la dirección de alto rendimiento del INDER le solicita al entrenador realizar el plan gráfico. Este consiste en una planificación general con vista a la competencia fundamental del período. El caso de uso termina cuando el plan gráfico es discutido y aprobado por la dirección de alto rendimiento del INDER.
Flujo Normal de Eventos	
Sección "A"	
Acción del Actor	**Respuesta del Negocio**
A.1 La dirección de alto rendimiento del INDER solicita la realización del plan gráfico del año en vista a la competencia fundamental.	A.2 El entrenador registra la competencia fundamental y divide el plan gráfico en dos macrosistemas, sino pasar a la **Sección "A.2.a".**
	A.3 El entrenador divide cada macrosistema en cinco mesosistemas (mesosistema de preparación física general (MPFG), mesosistema de preparación física especial variada (MPFEV), mesosistema de preparación física especial (MPFE), mesosistema de

	obtención de la forma deportiva (MOFD) y mesosistema de estabilización de la forma deportiva (MEFD)).
	A.4 El entrenador a cada uno de estos mesosistemas le asigna una cantidad de microsistemas (semanas) determinada y la cantidad de sesiones, horas y minutos que se va a trabajar en cada mesosistema, además de la cantidad de combates a realizar, rango de tolerancia de peso, y el volumen e intensidad inicial y final con que se va a trabajar.
	A.5 El entrenador hace la planificación de los test pedagógicos, pruebas médicas, competencias y campos de entrenamientos en los que van a participar en cada microsistema.
	A.6 El entrenador envía a la dirección de alto rendimiento del INDER el plan gráfico terminado.
A.7 La dirección de alto rendimiento del INDER revisa el plan gráfico y lo aprueba, sino pasar a la **Sección "A.7.a"**	
Curso Alternativo de los Eventos.	
Sección "A.2.a"	
Acción del Actor	**Respuesta del Negocio**
	A.2.1 El entrenador registra la

	competencia fundamental y divide el plan gráfico en tres macrosistemas. Ir a la Sección A.3.
Sección "A.7.a"	
Acción del Actor	**Respuesta del Negocio**
A.7.1 La dirección de alto rendimiento del INDER no aprueba el plan gráfico, proponiéndole cambios para su posterior aprobación.	A.7.2 El entrenador hace las correcciones al plan gráfico. Ir a la Sección A.6.
Poscondiciones	-

Tabla 3: Descripción del caso de uso del negocio Realizar Plan Gráfico.

Caso de Uso:	Realizar Test Pedagógico
Actores:	Dirección de alto rendimiento del INDER.
Trabajadores:	Entrenador
Resumen:	El caso de uso inicia cuando el entrenador envía a la dirección de alto rendimiento del INDER el plan gráfico. El caso de uso termina cuando el entrenador obtiene los resultados de la aplicación del test pedagógico.
Flujo Normal de Eventos	
Sección "A"	
Acción del Actor	**Respuesta del Negocio**
	A.1 El entrenador envía a la dirección de alto rendimiento del INDER el plan gráfico.

	A.2 El entrenador reúne a las atletas para realizarles el test pedagógico.
	A.3 El entrenador aplica a las atletas el test pedagógico, el cual contiene las siguientes pruebas: - Fuerza (Pron, Halon, Cuclillas y Fuerza de Arranque). - Resistencia a la Fuerza (Barra Fija, Paralela, Soga, Escalera, Barra Invertida y Tubo). - Velocidad (50 metros). - Resistencia a la velocidad (200 metros). - Resistencia 1500 metros. - Flexibilidad (Flexión Ventral, Arqueo, Hiper-Extensión y Sapo) - Prueba Teórica. - Prueba Técnica (4 minutos de proyecciones, 30 segundos y 1 minuto).
	A.4 El entrenador registra los resultados alcanzados por cada una de las atletas.
	A.5 El entrenador otorga un nivel que puede ser de 1 a 5, según la división a la que pertenece la atleta, en función de los resultados obtenidos en el test pedagógico.

Poscondiciones	-

Tabla 4: Descripción del caso de uso del negocio Realizar Test Pedagógico.

2.4.5 Diagrama de Casos de Uso del Negocio

Describe el negocio en términos de casos de uso, que corresponde a lo que generalmente se le llama procesos.

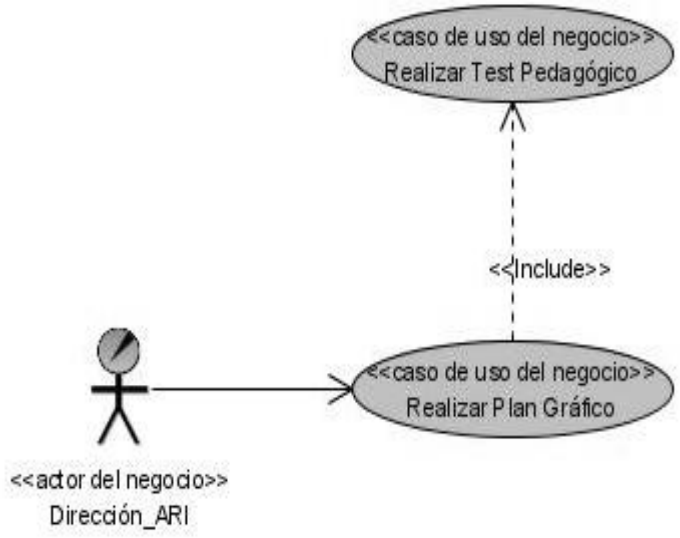


Figura 4: Diagrama de casos de uso del negocio.

2.4.6 Diagramas de Actividades

Describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio.

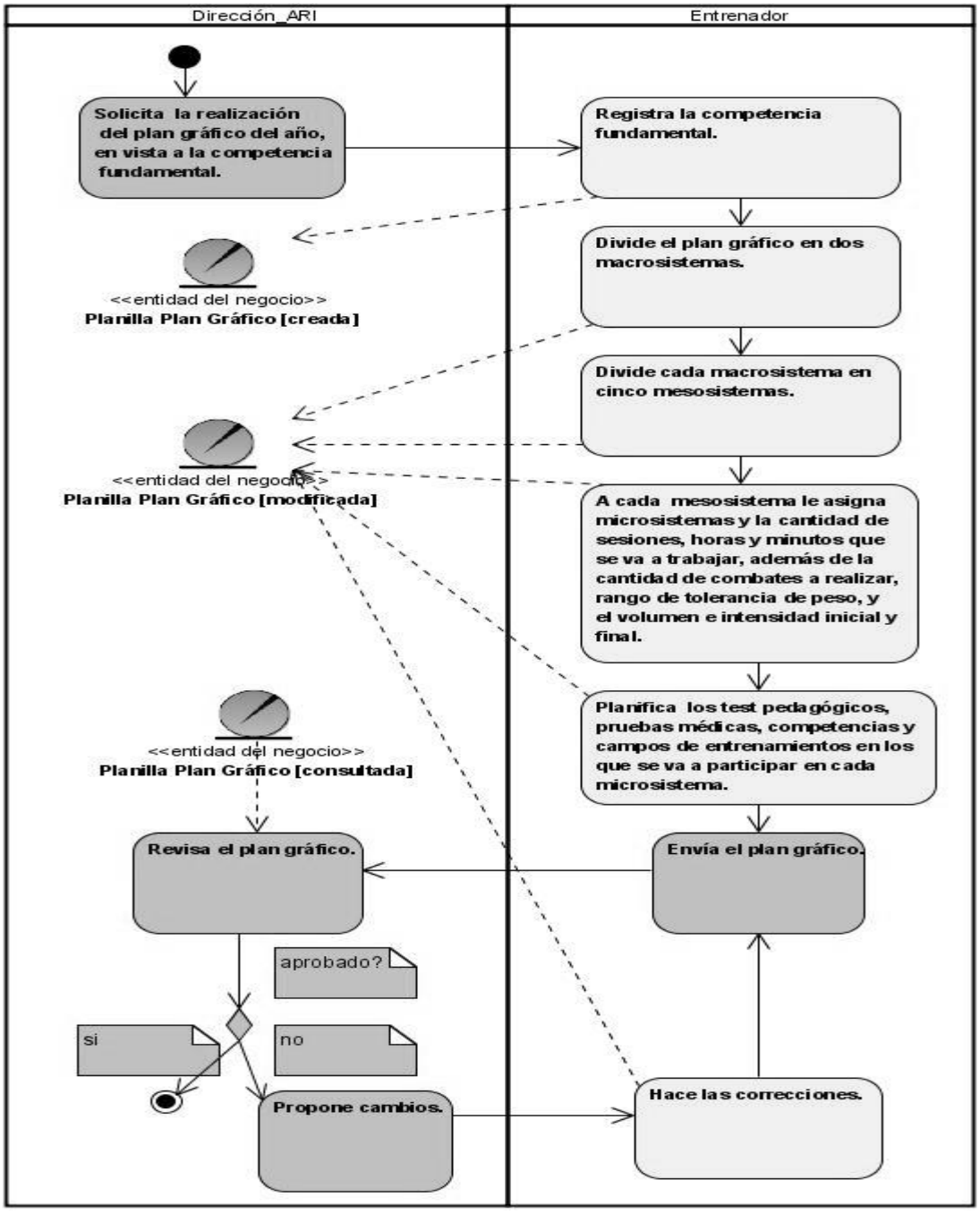


Figura 5: Diagrama de Actividades del caso de uso Realizar Plan Gráfico.

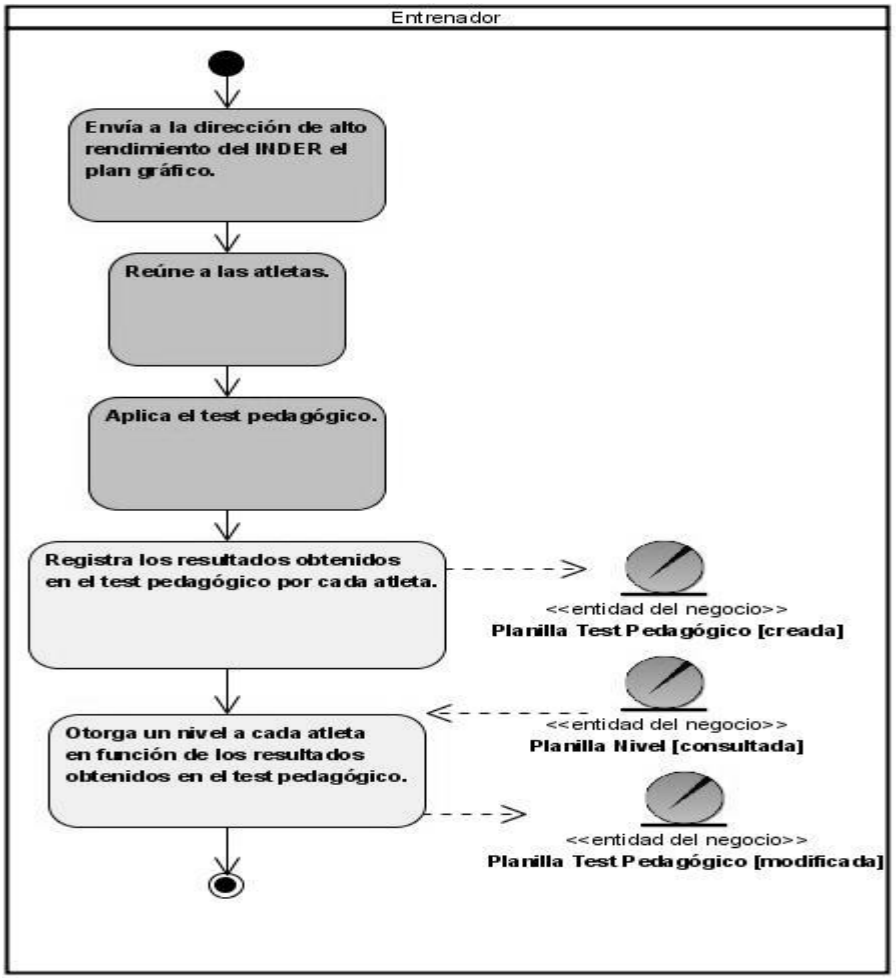


Figura 6: Diagrama de Actividades del caso de uso Realizar Test Pedagógico.

2.4.7 Modelo de Objetos del Negocio

El modelo de objetos del negocio describe cómo colaboran los trabajadores y las entidades del negocio dentro del flujo.

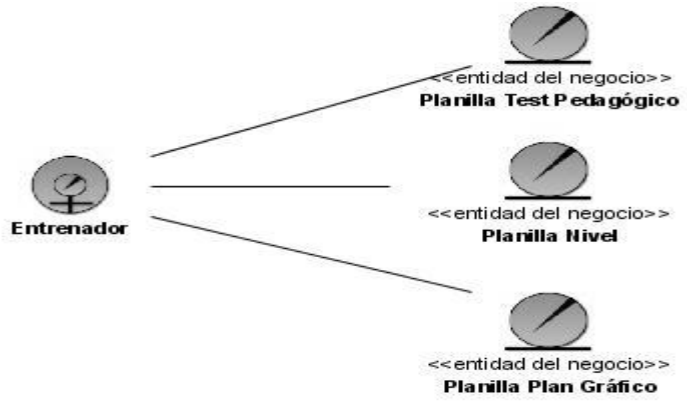


Figura 7: Modelo de objetos del negocio.

2.5 Requerimientos

2.5.1 Requerimientos Funcionales

Los requerimientos funcionales especifican las capacidades o condiciones que el sistema debe ser capaz de cumplir.

RF1 Gestionar Atleta

RF1.1 Insertar datos de un atleta: nombre, apellidos, número de CI, dirección particular (provincia, municipio, localidad, calle, número, entre calles), edad, edad deportiva, tiempo en el equipo nacional, foto, división, teléfono, nombre de la madre, nombre del padre, nivel de escolaridad, centro de estudio, organizaciones políticas, activo y número de pasaporte.

RF1.2 Consultar datos de un atleta

RF1.3 Modificar datos de un atleta

RF1.4 Exportar a PDF los datos de un atleta

RF2 Gestionar Test Pedagógico

RF2.1 Insertar test pedagógico: número macrosistema, cantidad de barra fija, paralela, soga, escalera, barra invertida, tubo, pron, halon, cuclilla, arranque de fuerza, velocidad 50 metros, velocidad 50 metros/volante, resistencia a la fuerza 200 metros, resistencia 1500 metros,

flexión ventral, arqueo, hiper-extensión, sapo, prueba teórica, prueba técnica (4 minutos de proyecciones, 30 segundos de proyecciones y 1 minuto de proyecciones).

RF2.2 Consultar test pedagógico

RF2.3 Modificar test pedagógico

RF2.4 Exportar a PDF los datos de un test pedagógico

RF3 Gestionar Plan Gráfico

RF3.1 Insertar plan gráfico: nombre del plan gráfico, año, competencia fundamental, lugar, para cada macrosistema poner la fecha de inicio y de fin de cada uno de sus mesosistemas, de cada mesosistema la cantidad de horas/sesiones, combates, rango tolerancia de peso inicial y final, volumen inicial y final e intensidad inicial y final, para cada microsistema los test pedagógicos, pruebas médicas, competencias y campos de entrenamiento que se van a realizar, además de la planificación de cada uno de los microsistemas de cada mesosistema.

RF3.2 Consultar plan gráfico

RF3.3 Modificar plan gráfico

RF3.4 Exportar a PDF los datos de un plan gráfico

RF4 Gestionar Usuario

RF4.1 Insertar un nuevo usuario. (Solicitar: nombre, apellidos, CI, usuario, contraseña y privilegio)

RF4.2 Modificar datos de un usuario

RF4.3 Eliminar cuenta de usuario

RF4.4 Consultar datos de un usuario registrado. (usuario, nombre y apellidos)

RF5 Autenticar Usuario

RF5.1 Permitir que el usuario entre al sistema dado el nombre de usuario y contraseña.

RF5.2 Permitir que el usuario cambie su contraseña una vez autenticado.

RF5.3 Permitir que el usuario pueda salir de la sesión de trabajo una vez autenticado.

RF5.4 Permitir que el usuario sólo pueda acceder a los recursos que tiene privilegio.

2.5.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales.

RNF1 Requerimiento de software

1.1 Servidor Web *Apache* 2.6.5

1.2 *Postgres* SQL 8.2.1

1.3 PHP 5

RNF2 Requerimiento de hardware

2.1 El servidor debe tener como mínimo las siguientes características: Una PC con procesador *Pentium* II, 1Gb de memoria RAM y 10Gb de capacidad en disco duro.

