Facultad 8

Universidad de las Ciencias Informáticas

Título: "Análisis, diseño e implementación del módulo de estudio técnico – táctico de los jugadores a la defensa en el béisbol"

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

Autor(es):

Yaniris Corbea Gómez

Darien Luciano Alba Ramírez

Tutora:

Ing. Nilet María Soto López

Ciudad de La Habana

19 Junio de 2009

Sé el cambio que quieres ver en el mundo…

Mahatma Gandhi

DECLARACIÓN DE AUTORÍA

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

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

Yaniris Corbea Gómez

_____

Darien Luciano Alba Ramírez

_____

Ing. Nilet María Soto López

_____

AGRADECIMIENTOS

Quiero agradecerle:

A mi papi y a mi mami por el apoyo y por confiar en mí.

A mis amistades por estar presente en todos los momentos.

A mis hermanos (Aniley y Juan José) por ser mi inspiración.

A mis Súper poderosas (Avila, López, Adriana, Lisandra y Yanet) por estar ahí cuando más lo necesité, por aguantar mis malcriadeces y por ser las personitas más bellas que he conocido.

A Darluin a pesar de todo, gracias por ayudarme y hacerme entender que uno puede luchar por lo que quiere aunque las personas queridas no estén a tu lado.

A mi primo Leonel por preocuparse por mi y enseñarme que a veces quieren más los de afuera que la propia familia. Gracias primo.

A nuestra tutora Nilet por su dedicación.

A Lannie por ayudarnos a realizar este sueño incondicionalmente.

A todas aquellas personas que de una forma u otra ayudaron a la realización de este trabajo.

A todos muchas gracias.

Yaniris Corbea Góme

AGRADECIMIENTOS

Agradezco la realización de este trabajo:

A nuestra tutora Nilet, por haber sido tan exigente y sincera con nosotros.

Al tribunal que nos ha evaluado a lo largo de este período.

A todos los profesores que han intervenido en mi formación y a todas las personas que de una forma u otra me apoyaron durante estos cinco años de estudios.

A todos muchas gracias.

Darien Luciano Alba Ramírez

DEDICATORIA

A mi abuelita aunque ya no esté, siempre me concedió el honor de decir: Mi abuela es la mejor del mundo. Gracias abue por todo el cariño que me diste nunca te voy a olvidar. Por eso y mucho más quisiera que donde estés, estuvieras orgullosa de mí.

A mi mami por ser la única, la comprensiva, porque sabe lo que pienso y cómo me siento. Porque adivina mis ideas, por su apoyo en todo momento, en esos momentos en que la vida nos ha puesto obstáculos tan grandes. Por llamarme siempre y estar pendiente de mis cosas, por la preocupación con respecto a mi salud y cuando la necesito en el momento que sea preciso.

A mi papi aunque sea súper peleón pero no se puede negar que es el hombre de mi vida y que lo quiero cantidad, gracias por confiar en mí y dejar volar mis sueños para lograr ser la mujer que soy hoy, y recuerda que siempre seré para ti "Tu Niñita".

A mi hermana a quien quiero cantidad y por ser el ejemplo más lindo que he tenido. Mi hermanita gracias por tu amor, por escucharme, apoyarme y por tus celos.

A mi hermanito (El médico) que lo adoro, herma aunque no lo creas siempre voy a estar ahí para cuando me necesites.

A mi sobrinito que aunque es muy intranquilo lo quiero con la vida y quisiera ser un ejemplo para él, para que algún día diga "Quiero ser como mi tía Yaniris".

A mis cuñados por ayudarme a realizar este lindo sueño.

A mis antiguas amistades, esas que han quedado después de mis primeros pasos en la vida, a ustedes les debo la comprensión de que la amistad sincera existe. ¡Gracias! A mis amistades de ahora: que luego de 5 años llegamos a unirnos como Las Súper Poderosas. A ustedes les debo muchos momentos de alegrías que jamás olvidaré. ¡Gracias por esas vacaciones maravillosas! A mi grupo de siempre, ese con el que comencé mis primeros años de Universidad. Desde el primer día me sentí satisfecha por compartir con ustedes las primeras experiencias, los primeros tropiezos aquí en la UCI.

Yaniris Corbea Gómez

DEDICATORIA

Dedico este trabajo a mi gran familia por el apoyo que me han dado y la confianza que siempre han tenido en mí.

A mi nueva familia, la que yo he formado, mi hijo Darien y mi esposa Nahilan para los cuales me he formado como profesional.

Pero especialmente me gustaría dedicarlo a tres personas. Mi abuela María García ya fallecida, a mi madre Maritza Ramírez y a mi padre Luciano Alba los cuales han sido los principales guías en mi formación, agradezco a ellos la educación que me han dado, y sé que hoy se sienten más contentos y orgullosos que yo mismo, por eso me gustaría dedicar especialmente este trabajo a ustedes.

Darien Luciano Alba Ramírez

RESUMEN

El presente trabajo está dirigido a la creación de un sistema encargado de controlar el rendimiento táctico-defensivo del conjunto de jugadores de béisbol que se ubican en la línea central de un juego; es decir aquellas posiciones que en su selección y evaluación priman en primer orden este parámetro, estas son la llamada línea central o columna vertebral del equipo (segunda base, torpedero, jardinero central y receptor).

El sistema brinda información complementaria de las características y acciones que se realizan por estas posiciones en cada juego; información necesaria para el perfeccionamiento de la planificación, organización, ejecución, control y evaluación del entrenamiento deportivo, lo que hace que los mismos sean más reales y con un carácter cada vez mas científico, así como permite una caracterización más aproximada a la realidad del jugador y del juego de Béisbol.

PALABRAS CLAVES: Sistema, Software, Béisbol, Diseño, Implementación.

Índice

Índice de Figuras

Introducción

Hoy en día el deporte moderno exige del desarrollo de la ciencia y la técnica, una exigencia que no debe ser al azar, sino encaminada a solucionar aquellos problemas que limiten el rendimiento y nivel del mismo; dichos problemas no se limitan en el deporte de alta competición, sino también en aquel que cumple la función de soporte de este. De estos problemas que están en el orden subjetivo y donde el hombre cumple un papel preponderante están la selección de talentos sobre la base de las características y cualidades de todos aquellos practicantes en cualesquiera de las modalidades y niveles de participación, otro problema lo constituye la preparación y el entrenamiento en todos sus sentidos, en la medida en que los mismos sean más reales, y en estrecha correspondencia con las características y exigencias de cada disciplina deportiva, mejores y mayores niveles de rendimiento y resultados, se podrán alcanzar a todos los niveles de participación deportiva.

El béisbol como disciplina deportiva, no está exento de tales situaciones. Con los cambios que ha experimentado el mismo, en cuanto a las características de los jugadores que en los últimos años han participado en torneos internacionales, se hace necesario buscar fórmulas, medios y métodos que permitan dar soluciones o al menos disminuir los efectos que traen consigo los problemas antes mencionados de forma tal que eleve el nivel de nuestros jugadores, en todas las categorías, para de esta forma continuar en la élite Mundial, y buscar con el apoyo de la ciencia y la técnica, un camino o medio para solucionar aquellos aspectos que aún tenemos en nuestro deporte que constituyen limitantes en nuestro desarrollo.

Situación Problémica

La práctica del Béisbol arroja una rica y abundante estadística que permite evaluar el rendimiento de los jugadores en cada una de las áreas de juego, es decir la ofensiva, la defensiva y el picheo. Estas estadísticas son recogidas manualmente y archivadas en cada juego efectuado, ya que nuestro país no cuenta con un sistema de seguimiento para la defensa en el béisbol, lo que hace más difícil controlar el rendimiento defensivo que ayude a calcular al menos aquellas posiciones que son necesariamente más defensiva dentro del juego, es decir aquellas posiciones que en su selección y evaluación priman en primer orden este parámetro, estas son la llamada línea central o columna vertebral del equipo (segunda base, torpedero, jardinero central y receptor).

Por tanto, el ***problema a resolver*** sería: ¿Cómo facilitar el control del rendimiento táctico-defensivo de la línea central en el béisbol cubano?

El **objeto de estudio** lo constituye el control de información por parte de la Federación Cubana de Béisbol Aficionado.

Como **objetivo general**, desarrollar un sistema que sea capaz de facilitar el control del rendimiento táctico-defensivo en la línea central del béisbol cubano.