2.2 Las computadoras cliente deben tener como mínimo las siguientes características: Procesador *Pentium* III, 64 Mb de memoria RAM y 10Gb de capacidad en disco duro.

RNF3 Apariencia o interfaz externa

3.1 Interfaz con un diseño sencillo, de forma tal que facilite el uso a los usuarios que no tengan entrenamiento previo en sistemas con estas características; sin dejar de ser amigable, legible, interactiva y fácil de usar.

RNF4 Usabilidad

4.1 El sistema podrá ser usado por cualquiera de las personas que integra el colectivo técnico, por lo que es necesario que cuente con un diseño de interfaz de fácil uso.

RNF5 Rendimiento

5.1 La aplicación debe ser rápida y precisa. Estará implementada sobre una tecnología Web, facilitando su uso a través de la red.

RNF6 Portabilidad

6.1 Puede funcionar en cualquier sistema operativo, debido a que el servidor Web y el de base de datos son multiplataforma.

6.2 El producto debe correr sobre una plataforma Web, codificada en PHP y sus sistemas de base de datos en *PostgresSQL.*

RNF7 Requerimiento de Soporte

7.1 Servidor de BD debe soportar grandes volúmenes de datos y tener buena velocidad de procesamiento.

7.2 El sistema debe ser de fácil instalación.

7.3 El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

RNF8 Requerimientos de Seguridad

8.1 El sistema debe caracterizarse por su disponibilidad e integridad.

8.2 La información debe ser confidencial, para ello se pretende establecer un sistema de permisos y usuarios para el acceso a la información. Para mantener la integridad en el mismo solo se podrá acceder al sistema después de autenticarse encriptando la contraseña directamente en la máquina cliente utilizando el algoritmo de encriptación MD5, por lo que viaja hacia el servidor de forma segura.

RNF9 Confiabilidad

9.1 El sistema debe ser capaz de mantener la integridad de los datos.

9.2 Solo el personal autorizado podrá acceder a la información solicitada.

2.6 Modelo de casos de uso del sistema

Se basa en la descripción de elementos o usuarios externos al sistema (actores) y de la funcionalidad del sistema (casos de uso). Un modelo de casos de uso describe los requerimientos funcionales de un actor en términos de las interacciones que éste ejecuta con el sistema. El modelado de casos de uso es una técnica efectiva y a la vez simple para modelar los requerimientos del sistema desde la perspectiva del

usuario. Presenta el sistema desde la perspectiva de su uso y esquematiza como proporcionará valor a sus usuarios.

2.6.1 Descripción de los actores del sistema

Son las personas cuyo rol en el negocio de trabajadores los llevan a ser actores del sistema.

Actor	Descripción
Entrenador	En el sistema es la persona encarga de gestionar el plan gráfico, los test pedagógicos y los atletas.
Administrador_Sistema	Es el encargado de gestionar a los usuarios y asignarles el rol que les corresponde.
Usuario (Entrenador y Administrador_Sistema)	Son todas las personas registradas que se autentican en el sistema.

Tabla 5: Descripción de los actores del sistema.

2.6.2 Diagrama de casos de uso del sistema

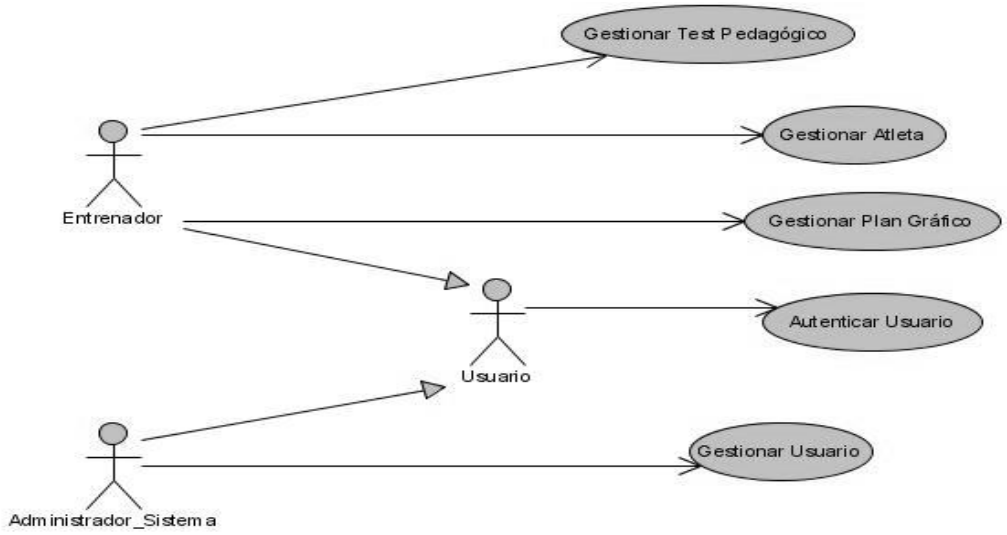


Figura 8: Diagrama de casos de uso del sistema.

2.6.3 Descripción de los casos de uso del sistema

Caso de uso	
	Gestionar Atleta.
Propósito	Insertar, consultar, modificar y exportar a PDF los datos de un atleta.
Actor: Entrenador	
Resumen: El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un atleta en el sistema. El actor puede insertar, consultar, modificar y exportar a PDF los datos de un atleta. En caso de que seleccione la opción de insertar un nuevo atleta, el sistema dará la posibilidad de introducir los datos de un atleta tales como: nombre, apellidos, número de CI, dirección particular (provincia, municipio, localidad, calle, número, entre calles), edad, edad deportiva, tiempo en el equipo nacional, foto, división, teléfono, nombre de la madre, nombre del padre, nivel de escolaridad, centro de estudio, organizaciones políticas, activo y número de pasaporte. Si el actor selecciona la opción de consultar los datos de un atleta, el sistema mostrará un listado con los siguientes datos de las atletas (foto, carné, nombre, apellidos, división, edad, provincia y pasaporte), además brindará la posibilidad de exportar a PDF los datos de un atleta seleccionado. Si el actor selecciona la opción de modificar los datos de un atleta, el sistema le permitirá seleccionar el atleta al cual desea modificar los datos y mostrará los datos que pueden ser modificados (nombre, apellidos, dirección particular (provincia, municipio, localidad, calle, número, entre calles), edad, edad deportiva, tiempo en el equipo nacional, foto, división, teléfono, nombre de la madre, nombre del padre, nivel de escolaridad, centro de estudio, organizaciones políticas, activo y número de pasaporte), y una vez realizados los cambios, guardará las modificaciones dando fin al caso de uso.	

Tabla 6: Descripción del caso de uso del sistema Gestionar Atleta.

Caso de uso	
	Gestionar Test Pedagógico.

Propósito	Insertar, consultar, modificar y exportar a PDF los datos de un test pedagógico.
Actor: Entrenador	
Resumen: El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un test pedagógico en el sistema. El actor puede insertar, consultar, modificar y exportar a PDF los datos de un test pedagógico. En caso de que seleccione la opción de insertar un nuevo test pedagógico, el sistema dará la posibilidad de introducir los datos de un test pedagógico tales como: número macrosistema, cantidad de barra fija, paralela, soga, escalera, barra invertida, tubo, pron, halon, cuclilla, arranque de fuerza, velocidad 50 metros, velocidad 50 metros/volante, resistencia a la fuerza 200 metros, resistencia 1500 metros, flexión ventral, arqueo, hiper-extensión, sapo, prueba teórica, prueba técnica (4 minutos de proyecciones, 30 segundos de proyecciones y 1 minuto de proyecciones). Si el actor selecciona la opción de consultar los datos de un test pedagógico, el sistema permitirá seleccionar el test pedagógico que desea consultar según el número del test pedagógico, macrosistema y plan gráfico al que pertenece, además brindará la posibilidad de exportar a PDF los datos del test pedagógico consultado. Si el actor selecciona la opción de modificar los datos de un test pedagógico, el sistema le permitirá seleccionar el test pedagógico al cual desea modificar los datos y el número del macrosistema al cual pertenece y mostrará los datos que pueden ser modificados, y una vez realizados los cambios guardará las modificaciones, dando fin al caso de uso.	

Tabla 7: Descripción del caso de uso del sistema Gestionar Test Pedagógico.

Caso de uso	
	Gestionar Plan Gráfico.
Propósito	Insertar, consultar, modificar y exportar a PDF los datos de un plan gráfico.
Actor: Entrenador	

Resumen: El caso de uso inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un plan gráfico en el sistema. El actor puede insertar, consultar, modificar y exportar a PDF los datos de un plan gráfico. En caso de que seleccione la opción de insertar un nuevo plan gráfico, el sistema dará la posibilidad de introducir los datos de un plan gráfico tales como: nombre del plan gráfico, año, competencia fundamental, lugar, para cada macrosistema poner la fecha de inicio y de fin de cada uno de sus mesosistemas, de cada mesosistema la cantidad de horas/sesiones, combates, rango tolerancia de peso inicial y final, volumen inicial y final e intensidad inicial y final, para cada microsistema los test pedagógicos, pruebas médicas, competencias y campos de entrenamiento que se van a realizar, además de la planificación de cada uno de los microsistemas de cada mesosistema. Si el actor selecciona la opción de consultar los datos de un plan gráfico, el sistema permitirá seleccionar el plan gráfico que desea consultar y mostrará los datos del mismo, además brindará la posibilidad de exportar a PDF los datos del plan gráfico consultado. Si el actor selecciona la opción de modificar los datos de un plan gráfico, el sistema le permitirá seleccionar el plan gráfico al cual desea modificarle los datos y mostrará los datos que pueden ser modificados, y una vez realizados los cambios, guardará las modificaciones.

Tabla 8: Descripción del caso de uso del sistema Gestionar Plan Gráfico.

Caso de uso	
	Gestionar Usuario.
Propósito	Insertar, consultar y modificar los datos de un usuario, así como eliminar la cuenta de un usuario existente.
Actor: Administrador_ Sistema	

Resumen: El caso de uso se inicia cuando el actor selecciona la opción que le permite realizar una acción sobre un usuario en el sistema. El actor puede insertar, consultar y modificar los datos de un usuario, así como eliminar la cuenta de un usuario existente. En caso de que seleccione la opción de insertar un nuevo usuario, el sistema dará la posibilidad de introducir los siguientes datos: nombre, apellidos, CI, usuario, contraseña y privilegio, una vez introducidos dichos datos, los almacenará. Si el atleta selecciona la opción de consultar los datos de los usuarios, el sistema le mostrará los datos de los usuarios tales como: nombre, apellidos y usuario. Si el actor selecciona la opción de modificar los datos de un usuario, el sistema brindará la posibilidad de seleccionar el usuario deseado y mostrará los datos que pueden ser modificados como son: nombre, apellidos, contraseña y privilegio, luego guardará las modificaciones realizadas. Si el actor selecciona la opción de eliminar una cuenta de un usuario, el sistema brindará la posibilidad de seleccionar el usuario deseado y procederá a realizar esta operación dando fin al caso de uso.

Tabla 9: Descripción del caso de uso del sistema Gestionar Usuario.

Caso de uso	
	Autenticar Usuario.
Propósito	Entrar al sistema dado el nombre de usuario y contraseña, cambiar contraseña una vez autenticado, salir de la sesión de trabajo una vez autenticado y permitir al usuario acceder solo a los recursos que tiene privilegios.
Actor: Usuario	

El caso de uso se inicializa cuando el actor introduce en la interfaz de autenticación su usuario y contraseña, el sistema deberá comprobar que son correctos para permitir su acceso a los recursos que están disponibles, teniendo en cuenta los privilegios con los que cuenta. El actor puede cambiar su contraseña y salir de la sesión de trabajo, una vez autenticado en el sistema. Si el actor selecciona la opción de cambiar su contraseña, el sistema deberá solicitar que introduzca su usuario, contraseña anterior y la nueva contraseña, luego deberá comprobar que son válidos los datos introducidos y en este caso guardará las modificaciones. Si el actor selecciona la opción de salir de la sesión de trabajo, el sistema procederá a realizar esta operación, dando fin al caso de uso.

Tabla 10: Descripción del caso de uso del sistema Autenticar Usuario.

2.7 Conclusiones

En este capítulo se realizó un estudio detallado del negocio y se comenzó a desarrollar la propuesta de solución del sistema. Además se realizó el levantamiento de requisitos funcionales y no funcionales, y se agruparon los funcionales en casos de uso, con los cuales se conformó el diagrama de casos de uso, el cual constituye la base del futuro sistema.

Capítulo 3 Análisis y diseño del sistema

3.1 Introducción

En este capítulo se abordará el tema de análisis y diseño de la propuesta del sistema, específicamente la confección de los diagramas de clases del análisis y del diseño. También se seleccionará el tipo de arquitectura a utilizar y se describirá cada una de las clases del diseño. Además se van a elaborar los diagramas de interacción del diseño.

3.2 ¿Qué es el análisis y el diseño?

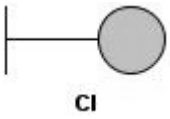
El propósito del análisis y el diseño es transformar los requerimientos en un diseño de lo que será el sistema, además de desarrollar una arquitectura robusta para el sistema y adaptar el diseño para que sea consistente en el entorno de implementación. En la fase de inicio, la disciplina de análisis y diseño se preocupa por establecer si la visión del sistema es factible, y en determinar las tecnologías potenciales para la solución de software. Si se considera que pocos riesgos se asocian al desarrollo (porque el dominio se entiende muy bien, el sistema no es novedoso o cualquier otra razón parecida) entonces este aspecto se omite del flujo de trabajo. En la parte inicial de la fase de elaboración se enfoca el esfuerzo en definir una arquitectura candidata del sistema. Durante la fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes. El resultado es un conjunto inicial de componentes los cuales en un futuro son refinados dentro de la implementación.

3.3 Modelo de Análisis

Este modelo es usado para representar la estructura global del sistema, describe la realización de casos de uso, sirve como una abstracción del modelo de diseño y se centra en los requerimientos funcionales. Consiste en obtener una visión del sistema que se preocupa de ver que debe hacer, sin tener que preocuparse por cuestiones propias del lenguaje.

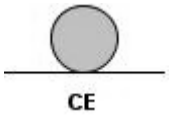
RUP propone clasificar a las clases del análisis en:

Interfaz: Modelan la interacción entre el sistema y sus actores.



Control: Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

Entidad: Modelan información que posee larga vida y que es a menudo persistente.



3.3.1 Diagrama de clases del análisis

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas.

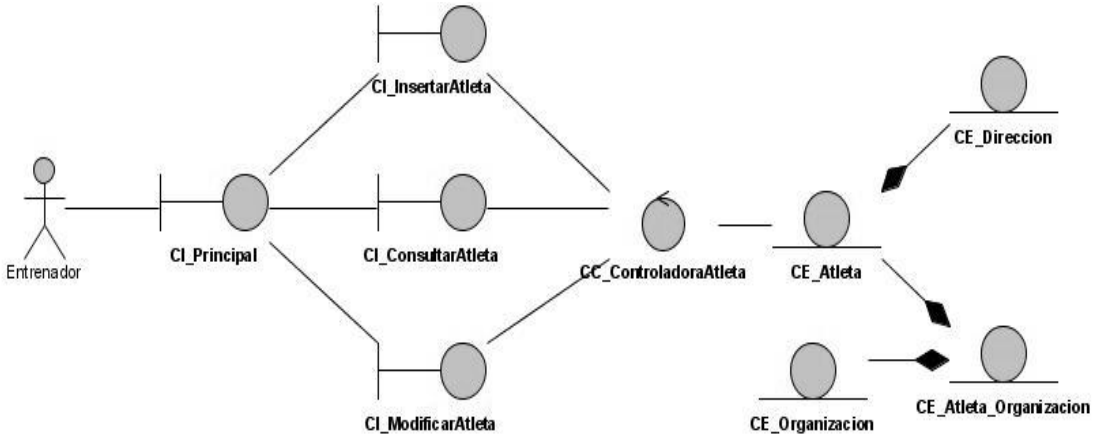


Figura 9: Diagrama del análisis caso de uso gestionar atleta.

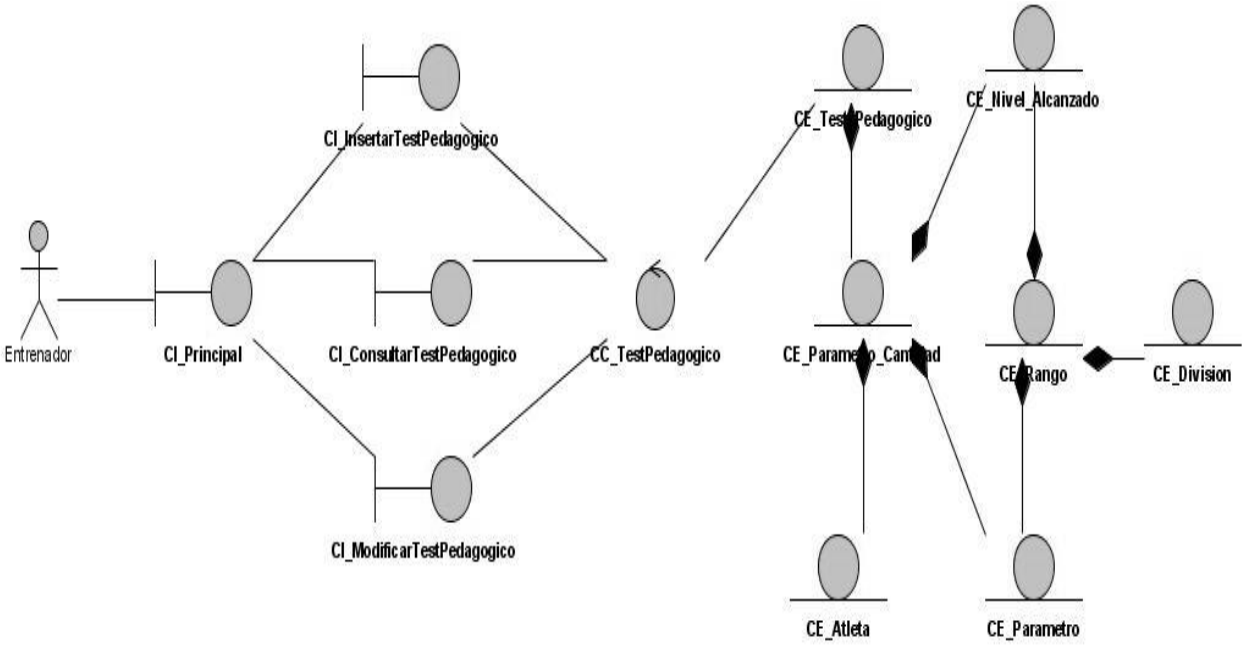


Figura 10: Diagrama del análisis caso de uso gestionar test pedagógico.

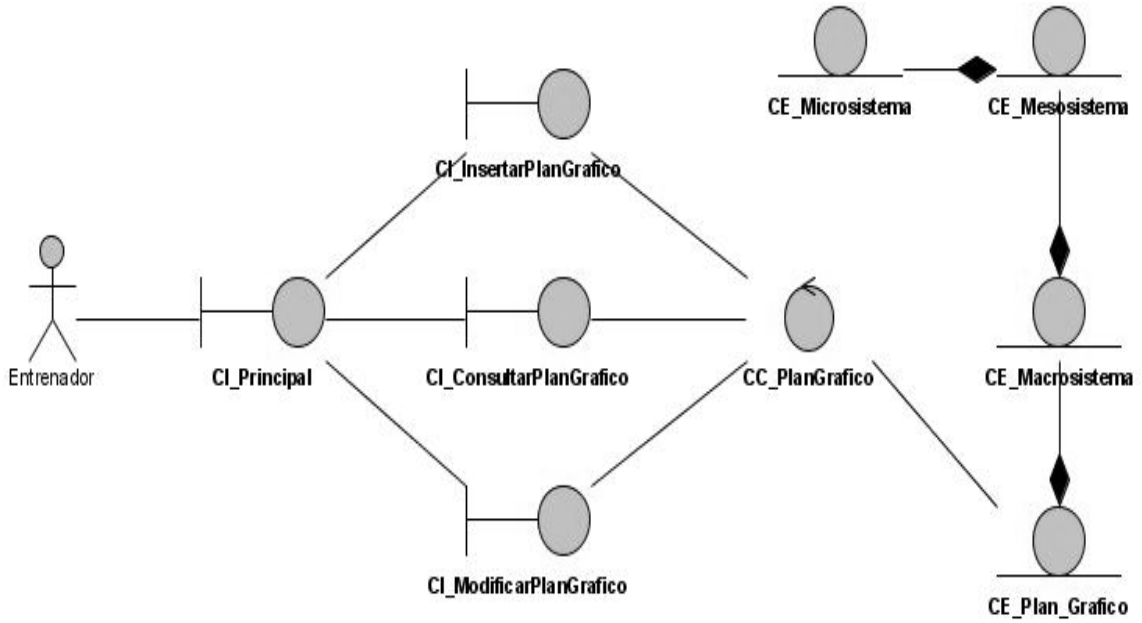


Figura 11: Diagrama del análisis caso de uso gestionar plan gráfico.

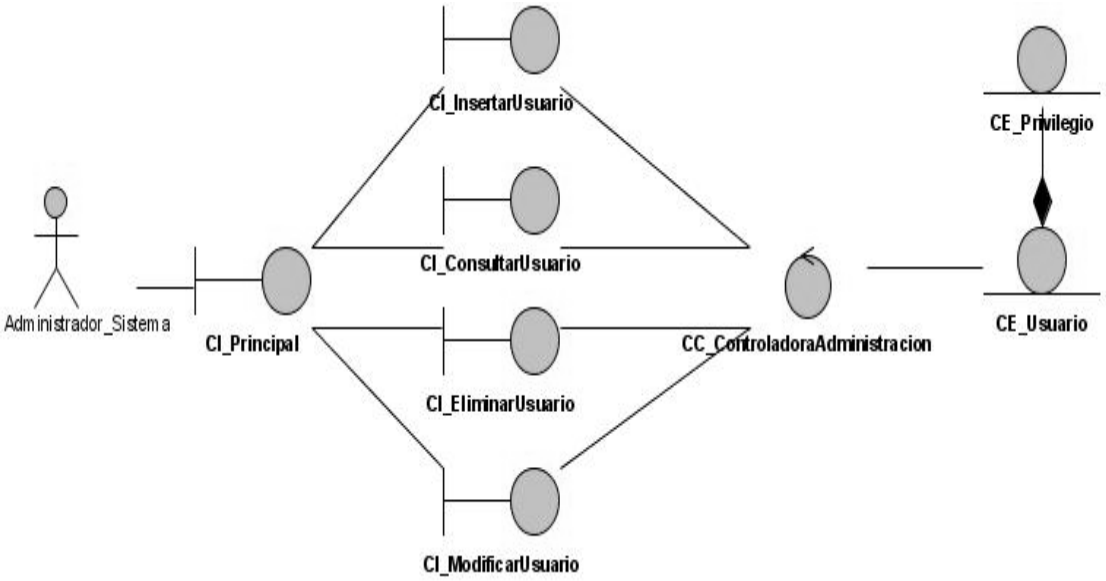


Figura 12: Diagrama del análisis caso de uso gestionar usuario.

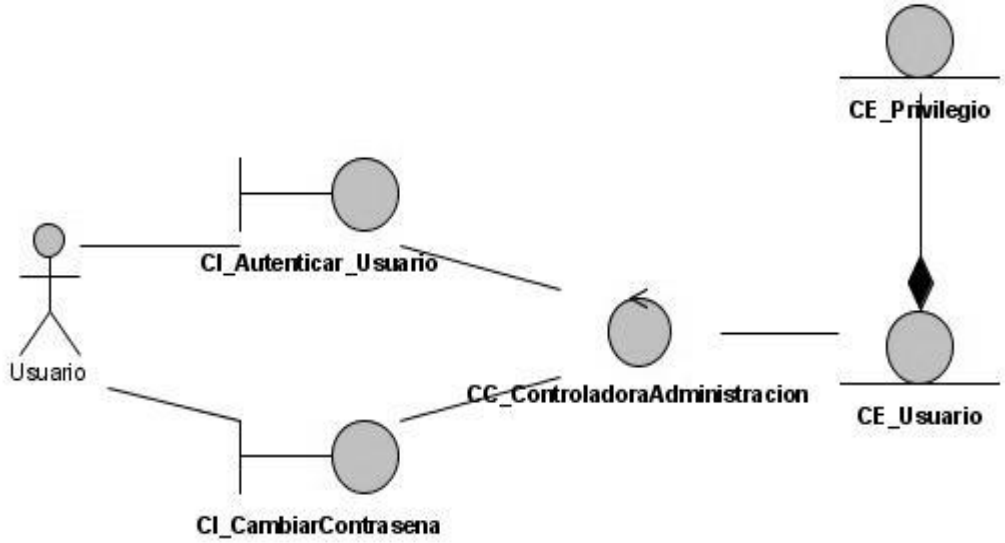


Figura 13: Diagrama del análisis caso de uso autenticar usuario.

3.4 Modelo de Diseño

El diseño es un refinamiento que tiene en cuenta los requerimientos no funcionales, por lo cual se centra en cómo el sistema cumple sus objetivos. Es una abstracción del modelo de implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño, es usado como

entrada esencial en las actividades relacionadas a implementación. Representa a los casos de uso en el dominio de la solución.

3.4.1 Diseño Arquitectónico

La estrategia tradicional de utilizar aplicaciones compactas causa cantidad de problemas de integración en sistemas de software complejos consistentes en más de una aplicación. Estas aplicaciones suelen encontrarse con importantes problemas de escalabilidad, disponibilidad, seguridad, integración, etc. Para resolver estos problemas se ha generalizado la división de las aplicaciones en capas logrando de esta manera obtener una potente arquitectura que brinda algunas ventajas:

- Centralización de los aspectos de seguridad y transaccionalidad, que serían responsabilidad del modelo.
- No replicación de lógica del negocio en los clientes: esto permite que las modificaciones y mejoras sean automáticamente aprovechadas por el conjunto de los usuarios, reduciendo los costes de mantenimiento.
- Mayor sencillez de los clientes.

Para la implementación del sistema propuesto el tipo de arquitectura utilizada en tecnologías cliente-servidor fue el diseño arquitectónico en 3 capas.

- **Capa de Datos**: Esta capa es la encargada de abstraer los datos a utilizar en la aplicación. Aquí se puede utilizar un sistema gestor de base de datos.
- **Capa de Aplicación o Lógica del Negocio**: Esta capa se encarga de la intercomunicación entre la capa de datos y la de presentación.
- **Capa de Presentación**: Es la capa donde se presentan los datos al usuario, puede ser a través de HTML, archivos de texto, etc.

3.4.2 Principios del Diseño

Se tuvieron en cuenta algunos principios a la hora de diseñar la aplicación, los cuales fueron:

- Uso equiparable: Las características de privacidad, garantía y seguridad deben estar igualmente disponibles para todos los usuarios. El diseño debe ser atractivo.

- Uso flexible: Ofrecer posibilidades de elección en los métodos de uso, facilitar al usuario la exactitud y precisión.
- Simple e intuitivo: Que elimine la complejidad innecesaria y proporcione avisos eficaces y métodos de respuesta durante y tras la finalización de la tarea.
- Tolerancia al error: Se ofrecerá la posibilidad de que el usuario pueda cancelar alguna acción que haya realizado en el sistema, como eliminar y/o modificar datos del sistema. Además que disponga de elementos para minimizar los riesgos y errores.

3.4.3 Descripción de las clases del diseño

Las clases del diseño se describirán de la siguiente manera:

- Nombre de la clase: el nombre de la clase se escribirá con mayúscula, comenzará con un identificador del tipo de clase (entidad: CED, controladora: CCD, acceso a datos: AD, lógica de interfaz de usuario: SP, interfaz de usuario: CP) seguido de un guión bajo (_), por ejemplo para la clase entidad Usuario se escribiría: CED_Usuario.
- Los métodos se escribirán con mayúscula y los atributos con minúscula, y las clases, los métodos y los atributos no se tildarán.

Las clases entidad modelan información que posee larga vida y que es a menudo persistente. Estas clases estan compuestas por atributos y métodos, los métodos que las componen son:

- Un constructor: es un método (función) que se ejecuta automáticamente al crear una nueva instancia (objeto) de una clase determinada, se utiliza principalmente para inicializar sus atributos.
- Métodos get: se utilizan para consultar la información que poseen los atributos de una clase.
- Métodos set: se utilizan para modificar la información que contienen los atributos de una clase.

Las clases entidad que se definen en el sistema son:

Atributos	**Tipo**
nombre	Varchar
apellidos	Varchar
ci	Bigint

usuario	Varchar
contrasena	Varchar

Tabla 11: Descripción de la CED_Usuario.

Atributos	**Tipo**
tipo	Varchar
lista_usuario	Lista

Tabla 12: Descripción de la clase CED_Privilegio.

Atributos	**Tipo**
nombre	Varchar
apellidos	Varchar
ci	Bigint
edad	Int
edad_deportiva	Int
tiempo_eq	Int
foto	Varchar
division	Varchar
telefono	Varchar
nombre_madre	Varchar
nombre_padre	Varchar
nivel_escolaridad	Varchar
centro_estudio	Varchar
activo	Bolean
numero_pasaporte	Int
obj_direccion	direccion

Tabla 13: Descripción de la clase CED_Atleta.

Atributos	**Tipo**
nombre	Varchar

Tabla 14: Descripción de la clase CED_Organizacion.

Atributos	Tipo
provincia	Varchar
municipio	Varchar
localidad	Varchar
calle	Varchar
numero	Int
entre_calles	Varchar

Tabla 15: Descripción de la clase CED_Direccion.

Atributos	Tipo
ci	Bigint
nombre	Varchar

Tabla 16: Descripción de la clase CED_Atleta_Organizacion.

Atributos	Tipo
numero_macro	Varchar
lista_mesosistema	Lista
lista_competencia	Lista
lista_test_pedagogico	Lista

Tabla 17: Descripción de la clase CED_Macrosistema.

Atributos	Tipo
nombre_pg	Varchar
competencia_fundamental	Varchar
lugar	Varchar
pronostico	Varchar
resultado	Varchar
lista_macrosistema	Lista

Tabla 18: Descripción de la clase CED_Plan_Grafico.

Atributos	**Tipo**
nombre_meso	Varchar
fecha_inicio	Date
fecha_fin	Date
periodo	Varchar
sesiones	Int
horas	Int
minutos	Int
combates	Int
rtp_minimo	Real
rtp_maximo	Real
volumen_minimo	Real
volumen_maximo	Real
intensidad_minimo	Real
intensidad_maximo	Real
lista_microsistema	Lista

Tabla 19: Descripción de la clase CED_Mesosistema.

Atributos	**Tipo**
numero_micro	Int
fecha_inicio	Date
fecha_fin	Date
test_pedagogico_micro	Bolean
prueba_medica_micro	Bolean
competencia_micro	Bolean
campo_entrenamiento_micro	Bolean
lmv	Varchar
mjs	Varchar

Tabla 20: Descripción de la clase CED_Microsistema.

Atributos	Tipo
numero_test	Int
fecha_inicio	Date
fecha_fin	Date
lista_atleta	Lista

Tabla 21: Descripción de la clase CED_Test_Pedagogico.

Atributos	Tipo
nombre_parametro	Varchar

Tabla 22: Descripción de la clase CED_Parametro.

Atributos	Tipo
cantidad	Varchar

Tabla 23: Descripción de la clase CED_Parametro_Cantidad.

Atributos	Tipo
minimo	Real
maximo	Real

Tabla 24: Descripción de la clase CED_Rango.

Atributos	Tipo
nivel	Int

Tabla 25: Descripción de la clase CED_Nivel_Alcanzado.

Atributos	Tipo
nombre_division	Varchar

Tabla 26: Descripción de la clase CED_Division.

Las clases de acceso a datos son las encargadas de realizar las consultas a la base de datos. Estas tienen un constructor con el mismo nombre de la clase, el cual no contiene parámetros.

Las clases de acceso a datos que se definen en el sistema son:

Métodos	Descripción

InsertarUsuario($ci,$tipo,$nombre,$apellidos,$usuario,$contrasena)	Permite insertar un nuevo usuario en el sistema.
ListaPrivilegios()	Muestra una lista de los diferentes tipos de privilegios que puede tener un usuario.
ListaUsuarios()	Devuelve una lista con todos los usuarios del sistema.
ModificarUsuario($ci,$tipo,$nombre,$apellidos,$usuario,$contrasena)	Permite modificar los datos de un usuario.
EliminarUsuario($ci)	Permite eliminar un usuario.

Tabla 27: Descripción de la clase AD_AccesoDatosAdministracion.