Como ***objetivos específicos*****:**

- Investigar sobre otros Sistemas semejantes existentes en Cuba y en el mundo.
- Realizar Análisis, Diseño e Implementación del Sistema de Estudio técnico-táctico de los jugadores a la Defensa en el Béisbol.
- Describir los Lenguajes de Programación y Modelado a utilizar en el desarrollo del sistema propuesto.
- Dar solución al problema, basado en la implementación de las funcionalidades del sistema.
- Obtener resultados satisfactorios en las pruebas realizadas al culminar el proceso de implementación.
- Dejar el Módulo listo para su despliegue y utilización en la Serie Nacional del año 2009 - 2010.

El ***campo de acción*** se encuentra enmarcado en el control de estadísticas de los juegos de béisbol de la Federación Cubana de Béisbol Aficionado.

La ***idea a defender*** en esta investigación es que si se logra el desarrollo de un sistema que permita controlar el rendimiento táctico-defensivo del conjunto de jugadores de béisbol que se ubican en la línea central de un juego, entonces se lograría un control más eficiente de las acciones defensivas que pueden usarse para formar mejores atletas.

Para dar cumplimiento a lo antes planteado se definieron las siguientes ***tareas*****:**

- Búsqueda bibliográfica de los antecedentes históricos del tema.
- Sostener de forma frecuente reuniones y entrevistas con el personal a las cuales está destinado la aplicación.
- Seleccionar las herramientas y tecnologías adecuadas para el desarrollo de la aplicación.
- Estudio y selección de las metodologías de desarrollo, para seleccionar la más adecuada para llevar a cabo el desarrollo de la aplicación.
- Realizar el análisis y diseño de la aplicación.
- Realizar la implementación de la aplicación.

El presente documento se estructura en los siguientes capítulos:

Capítulo 1. Fundamentación Teórica: este capítulo abordará los principales conceptos utilizados en el contexto del problema, de la existencia de algunas aplicaciones que se utilizan para el desempeño de este deporte y se profundizará en la situación problémica existente del trabajo. Se analizarán las metodologías y herramientas adecuadas para la producción de este tipo de software y se fundamentará la selección de las mismas.

Capítulo 2. Características del Sistema: este capítulo expondrá el resultado de los flujos de trabajo de negocio y requerimiento, desarrollando un modelo de dominio para una mejor comprensión del negocio y finalmente se realizarán los casos de usos con sus descripciones previamente.

Capítulo 3. Análisis y Diseño: este capítulo expondrá el resultado de los flujos de trabajo de análisis y diseño, para lograr un mejor entendimiento y facilitar la comprensión de la aplicación que se quiere desarrollar. Se realizarán los diagramas pertinentes en cada uno de ellos.

Capítulo 4. Implementación: este capítulo expondrá el tema de la implementación, donde se muestran las dependencias entre las partes del código del sistema y la organización física de los elementos de implementación (diagrama de componentes).

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En este capítulo se exponen una serie de argumentos y explicaciones que permiten comprender de manera clara y concisa la situación real existente en el entorno beisbolero. Como también el concepto de palabras claves utilizadas y algunas aplicaciones usadas en el béisbol. Se realiza un análisis de cómo se hallan en el mundo las tecnologías que pueden ser adecuadas para la construcción de la aplicación que se va desarrollar, por lo que se realizan algunas comparaciones que explican el por qué de la propuesta final.

1.2 Aplicaciones existentes para el béisbol

Sistema para el Manejo de un Tablero Electrónico de Béisbol

La tarea más importante que este sistema realiza es: llevar de manera electrónica y con más facilidad las anotaciones y/o estadísticas tanto del partido como de cada jugador durante el encuentro. Algunos de los datos almacenados son: el número de carreras anotadas así como quienes las anotan, y al igual los hits, errores, ponches, entre otros.

Para almacenar los datos de cada jugador y de cada partido se diseñó una base de datos relacional. Por otro lado para poder automatizar las anotaciones en los partidos de béisbol se hizo un estudio de éstas, de tal forma que estas anotaciones se hicieran de una forma automática y fácil para la persona que manejará el sistema.