Métodos	**Descripción**
InsertarAtleta($ci,$nombre,$apellidos,$edad,$edadd,$tiempo,$foto,$division,$telefono,$nombrep,$nombrem,$nivel,$centro,$activo,$pasaporte,$calle,$numero,$entre,$local,$muni,$pro,$o1,$o2)	Permite insertar un nuevo atleta en la aplicación.
ModificarAtleta($ci,$nombre,$apellidos,$edad,$edadd,$tiempo,$foto,$division,$telefono,$nombrep,$nombrem,$nivel,$centro,$activo,$pasaporte,$calle,$numero,$entre,$local,$muni,$pro,$o1,$o2)	Permite modificar los datos de una atleta.
ListaAtletas()	Muestra una lista de todos los atletas que existen en el sistema.
ListaAtletasOrganizacion()	Lista todas las organizaciones que hay en la base de datos de todos los atletas.
Direccion($ci)	Permite obtener la dirección de un atleta determinado.

Tabla 28: Descripción de la clase AD_AccesoDatosAtleta.

Métodos	Descripción
ListaPG()	Muestra una lista con todos los planes gráficos planificados.
ListaMacro($pg)	Dado un plan gráfico determinado, muestra una lista con todos los macrosistemas que contiene.
ListaMeso($pg,$num)	Dado un plan gráfico y el número de un macrosistema determinado, muestra una lista con todos los mesosistemas que contiene.
ListaMicro($pg,$num,$meso)	Dado un plan gráfico, el número de un macrosistema y el nombre de un mesosistema determinado, muestra una lista con todos los microsistemas que contiene.
BuscarTP($pg,$num,$meso)	Dado un plan gráfico, el número de un macrosistema y el nombre de un mesosistema determinado, busca todos los test pedagógicos que se planificaron.
GuardarMicro($micro,$pg,$num,$meso,$tp,$pm,$c,$ce)	Permite guardar un microsistema con su número, el plan gráfico, el número del macrosistema y el nombre del mesosistema al que pertenece, además de la planificación de las pruebas médicas, test pedagógicos, competencias y campos de entrenamiento.
GuardarMicro1($micro,$pg,$num,$meso,$l,$m)	Permite guardar la planificación de los ejercicios que se van a realizar diariamente.
GuardarTodo($pg,$num,$meso,$fi,$ff,$pe,$hs)	Guarda un plan gráfico, un macrosistema y un mesosistema con los valores por defecto.
SesionesMesoMacro($pg,$num,$meso)	Dado un plan gráfico, un macrosistema y un mesosistema determinado, devuelva las

	sesiones planificadas para ese mesosistema.
HorasMesoMacro($pg,$num,$meso)	Dado un plan gráfico, un macrosistema y un mesosistema determinado, devuelva las horas planificadas para ese mesosistema.
MinutosMesoMacro($pg,$num,$meso)	Dado un plan gráfico, un macrosistema y un mesosistema determinado, devuelva los minutos planificados para ese mesosistema.
GuardarMeso($meso,$pg,$num,$comb,$rtp1,$rtp2,$v1,$v2,$i1,$i2)	Permite guardar un mesosistema con su nombre, el plan gráfico al que pertenece, el número del macrosistema al que pertenece, cantidad de combates a realizar, el rango de tolerancia de peso mínimo y máximo, volumen mínimo y máximo, y la intensidad mínima y máxima de cada mesosistema.
ModificarMeso($pg,$num,$meso,$fi,$ff,$hs)	Permite modificar un mesosistema determinado.
LMV($micro,$pg,$num,$meso)	Permite modificar la planificación del entrenamiento para los días lunes, miércoles y viernes.
MJS($micro,$pg,$num,$meso)	Permite modificar la planificación del entrenamiento para los días martes, jueves y sábado.
InsertarPG($nombre,$cf,$l,$p)	Permite insertar un nuevo plan gráfico.
ModificarPG($nombre,$cf,$l,$p,$r)	Permite modificar un plan gráfico determinado.
BuscarMesoTP($pg,$num,$meso)	Dado un plan gráfico, un macrosistema y un mesosistema, permite buscar todos los test pedagógicos que se planificaron en ese mesosistema.
BuscarFechaMeso()	Busca la fecha de un mesosistema

	determinado.

Tabla 29: Descripción de la clase AD_AccesoDatosPlanGrafico.

Métodos	**Descripción**
ListaPG()	Muestra una lista con todos los planes gráficos.
ListaAtletasActivo()	Lista todos los atletas que se encuentran activos, o sea todos los atletas que no son baja.
ListaAtletas()	Muestra una lista de todos los atletas que existen en el sistema.
ListaParametros()	Muestra una lista con los resultados de todos los parámetros de un test pedagógico.
ListaRangos()	Devuelve una lista de rangos.
ListaParametro_Cantidades()	Devuelve una lista con todos los parametros_cantidad.
ListaCITP($pg,$macro,$tp)	Devuelve una lista con el carné de identidad de todos los atletas que realizaron un test pedagógico determinado.
ListaTestPed()	Devuelve una lista de todos los test pedagógicos.
ListaMacro($pg)	Dado un plan gráfico determinado, muestra una lista con todos los macrosistemas que contiene.
ListaTP_PG($pg,$macro)	Permite conocer la cantidad de test pedagógicos planificados en un macrosistema de un plan gráfico determinado.
ListaDirecciones()	Devuelve la cantidad de test pedagógicos hechos en un plan gráfico y un macrosistema determinado.
ListaDivisiones()	Devuelve una lista con todas las divisiones.
PGActual()	Devuelve el plan gráfico con que se esta trabajando en ese momento.
CantidadTPPlanificado($pg,$macr	Permite conocer la cantidad de test pedagógicos

o)	planificados en un macrosistema de un plan gráfico determinado.
CantidadTPHecho($pg,$macro)	Devuelve la cantidad de test pedagógicos hechos en un plan gráfico y un macrosistema determinado.
AdicionarParametro_Cantidad($pci,$nombre,$cantidad,$id,$pg,$macro)	Permite adicionar un parametro_cantidad.
MostrarParametro_Cantidad($ci,$parametro,$cantidad)	Permite mostrar un parametro_cantidad.
AdicionarTP($id,$pg,$macro,$fi,$ff)	Permite adicionar un nuevo test pedagógico.
FechaInicioTP($pg,$macro)	Muestra la fecha de inicio del test pedagógico a realizar.
FechaFinTP($pg,$macro)	Muestra la fecha de fin del test pedagógico a realizar.
FechaFinTPM($pg,$macro,$tp)	Muestra la fecha de fin de un test pedagógico determinado.
FechaInicioTPM($pg,$macro,$tp)	Muestra la fecha de inicio de un test pedagógico determinado.
CantidadTP($pg,$macro,$tp,$ci,$pa)	Devuelve la cantidad de ejercicios de un parámetro determinado hechos por un atleta en un test pedagógico.
CantidadFisica($pg,$macro,$tp,$ci)	Muestra la cantidad física de un test pedagógico.
CantidadTecnica($pg,$macro,$tp,$ci)	Muestra la cantidad técnica de un test pedagógico.
NivelTP($pg,$macro,$tp,$ci,$pa)	Muestra el nivel por ejercicios de un atleta en un test pedagógico determinado.

BuscarNombreMesoTP($pg,$macro,$tp)	Dado el nombre de un plan gráfico, el número de un macrosistema y un test pedagógico determinado, devuelve el nombre del mesosistema donde se encuentra planificado el test pedagógico.

Tabla 30: Descripción de la clase AD_AccesoDatosTestPed.

Las clases controladoras permiten manejar los eventos del caso de uso, transacciones; usualmente encapsulan el control de un caso de uso. Estas clases contienen todos los métodos que permiten el acceso de la capa de presentación a los datos a través de la aplicación. Estas tienen un constructor con el mismo nombre de la clase, el cual no contiene parámetros.

Las clases controladoras que se definen en el sistema son:

Atributos	**Tipo**
listausuario	Lista
Métodos	**Descripción**
GetListaUsuario()	Devuelve una lista con todos los usuarios.
SetListaUsuario($plista)	Permite modificar una lista con todos los usuarios.
BuscarUsuario($ci)	Dado el carné de identidad, busca al usuario.
BuscarUsuarioU($usuario)	Dado un usuario permite conocer todos sus datos.
InsertarUsuario($ci,$tipo,$nombre,$apellidos,$usuario,$contrasena)	Permite insertar un nuevo usuario en el sistema.
ModificarUsuario($ci,$tipo,$nombre,$apellidos,$usuario,$contrasena)	Permite modificar los datos de un usuario.
EliminarUsuario($ci,$usuario)	Permite eliminar un usuario.

Tabla 31: Descripción de la CCD_ControladoraAdministracion.

Atributos	**Tipo**
listaAtleta	Lista
Métodos	**Descripción**

GetListaAtleta()	Devuelve una lista con todos los atletas.
SetListaAtleta($plista)	Permite modificar una lista con todos los atletas.
ListaAtletasActivos()	Muestra una lista de todos los atletas activos del sistema.
BuscarAtleta($ci)	Busca a un atleta, dado el carné de identidad.
InsertarAtleta($ci,$nombre,$apellidos,$edad,$edadd,$tiempo,$foto,$division,$telefono,$nombrep,$nombrem,$nivel,$centro,$activo,$pasaporte,$calle,$numero,$entre,$local,$muni,$pro,$o1,$o2)	Permite insertar un nuevo atleta en la aplicación.
ModificarAtleta($ci,$nombre,$apellidos,$edad,$edadd,$tiempo,$foto,$division,$telefono,$nombrep,$nombrem,$nivel,$centro,$activo,$pasaporte,$calle,$numero,$entre,$local,$muni,$pro,$o1,$o2)	Permite modificar los datos de una atleta.
ListaAtletasOrganizacionActivos()	Lista todas las organizaciones que hay en la base de datos de los atletas que están activos.
BuscarAtletaOrganizacion($ci,$nombre)	Muestra todas las organizaciones a las que pertenece un atleta.
ListaAtletasOrganizacion()	Lista todas las organizaciones a las que pertenece cada atleta.

Tabla 32: Descripción de la CCD_ControladoraAtleta.

Atributos	**Tipo**
lista_pg	Lista
Métodos	**Descripción**
GetListaPG()	Devuelve una lista con todos los planes

	gráficos.
SetListaPG($plista_pg)	Permite modificar una lista con todos los planes gráficos.
BuscarMacro($pg, $num)	Busca un macrosistema dado el nombre del plan gráfico al que pertenece y el número del microsistema.
BuscarPG($pg)	Busca un plan gráfico dado el nombre.
BuscarMeso($pg, $num, $meso)	Busca un mesosistema dado el nombre del plan gráfico al que pertenece, el número del microsistema y el nombre del mesosistema.
BuscarMicro($pg, $num, $meso, $micro)	Busca un microsistema dado el nombre del plan gráfico al que pertenece, el número del macrosistema, el nombre del mesosistema y el número del microsistema.
BuscarTP($pg,$num,$meso)	Dado un plan gráfico, el número de un macrosistema y el nombre de un mesosistema determinado, busca todos los test pedagógicos que se planificaron.
ListaMeso($pg,$num)	Dado un plan gráfico y el número de un macrosistema determinado, muestra una lista con todos los mesosistemas que contiene.
ListaMacro($pg)	Dado un plan gráfico determinado, muestra una lista con todos los macrosistemas que contiene.
ListaMicro($pg,$num,$meso)	Dado un plan gráfico, el número de un macrosistema y el nombre de un mesosistema determinado, muestra una lista con todos los microsistemas que contiene.
AdicionarMacro($pg, $num, $fi1, $fi2, $fi3, $fi4, $fi5, $ff1, $ff2, $ff3, $ff4, $ff5, $hs1, $hs2, $hs3, $hs4, $hs5)	Adiciona un nuevo macrosistema.
ModificarMacro($pg, $num, $fi1,	Permite modificar los datos de un microsistema.

$fi2, $fi3, $fi4, $fi5, $ff1, $ff2, $ff3, $ff4, $ff5, $hs1, $hs2, $hs3, $hs4, $hs5)	
CantMicro($pg, $num, $meso)	Dado el nombre de un plan gráfico, el número de un macrosistema y el nombre de un mesosistema, devuelve la cantidad de microsistemas que tiene ese mesosistema.
FechaInicio($pg, $num, $meso)	Devuelve la fecha de inicio de un mesosistema dado.
FechaFin($pg, $num, $meso)	Devuelve la fecha de fin de un mesosistema dado.
RTP1($pg, $num, $meso)	Devuelve el rango de tolerancia de peso mínimo de un mesosistema dado.
RTP2($pg, $num, $meso)	Devuelve el rango de tolerancia de peso máximo de un mesosistema dado.
Vol1($pg, $num, $meso)	Devuelve el volumen mínimo de un mesosistema dado.
Vol2($pg, $num, $meso)	Devuelve el volumen máximo de un mesosistema dado.
Int1($pg, $num, $meso)	Devuelve la intensidad mínima de un mesosistema dado.
Int2($pg, $num, $meso)	Devuelve la intensidad máxima de un mesosistema dado.
Combate($pg, $num, $meso)	Devuelve la cantidad de combates que se van a realizar en un mesosistema dado el nombre del plan gráfico al que pertenece, el número del macrosistema y el nombre del mesosistema deseado.
ValidarFecha($fi1, $fi2, $fi3, $fi4, $fi5, $ff1, $ff2, $ff3, $ff4, $ff5)	Permite validar las fechas de inicio y fin de todos los mesosistemas, para que no se vaya a introducir en el sistema una fecha incorrecta.
ValidarAno($fi, $ff, $ano)	Valida el año de un plan gráfico.

ValidarFechaAux($fecha1,$fecha2)	Valida que una fecha sea mayor que la otra.
ValidarFechaAux1($fecha1,$fecha2)	Valida que un microsistema no tenga menos de 7 días.
ValidarMacroFecha($fecha_inicio)	Valida la fecha de inicio de un macrosistema.
ConvertirFecha($fecha)	Convierte una fecha de un formato a otro. Ejemplo: (10-05-2009) a (10/05/2009).
Sesiones($pg, $num, $meso)	Permite conocer la cantidad de sesiones que va a tener un mesosistema.
Horas($pg, $num, $meso)	Permite conocer la cantidad de horas que va a tener un mesosistema.
Minutos($pg, $num, $meso)	Permite conocer la cantidad de minutos que va a tener un mesosistema.
GuardarMeso($meso, $pg, $num, $comb, $rtp1, $rtp2, $v1, $v2, $i1, $i2)	Permite guardar un mesosistema con su nombre, el plan gráfico al que pertenece, el número del macrosistema al que pertenece, cantidad de combates a realizar, el rango de tolerancia de peso mínimo y máximo, volumen mínimo y máximo, y la intensidad mínima y máxima de cada mesosistema.
GuardarMicro($micro, $pg, $num, $meso, $tp, $pm, $c, $ce)	Permite guardar un microsistema con su número, el plan gráfico, el número del macrosistema y el nombre del mesosistema al que pertenece, además de la planificación de las pruebas médicas, test pedagógicos, competencias y campos de entrenamiento.
GuardarMicro1($micro, $pg, $num, $meso, $lmv, $mjs)	Permite guardar la planificación de los ejercicios que se van a realizar diariamente.
LMV($micro, $pg, $num, $meso)	Dado el nombre de un plan gráfico, el número de un macrosistema y el nombre del mesosistema deseado, devuelve las actividades planificadas los días lunes, miércoles y viernes.
MJS($micro, $pg, $num, $meso)	Dado el nombre de un plan gráfico, el número

	de un macrosistema y el nombre del mesosistema deseado, devuelve las actividades planificadas los días martes, jueves y sábado.
InsertarPG($nombre, $cf, $l, $p)	Permite insertar un nuevo plan gráfico, con su nombre, competencia fundamental, lugar y pronóstico.
ModificarPG($nombre, $cf, $l, $p, $r)	Permite modificar el nombre de un plan gráfico, competencia fundamental, lugar y pronóstico.
PGModificar($pg)	Permite devolver los datos de un plan gráfico.
EncriptarURL($valor)	Encripta con MD5 el valor pasado por parámetro.
Encripta($cadena)	Encripta el valor pasado por parámetro.
DesEncripta($cadena)	DesEncripta el valor pasado por parámetro.
BuscarFechaMeso($f)	Devuelve la fecha de un mesosistema determinado.

Tabla 33: Descripción de la CCD_PlanGrafico.

Atributos	**Tipo**
listaTestPed	Lista
Métodos	**Descripción**
GetListaTestPed()	Devuelve una lista con todos los test pedagógicos.
SetListaTestPed($plistaTestPed)	Permite modificar una lista con todos los test pedagógicos.
ListaAtletasActivos()	Lista todos los atletas que se encuentran activos, o sea todos los atletas que no son baja.
ListaAtletas()	Muestra una lista de todos los atletas que existen en el sistema.
ListaAtletasTP($pg,$macro,$tp)	Lista todos los atletas que estén dentro de un plan gráfico, macrosistema y test pedagógico específico.
ListaPG()	Lista todos los planes gráficos.

ListaTestPG($pg,$macro)	Permite conocer la cantidad de test pedagógicos planificados en un macrosistema de un plan gráfico determinado.
ListaParametros()	Muestra una lista con los resultados de todos los parámetros de un test pedagógico.
ListaRangos()	Devuelve una lista de rangos.
MostrarParametro_Cantidad($ci,$parametro,$cantidad)	Permite mostrar un parametro_cantidad.
AdicionarParametro_Cantidad($ci,$pa,$cantidad,$tp,$pg,$ma)	Permite adicionar un parametro_cantidad.
AdicionarTP($tp,$pg,$ma,$fi,$ff)	Permite adicionar un nuevo test pedagógico.
BuscarMacroNTP($macro)	Permite buscar un macrosistema donde no se haya hecho test pedagógico.
BuscarTPPlanificado($pg,$macro)	Permite conocer la cantidad de test pedagógicos planificados en un macrosistema de un plan gráfico determinado.
BuscarTPHecho($pg,$macro)	Devuelve el número de test pedagógicos hechos hasta el momento en un plan gráfico y un macrosistema determinado.
PGActual()	Devuelve el plan gráfico con que se esta trabajando en ese momento.
FechaInicioTP($pg,$macro)	Muestra la fecha de inicio del test pedagógico a realizar.
FechaFinTP($pg,$macro)	Muestra la fecha de fin del test pedagógico a realizar.
FechaInicioTPM($pg,$macro,$tp)	Muestra la fecha de fin de un test pedagógico determinado.
FechaFinTPM($pg,$macro,$tp)	Muestra la fecha de inicio de un test pedagógico determinado.
ConvertirFecha($fecha)	Cambia el formato de una fecha determinada. Ejemplo: (10-05-2009) a (10/05/2009).

CantidadTP($pg,$macro,$tp,$ci,$pa)	Devuelve la cantidad de ejercicios de un parámetro determinado hechos por un atleta en un test pedagógico.
CantidadFisica($pg,$macro,$tp,$ci)	Muestra la cantidad física de un test pedagógico.
CantidadTecnica($pg,$macro,$tp,$ci)	Muestra la cantidad técnica de un test pedagógico.
NivelTP($pg,$macro,$tp,$ci,$pa)	Muestra el nivel por ejercicios de un atleta en un test pedagógico determinado.
ExisteTP($pg,$macro,$tp)	Verifica si existe un test pedagógico determinado.
BuscarNombreMesoTP($pg, $macro, $tp)	Dado el nombre de un plan gráfico, el número de un macrosistema y un test pedagógico determinado, devuelve el nombre del mesosistema donde se encuentra planificado el test pedagógico.

Tabla 34: Descripción de la CCD _TestPedagogico.

Las clases lógicas de interfaz de usuario son páginas PHP puras que tienen la responsabilidad de permitir la captura y actualización de los datos de las interfaces de usuario.

Las clases lógicas de interfaz de usuario que se definen en el sistema son:

Clase	**Descripción**
SP_Autenticar_Usuario	Permite al usuario autenticarse para poder acceder al sistema.
SP_Atleta	Permite al entrenador insertar un nuevo atleta, consultar y modificar los datos de un atleta.
SP_Insertar_Plan_Grafico	Permite al entrenador insertar un nuevo plan gráfico.
SP_Consultar_Plan_Grafico	Permite al entrenador consultar los

	datos de un plan gráfico deseado, además de exportar los datos a PDF.
SP_Modificar_Plan_Grafico	Permite realizarle modificaciones a un plan gráfico determinado.
SP_Gestionar_Test_Pedagogico	Permite al entrenador insertar un nuevo test pedagógico, consultar y modificar los datos de un test pedagógico.
SP_Gestionar_Usuario	Permite al administrador del sistema insertar un nuevo usuario, consultar y modificar los datos de un usuario, así como eliminar la cuenta del usuario que desee.

Tabla 35: Descripción de las clases server page.

Las clases interfaz de usuario contienen atributos que permiten la petición de datos al sistema a través de los formularios que la componen.

Las clases interfaz de usuario que se definen en el sistema son:

Clases	
1. CP_Autenticar_Usuario	14. CP_CrearMacrosistema1
2. CP_MenuEntrenador	15. CP_CrearMacrosistema2
3. CP_InsertarAtleta	16. CP_CrearMacrosistema3
4. CP_ConsultarAtleta	17. CP_ConsultarPlanGrafico
5. CP_ModificarAtleta	18. CP_ConsultarPlanGrafico1
6. CP_InsertarTestPedagogico	19. CP_ConsultarPlanGrafico2
7. CP_ConsultarTestPedagogico	20. CP_ConsultarPlanGrafico3
8. CP_ModificarTestPedagogico	21. CP_ModificarPlanGrafico
9. CP_InsertarUsuario	22. CP_ModificarMacrosistema1
10. CP_ConsultarUsuario	23. CP_ModificarMacrosistema2
11. CP_ModificarUsuario	24. CP_ModificarMacrosistema3
12. CP_EliminarUsuario	25. CP_MenuAdministrador

13. CP_InsertarPlanGrafico	

Tabla 36: Clases client page.

3.4.4 Interfaz de la Aplicación

La página principal de la aplicación mostrará una descripción del entrenamiento y el formulario para loguearse y acceder al resto de los módulos. Se utilizarán los colores gris, blanco y azul, en tonos claros y fuertes.

Se utilizarán menús en la parte superior de la página, el tipo de letra será verdana 10 y 12.

Los formularios de entradas ocuparán el centro superior. Estos tendrán una breve explicación del objetivo del formulario y alguna especificación con respecto a las entradas. Los formularios con muchas entradas se organizarán y dividirán de acuerdo a lo que representa.

3.4.5 Tratamiento de Errores

El sistema también contará con tratamientos de errores que facilitarán su buen funcionamiento. Se trata de minimizar la inserción manual de datos, para evitar posibles errores y hacer válida la entrada de datos.

Se utilizará *JavaScript* para validar los datos que se inserten manualmente, y en caso de haber errores se mostrarán mensajes informando la situación. Los errores que puedan ser generados por la base de datos se tratarán para que puedan ser entendidos por el usuario.

3.4.6 Seguridad

La página de inicio del sistema podrá ser accedida por todos los usuarios, pues esta mostrará una breve información del sistema. El resto de las páginas sólo permitirán el acceso a los usuarios que estén autentificados y que además tengan asignado el rol para poder acceder a dicha página.

Otro de los mecanismos implementados es el uso del algoritmo MD5 para encriptar las contraseñas de los usuarios que se almacenan en la BD como una cadena de treinta y cinco caracteres, evitando la legibilidad y garantizando la privacidad.

Se propone realizar salvas de la información en otros discos duros, ya que es de vital importancia para los entrenadores. Puede existir la posibilidad de que se pierda la información por problemas de orden tecnológico como la contaminación por virus, los fallos en dispositivos del sistema o la durabilidad de los soportes electrónicos, por lo que se propone que el servidor se destine solamente a servir al sistema,

evitando el uso de otros servicios que incrementen en gran medida las posibilidades de que el sistema falle.

3.4.7 Diagramas de Clases del Diseño

Describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Un diagrama de clases del diseño contiene la siguiente información.

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos
- Información sobre los tipos de atributos.
- Navegabilidad
- Dependencias

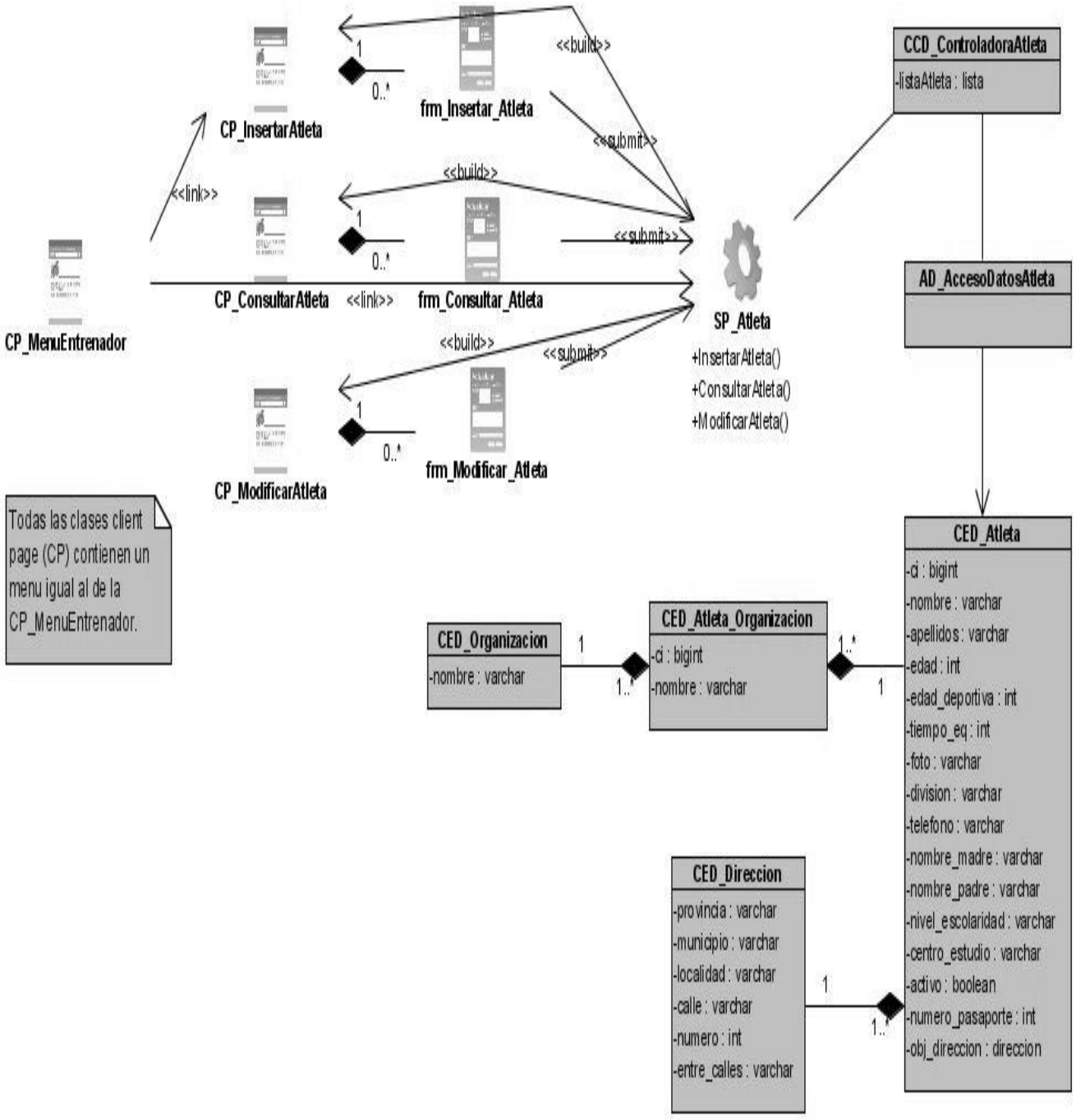


Figura 14: Diagrama del diseño caso de uso gestionar atleta.

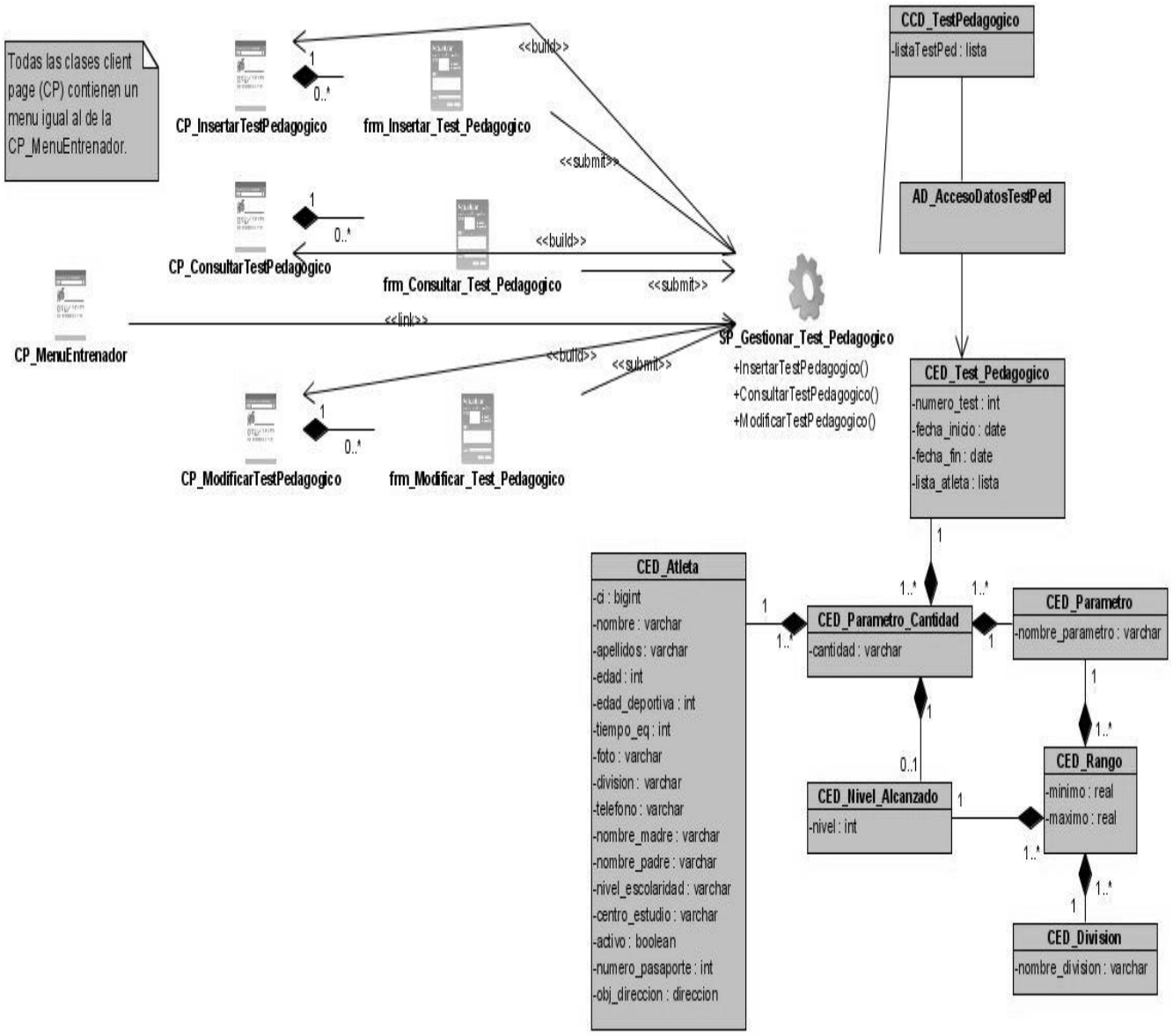


Figura 15: Diagrama del diseño caso de uso gestionar test pedagógico.

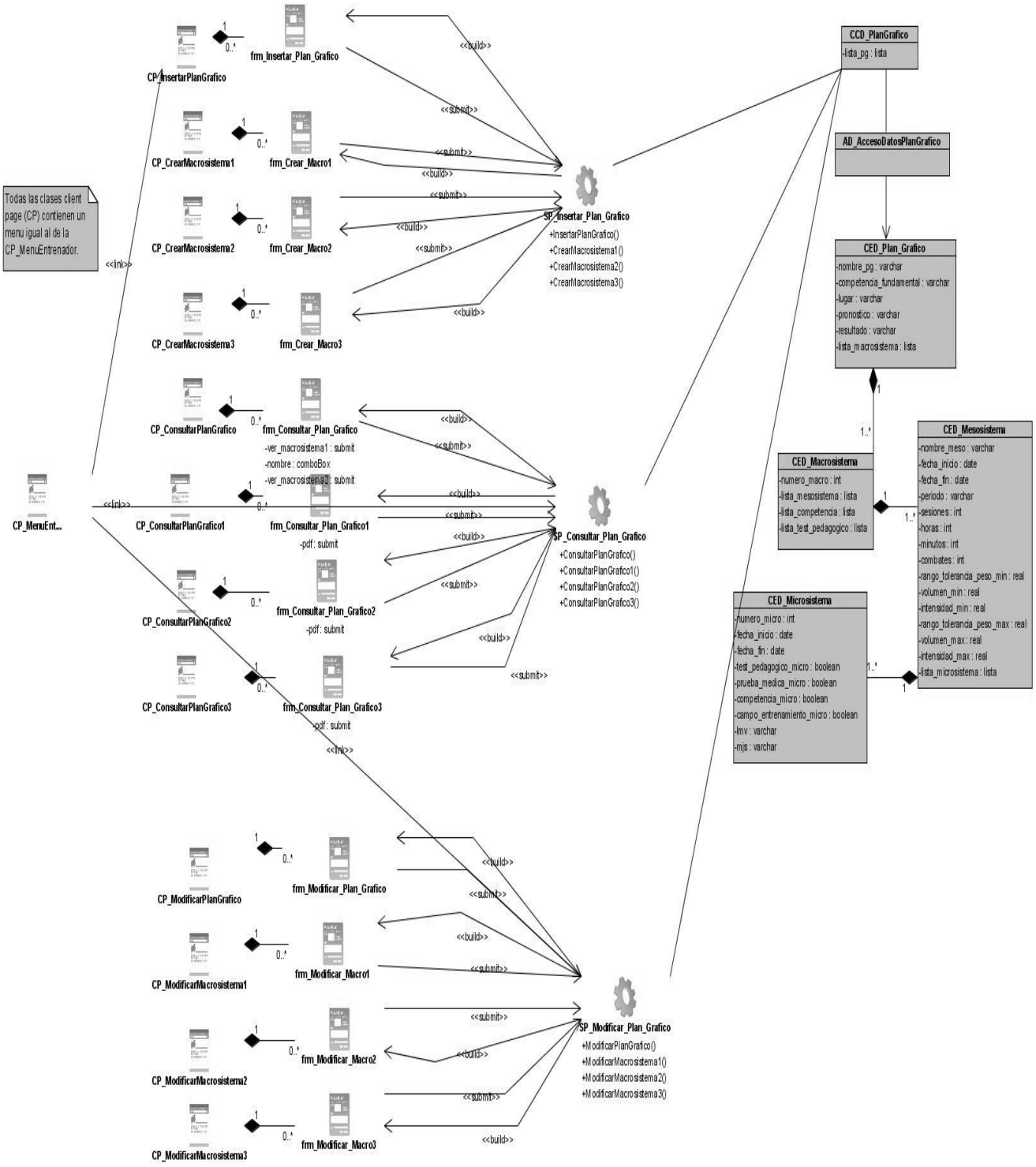


Figura 16: Diagrama del diseño caso de uso gestionar plan gráfico.

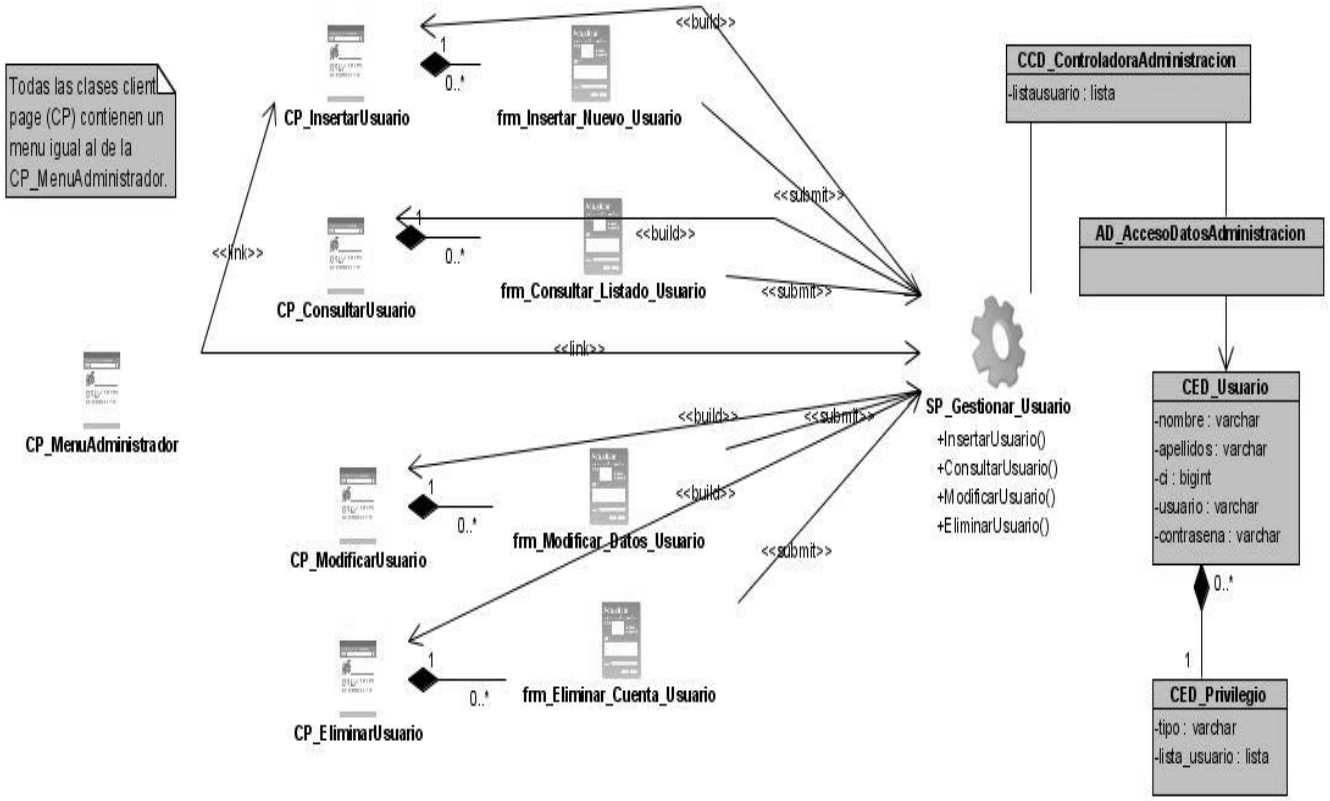


Figura 17: Diagrama del diseño caso de uso gestionar usuario.

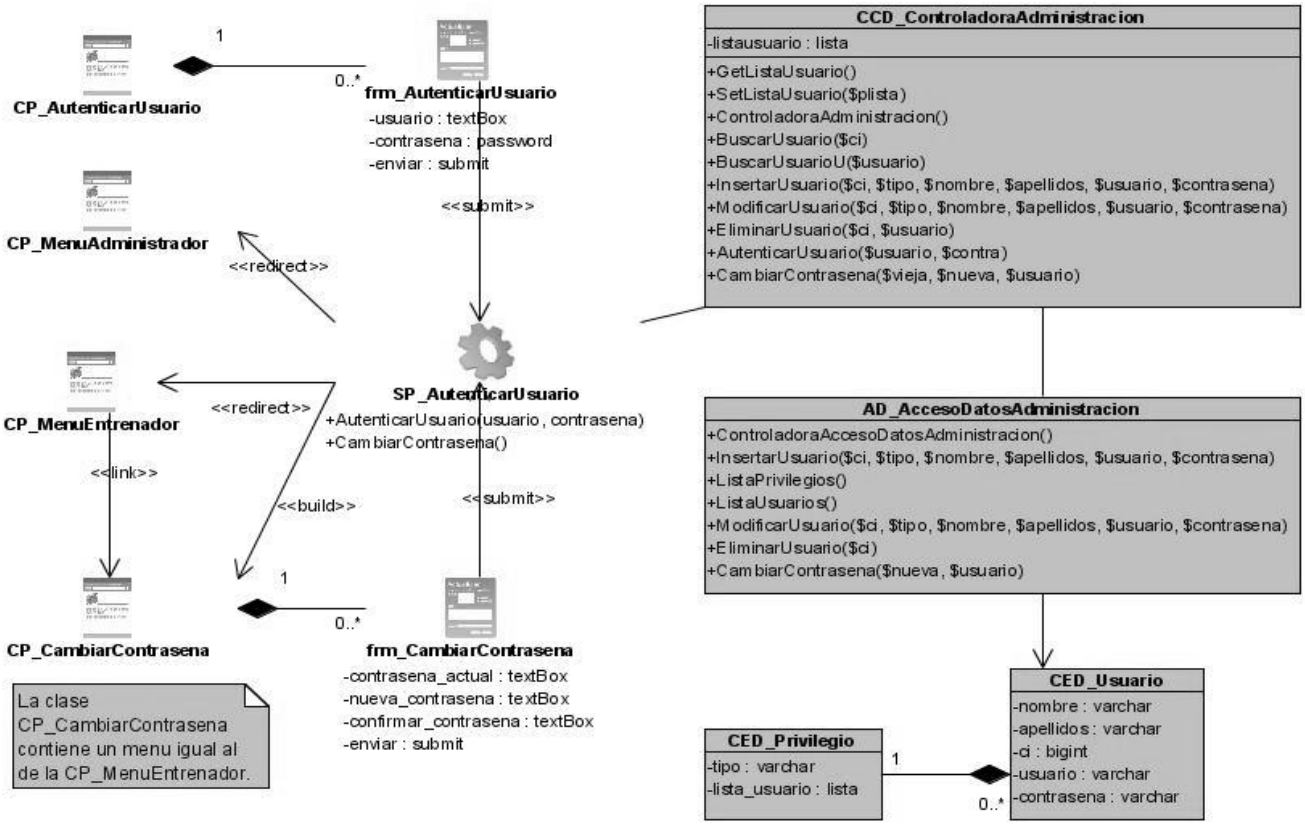


Figura 18: Diagrama del diseño caso de uso autenticar usuario.

3.4.8 Diagramas de interacción

Se utilizan para modelar los aspectos dinámicos de un sistema, también se pueden utilizar para visualizar, especificar, construir y documentar la dinámica de una sociedad particular de objetos, o para modelar un flujo de control particular de un caso de uso.

Se dividen en dos: diagramas de secuencia y de colaboración, a continuación se muestra la representación de algunos de los diagramas de colaboración.

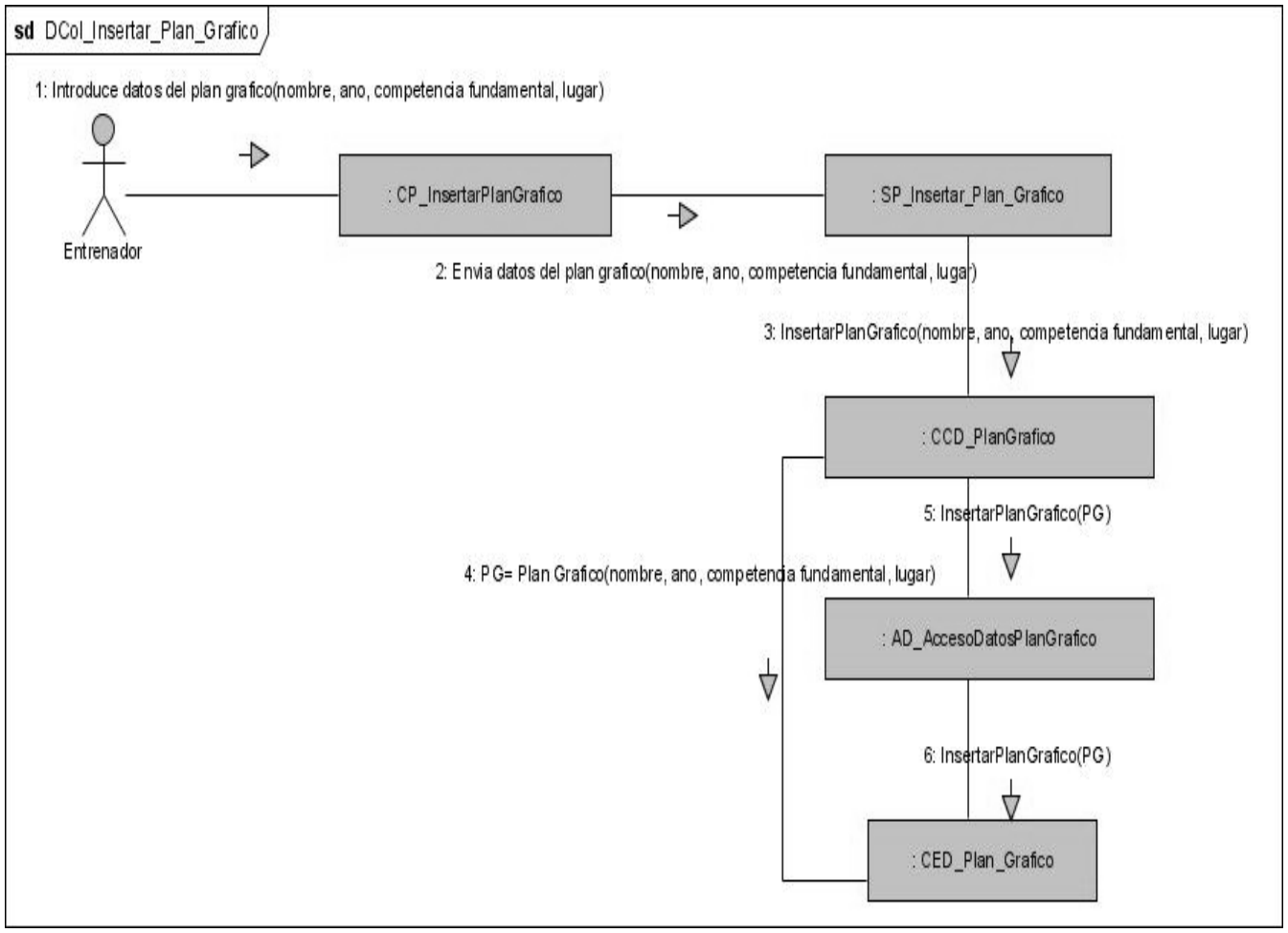


Figura 19: Diagrama de colaboración CU Gestionar plan gráfico: “Insertar_Plan_Gráfico”.

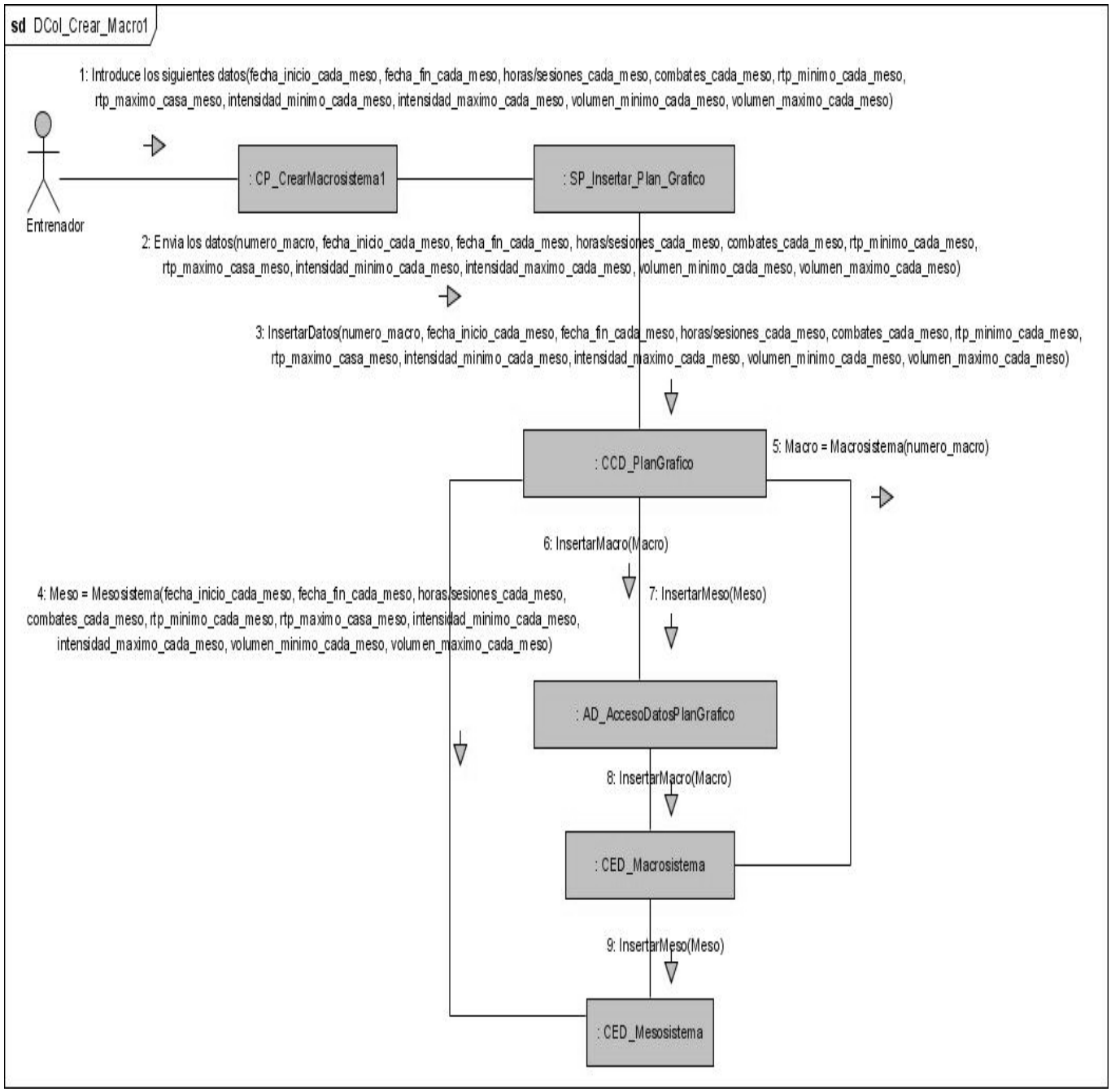


Figura 20: Diagrama de colaboración CU Gestionar plan gráfico: “Crear_Macro1”.

Figura 21: Diagrama de colaboración CU Gestionar plan gráfico: “Crear_Macro2”.

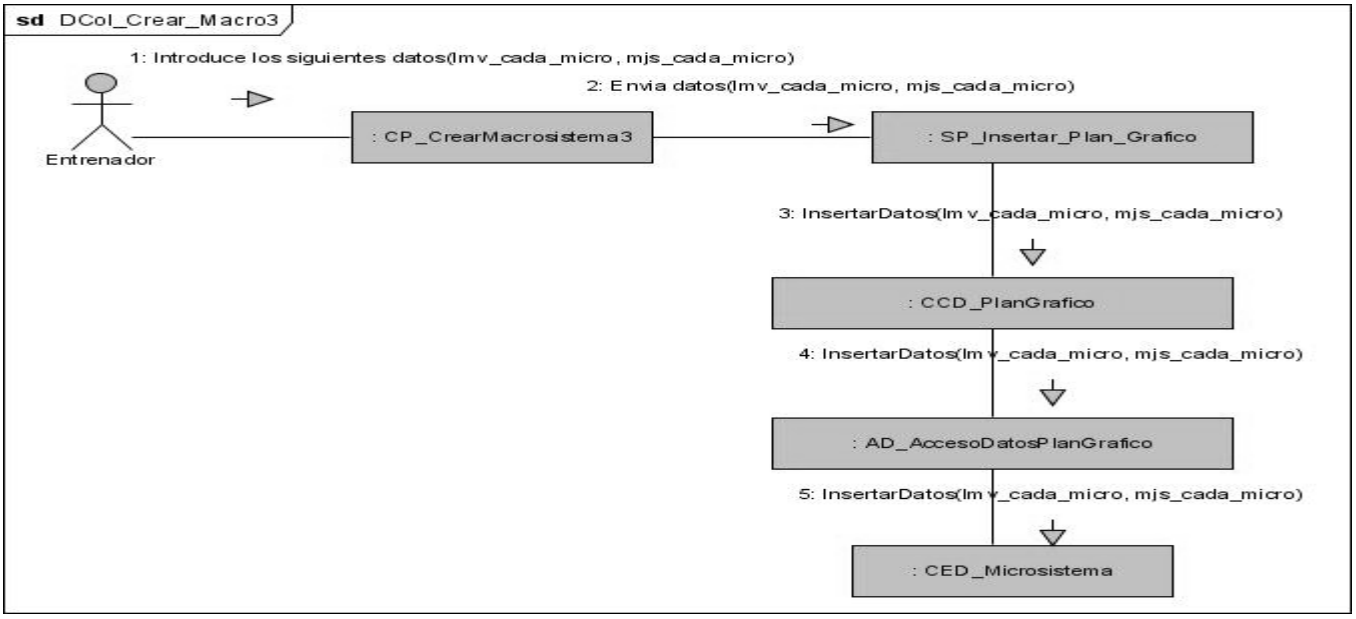


Figura 22: Diagrama de colaboración CU Gestionar plan gráfico: “Crear_Macro3”.

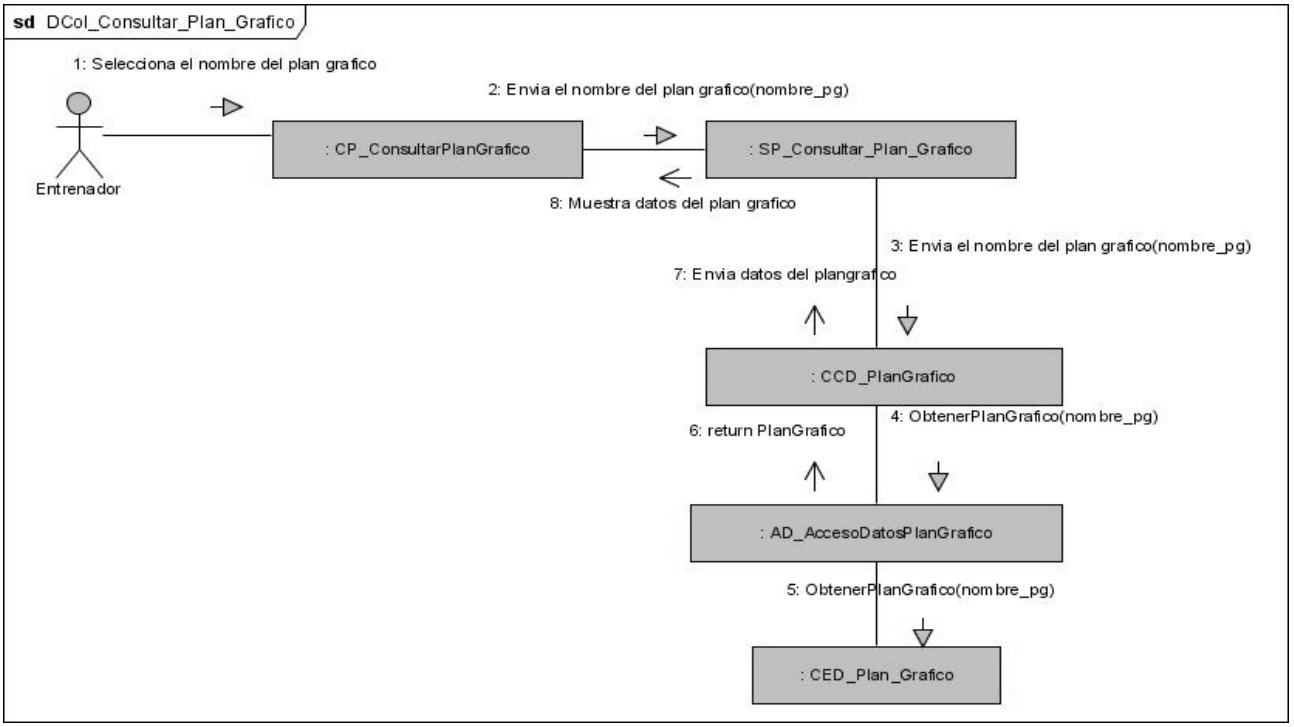


Figura 23: Diagrama de colaboración CU Gestionar plan gráfico: “Consultar_Plan_Gráfico”.

3.5 Conclusiones

En este capítulo se abordó el análisis y el diseño del sistema para la planificación del entrenamiento del judo femenino. Se desarrollaron los diagramas de clases del análisis y el modelo de diseño teniendo en cuenta el patrón de arquitectura 3 capas. Además se describió cada una de las clases del sistema, así como los principios del diseño utilizados. También se elaboraron los diagramas de interacción del sistema, específicamente los de colaboración.

Capítulo 4 Implementación del sistema

4.1 Introducción

En este capítulo se explicarán los temas relacionados con la implementación del sistema, a través del diagrama de despliegue y los diagramas de componentes.

4.2 Implementación

Un diagrama de implementación muestra las dependencias entre las partes de código del sistema (diagrama de componentes) y la estructura del sistema en ejecución (diagrama de despliegue).

4.2.1 Diagrama de despliegue

Los diagramas de despliegue se utilizan para modelar la vista de despliegue estática, estos muestran las relaciones físicas entre los componentes *hardware* y *s*oftware en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

El sistema se distribuirá en un servidor Web, el cliente podrá solicitar el servicio al servidor a través del protocolo HTTP, este servidor va a estar conectado al servidor de base de datos y la conexión entre ellos va a realizarse a través del protocolo ADO.

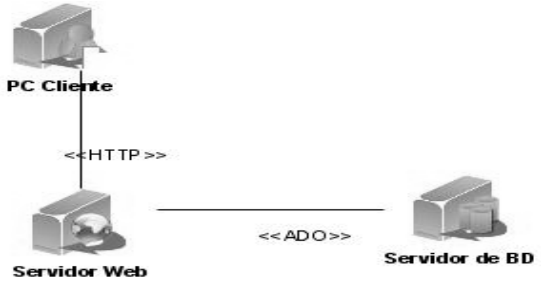


Figura 24: Diagrama de despliegue.

4.2.2 Diagrama de componentes

Los diagramas de componentes se utilizan para modelar la vista de implementación estática de un sistema, estos muestran las organizaciones y dependencias lógicas entre componentes software. Los elementos de modelado dentro de un diagrama de componentes son componentes y paquetes.

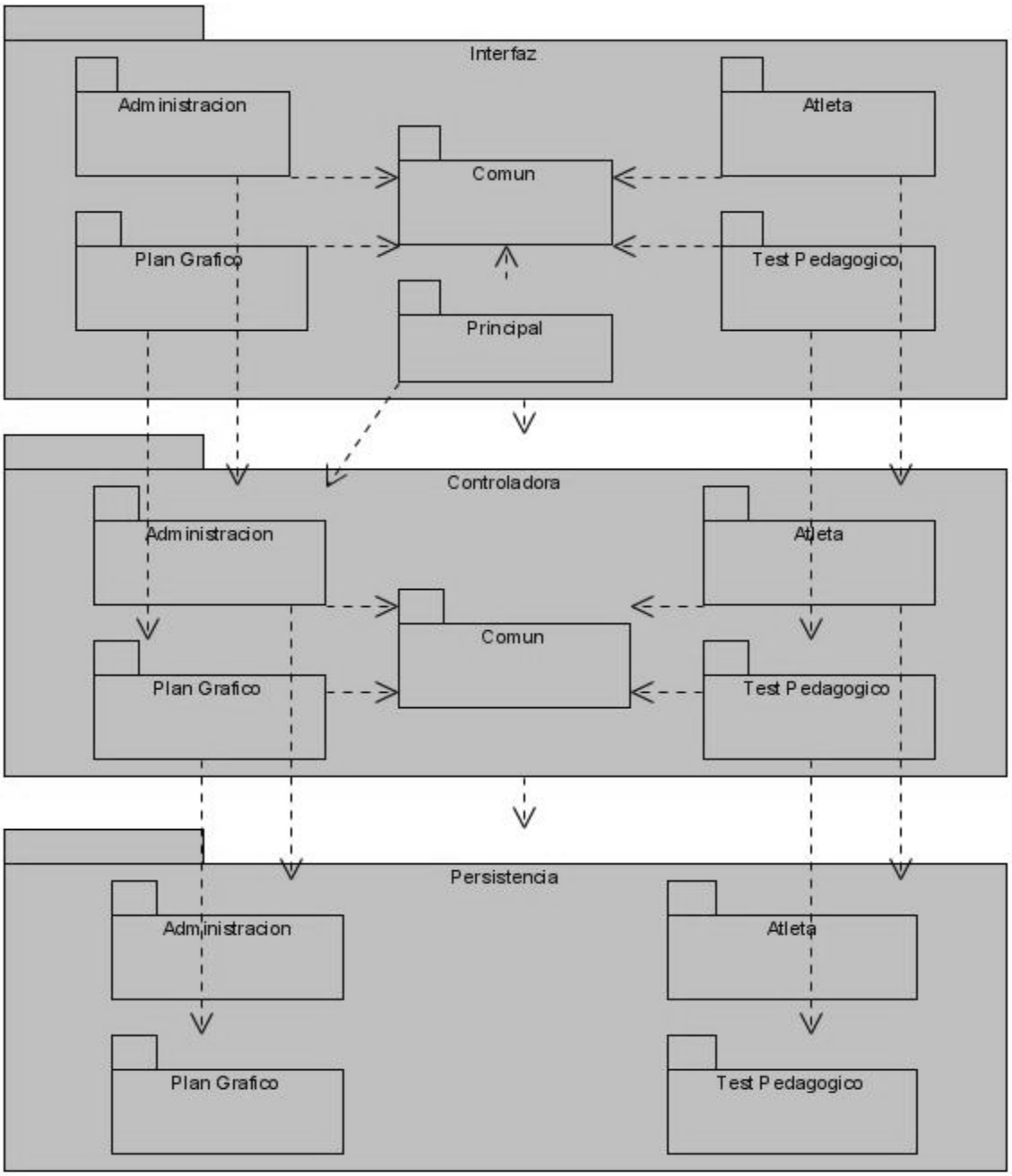


Figura 25: Diagrama de paquetes del sistema.

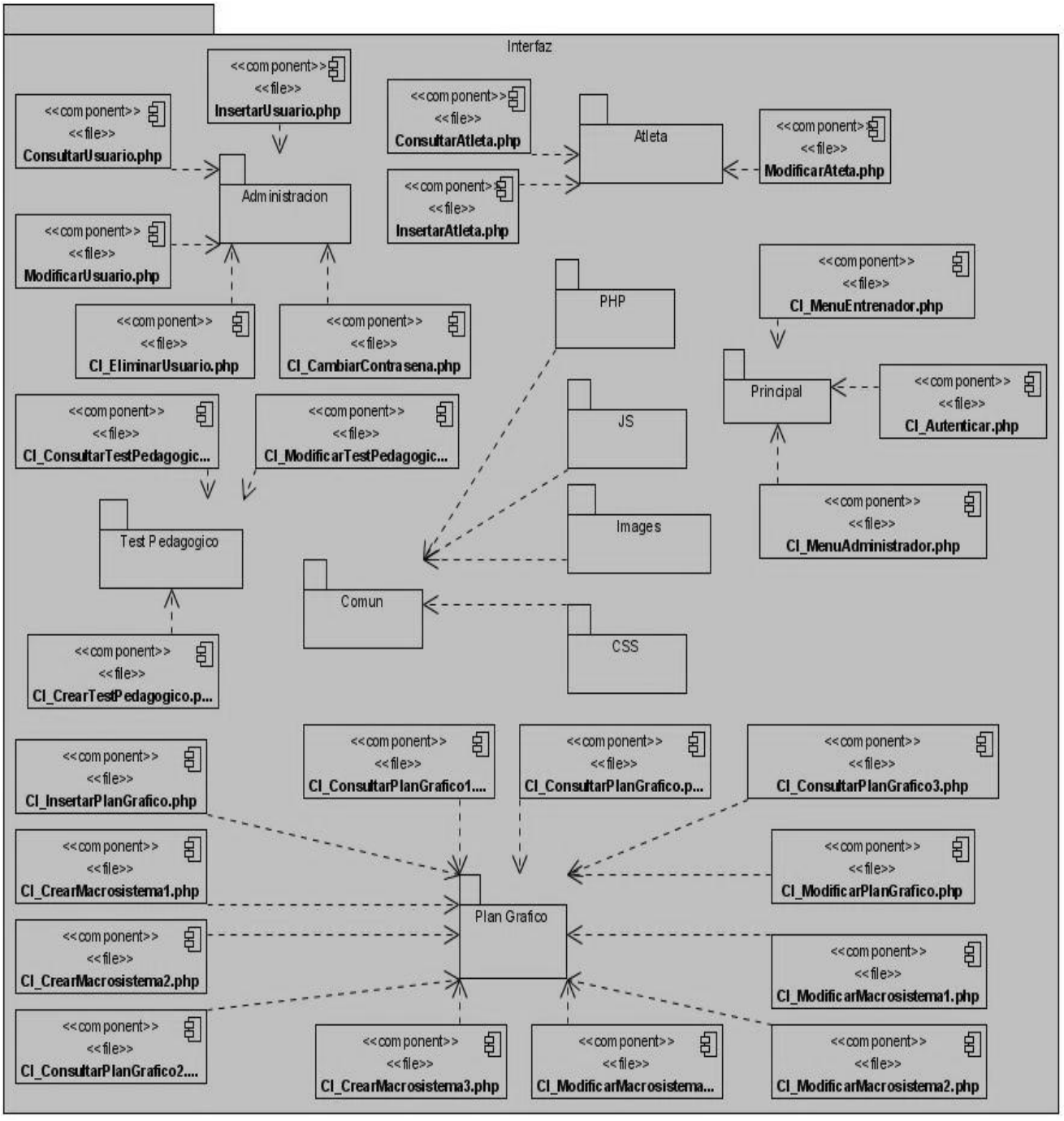


Figura 26: Diagrama de componentes del paquete interfaz.

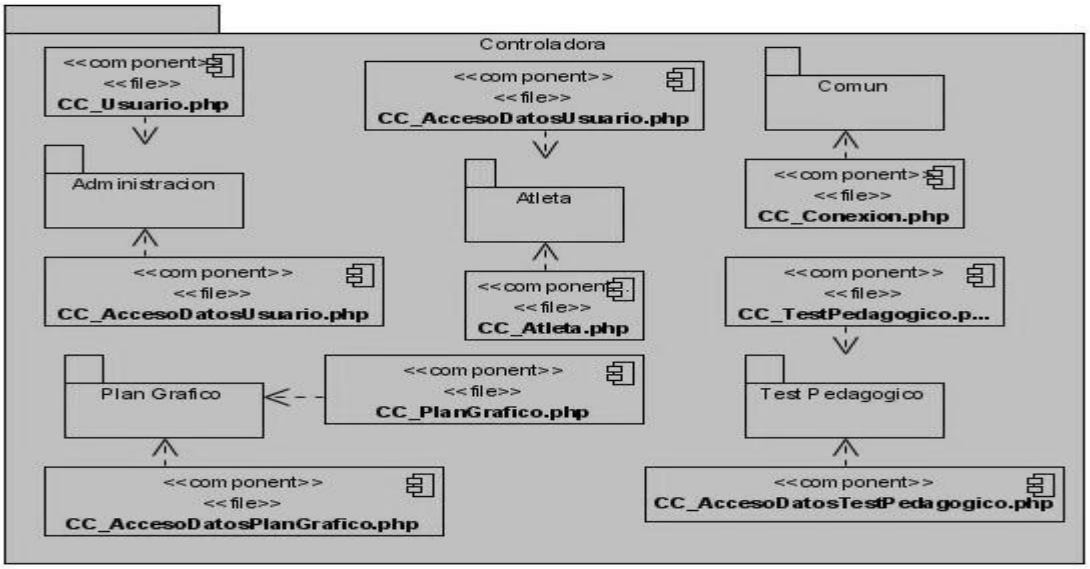


Figura 27: Diagrama de componentes del paquete controladora.

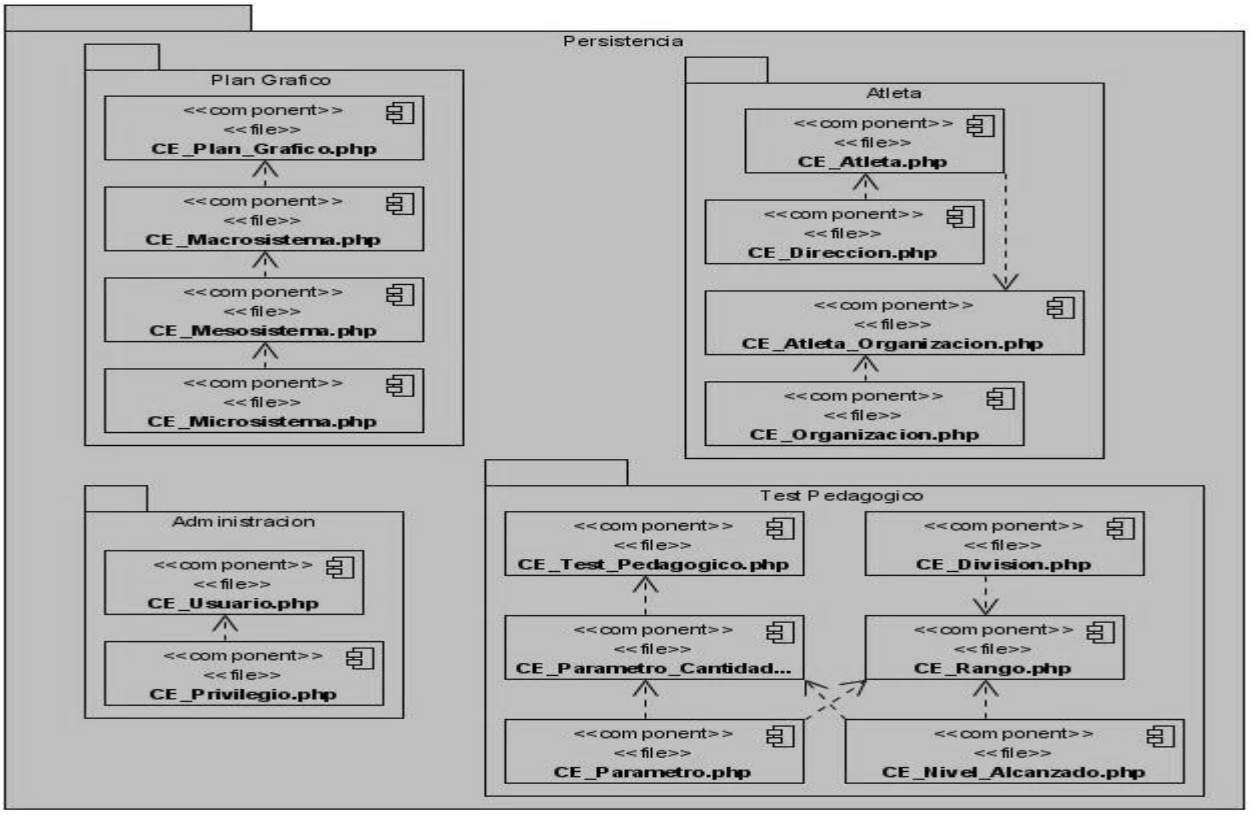


Figura 28: Diagrama de componentes del paquete persistencia.

4.3 Conclusiones

En este capítulo se presentaron los diagramas de implementación del sistema, ellos fueron: el diagrama de despliegue y los diagramas de componentes. Dentro de los diagramas de componentes, el del sistema y uno por cada paquete principal (interfaz, controladora y persistencia).

Conclusiones

Con la culminación de este trabajo se obtuvo como resultado, un análisis profundo de los sistemas de planificación de entrenamiento que existen actualmente tanto a nivel nacional como internacional, no solo para el deporte de judo, sino para diversos deportes, obteniéndose como resultado una gran cantidad de información de vital importancia para la realización del sistema. Además se seleccionaron las herramientas necesarias para realizar la aplicación con la calidad requerida, garantizando dentro de lo posible que fueran libres. De esta manera se logró obtener un sistema capaz de gestionar la planificación del entrenamiento de forma rápida y eficaz, logrando erradicar los problemas que existían anteriormente. Por todo lo antes mencionado se concluye que se ha cumplido con cada uno de los objetivos propuestos para la realización de esta aplicación.

Recomendaciones

Realizadas las conclusiones del trabajo, se recomienda:

- Ampliar las funcionalidades del sistema, de manera que sea capaz de mostrar a los usuarios diferentes tipos de reportes.
- Desarrollar la implementación de un módulo para la gestión de las competencias a las cuales se presentan las atletas tanto a nivel nacional como internacional.
- Desarrollar la implementación de un módulo para la gestión de las pruebas médicas, permitiendo registrar todas las pruebas que se les realizan a los atletas y el resultado de cada una de ellas.

Referencias Bibliográficas

Alvarez, M. A. (2002). "La tecnología Java para la creación de páginas Web con programación en el servidor." from http://www.desarrolloweb.com/articulos/831.php.

Alvarez, R. (2001). "Introducción al HTML." from http://www.desarrolloweb.com/articulos/534.php.

Carlos David Marrero, F. M., Henry rivero, Jasmin Sánchez Esculpi, Kiberley Kristal Santos Rosillo, Leslibeth Pessagno, Odalis Pereira. (2009). "MeRinde." from http://merinde.rinde.gob.ve/index.php?option=com_frontpage&Itemid=1.

Castillo, A. d. (2007). "Manual PHP. ¿Qué es PHP? una breve introducción." from http://www.lawebera.es/manuales/php/1.php.

Corsino, P. E. L. (2000). "Entrenamiento deportivo: Conceptos básicos para la planificación cíclica." from http://www.saludmed.com/CsEjerci/FisioEje/Entr-Cic.html.

Escribano, G. F. (2002). "Introducción a Extreme Programming." from http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.

Gallego, J. P. G. (2007). "Fundamentos de la Metodología RUP." from http://www.scribd.com/doc/297224/RUP.

Hinostroza, R. R. (2005). "LinuxCentro.net. Características de PHP." from http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

I. Jacobson, G. B. y. J. R. (2000a). "El Proceso Unificado de Desarrollo de Software." from http://bibliodoc.uci.cu/pdf/reg00060.pdf.

I. Jacobson, G. B. y. J. R. (2000b). "El Lenguaje Unificado de Modelado. Manual de Referencia." from http://bibliodoc.uci.cu/pdf/reg03050.pdf.

Lafuente, G. J. (2000). "GIDISWEB. UML Unified Modeling Languaje." from http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html.

Manso, J. M. G. (2009). "Planificación del Entrenamiento Deportivo." from http://www.sobreentrenamiento.com/ShopCE/Producto.asp?idp=691.

Marañón, G. A. (1999). "Qué es Java. Características del lenguaje Java." from http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.

Mellon, I. d. l. U. C. (2009). "Itera: Ingeniería de Software. RUP." from http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42.

Miguel Ángel Manzanedo del Campo, F. J. G. P. (1999). "Guía Rápida de Aprendizaje del Lenguaje Java." from http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto.Oct98/index.htm.

Navarro, D. N. M. S. d. C. y. D. M. M. M. (2005). "El Método Científico." from http://www.revistaciencias.com/publicaciones/EEFklVpyFkshdXimoL.php.

Ortega, F. O. G. (2005). "Automatización de la planificación del entrenamiento deportivo en diferentes deportes."

Palomeque, P. H. (2009). "Informática & Deportes." from http://www.entrenar.com.ar/contacto.php?action=consulta&ref=.

Sariola, J. A. M. (2008). "Espacio virtual para profesores y entrenadores de judo." from http://www.mirallas.org/.

Seco, I. (2000). "Elatleta.com." from http://www.elatleta.com.

Valdelli, I. (2006). "HTMLPOINT.com. CURSO JAVASCRIPT." from http://www.htmlpoint.com/javascript/corso/index.html.

Bibliografía

Alejandro Martínez, R. M. (2003). "Guía a Rational Unified Process." from http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf.

Artola, L. (2008). "Programania. Metodologías ágiles." from http://www.programania.net/php/comparativa-entre-zend-studio-y-eclipse-pdt/.

Benavidez, C. (2002). "Hojas de Estilo en Cascada." from http://www.sidar.org/recur/desdi/mcss/index.php.

Camilo. (2007). "WEBMASTERS EN LINEA." from http://www.webmastersenlinea.net/.

Carlos David Marrero, F. M., Henry rivero, Jasmin Sánchez Esculpi, Kiberley Kristal Santos Rosillo, Leslibeth Pessagno, Odalis Pereira. (2009). "MeRinde." from http://merinde.rinde.gob.ve/index.php?option=com_frontpage&Itemid=1.

González, L. (2007). "Proyecto EATS. Lenguajes del lado del servidor y del lado del cliente." from http://eats.wordpress.com/2007/01/17/lenguajes-del-lado-servidor-y-del-lado-cliente/.

Gracia, J. (2004). "WebEstilo. Manual de PHP." from http://www.webestilo.com/php/.

Gracia, J. (2005). "IngenieroSoftware. Análisis y Diseño. UML." from http://www.ingenierosoftware.com/analisisydiseno/uml.php.

Gregorio Robles, J. F. (2002). "Programación Extrema y Software Libre." from http://es.tldp.org/Presentaciones/200211hispalinux/ferrer/robles-ferrer-ponencia-hispalinux-2002.html.

León, E. (2008). "Blog de Eduardo León. Modelando con barro y unos cuantos bits." from http://slion2000.blogspot.com/2007/04/visual-paradigm-una-herramienta-de-lo.html.

Malpeceres, A. (2002). "Procesos de desarrollo: RUP, XP y FDD." from http://www.willydev.net/descargas/Articulos/General/cualxpfddrup.PDF.

Mehdi Achour, F. B., Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, Jakub Vrana. (2008). "PHP. Manual de PHP." from http://cl2.php.net/manual/es/index.php.

Méndez, N. D. D. (2005). "UNivirtual." from http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap11-2.html.

Miguel Ángel Manzanedo del Campo, F. J. G. P. (1999). "Guía Rápida de Aprendizaje del Lenguaje Java." from http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto.Oct98/index.htm.

Montalvo, J. C. (2008). "Calin Soft. Aplicaciones web ventajas y desventajas." from http://www.calinsoft.com/2008/08/aplicaciones-web-ventajas-y-desventajas.html.

Moyano, M. A. (2001). "ShopCe: Shopping de Productos sobre Ciencias del Ejercicio." from http://www.sobreentrenamiento.com.

Solís, M. C. (2003). "Una explicación de la programación extrema (XP)." from http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf.

Glosario de términos

Plan de Entrenamiento: Proceso organizado y sistemático encargado de ordenar e integrar el contenido del entrenamiento deportivo y de todas las medidas necesarias y medios disponibles que conducen a la realización efectiva de un entrenamiento y al desarrollo óptimo del rendimiento deportivo.

Test pedagógico: Es la prueba que se le realiza a las atletas para conocer su condición física y el nivel de preparación que tienen.

Plan gráfico: proceso que contiene de forma organizada toda la planificación de las actividades que se le van a realizar a los atletas, y el volumen e intensidad con que se van a aplicar para que alcancen la forma deportiva antes de presentarse a una competencia fundamental.

Mesosistemas: Representan etapas del proceso global de entrenamiento. Están orientados a lograr el desarrollo de una cualidad u objetivo parcial de todo el proceso. Su organización interna se realiza a base de microsistemas.

MPFG: Mesosistema de preparación física general.

MPFEV: Mesosistema de preparación física especial variada.

MPFE: Mesosistema de preparación física especial.

MOFD: Mesosistema de obtención de la forma deportiva.

MEFD: Mesosistema de estabilización de la forma deportiva.

Microsistema: Semana.

INDER: Instituto Nacional de Deporte, Educación Física y Recreación.

Macrosistema: Engloba el total de objetivos marcados en un proceso completo de entrenamiento.

Pron: Levantamiento de pesa acostado en un banco.

Halon: Ejercicio que se realiza agarrando una liga o extensor por los extremos con una o con ambas manos, dando comienzo de esta manera a la acción de extensión, a la vez que el atleta torcerá el tronco en la misma dirección del brazo o los brazos que se extienden.

Proyección: Aplicación de una técnica de combate.

Ippon: Se concede cuando la técnica es aplicada de forma correcta, o como resultado de proyectar al oponente de modo que al caer dé con la espalda por completo en el tatami.

División: Se le asigna a los atletas en dependencia de su peso corporal, las divisiones que existen en el judo femenino son 48kg, 52kg, 57kg, 63kg, 70kg, 78kg, +78kg.

Activo: Indica si un atleta esta retirado o no.

Plugin: Programa que proporciona alguna funcionalidad específica a otra aplicación mayor o más compleja.

Http (Protocolo de transferencia de hipertexto): Método mediante el que se transfieren documentos desde el sistema host o servidor a los navegadores y usuarios individuales.

ADO (ActiveX Data Objects): Es uno de los mecanismos que usan los programas de computadoras para comunicarse con las bases de datos, darles órdenes y obtener resultados de ellas.

Sesiones: La sesión como unidad básica del proceso de entrenamiento, está formada por ejercicios destinados al desarrollo y mejora de una o varias cualidades; estando determinadas por un número, orientación y distribución de los ejercicios.

QT o GTK+: Son librerías que se utilizan para hacer aplicaciones Web desktop.

DOM: Es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

Wizards: Programas que mediante pantallas interactivas facilitan y sirven de guía para el uso de otros programas o aplicaciones.

Debugger (en español depurador): Es un programa que permite depurar o limpiar los errores de otro programa informático.