El sistema permite simultáneamente registrar anotaciones y controlar de forma automática el tablero electrónico de béisbol, que muestra el marcador de un partido. Dicho tablero se encuentra en el fondo del estadio, por lo que es controlado por el sistema de forma remota. Además en caso de que no se quiera manejar el tablero electrónico y sólo obtener los datos, se puede deshabilitar el control del tablero. Como se mencionó anteriormente el tablero electrónico está en el estadio y el equipo local puede ir a jugar a otros estadios, y aun así se puede utilizar el sistema para obtener los datos.

Los datos también pueden ser consultados desde una página Web, con la finalidad de que las personas puedan seguir el marcador del partido de béisbol desde otras partes. (1)

Sistema de Escauteo

Estudiantes de la Universidad de las Ciencias Informáticas (UCI) implementan un software para facilitar estadísticas a los equipos y directivos del béisbol cubano. Este sistema se comenzó a utilizar objetiva y prácticamente desde el torneo José Antonio Huelga del 2008, pero existe desde mucho antes, incluso en el I Clásico Mundial se hicieron experimentos de escauteo desde Cuba sobre otros equipos, aunque no fue con la profundidad que tiene ahora.

Proviene de una página web de béisbol que se desarrolla en la UCI con muchos de estos datos, con la que la Federación Cubana de Béisbol (FCBA) se mostró interesada. Entonces Noel Miño y José A. Cabrera comenzaron a organizar este software, al que luego se insertaron un grupo de estudiantes. Durante el juego se llevan las referencias de cada lanzamiento: el tipo, zona donde cae, velocidad, área de impacto, tipo de jugada (strike zona caliente (dónde es más o menos seguro cada lanzador o bateador), los promedios cantado o en swing), si el bateador conecta, hacia qué banda del terreno lo hizo, tipo de conexión (fly, línea, rolling), etc.

Al final de cada juego todos los encargados de este seguimiento envían los datos a un puesto de mando creado para procesarlos, y este envía un reporte final que incluye la efectividad de estos, dividido en cada una de las nueve regiones en que se fracciona la zona de strike y hacia que parte del terreno un jugador dirige más sus batazos. También se llevan los average contra los tipos de lanzamiento, como batea cada pelotero en las diferentes situaciones: con un out o más, bases limpias o llenas y hombres en posiciones anotadoras. (2)

1.3 Argumentación de la Situación Problémica

En la actualidad nuestro país no cuenta con un sistema de seguimiento para la defensa de la línea central en el béisbol. Estados Unidos (EE.UU) cuenta con sistemas que proporciona este tipo de seguimiento, pero lamentablemente debido al bloqueo que golpea a nuestro país es imposible utilizarlo.

Actualmente el proceso de control de datos de referencia y posterior cálculo estadístico de los juegos de béisbol se lleva a cabo de forma manual. En cada estadio se encuentran uno o varios anotadores oficiales del encuentro, los cuales son los responsables de plasmar en el papel este trabajo. El anotador tiene que dominar a la perfección el manual de anotaciones para poder desempeñar su labor correctamente. Tras finalizar el encuentro se hacen todos los cálculos estadísticos y los datos son archivados en lugares físicos ya que nuestro país no cuenta con una aplicación que pueda realizar esta dura tarea.

Con este trabajo se pretende realizar una aplicación capaz de almacenar y calcular automáticamente los datos de average, de manera que no sea necesario que el usuario, en este caso los coordinadores del torneo, tengan que dominar de memoria todos los datos que son necesarios anotar, ni realizar manualmente todos los cálculos para cada jugador o equipo.

Desde el curso 2007- 2008 en la Universidad de las Ciencias Informáticas (UCI) se desarrolla en conjunto con la FCBA una aplicación de scouting (seguimiento) que se aplica en la actual serie nacional, la misma ha tenido gran connotación y reconocimiento por las grandes ventajas que propicia para el desarrollo de mejores peloteros y selecciones. En esta aplicación durante el juego se llevan las referencias de cada lanzamiento: el tipo, zona donde cae, velocidad, área de impacto, tipo de jugada (strike cantado o en swing), si el bateador conecta, hacia qué banda del terreno lo hizo, tipo de conexión (fly, línea, rolling), etc.

Siguiendo esta línea de trabajo, se desean crear otras aplicaciones de este tipo que cubran el área de seguimiento defensivo para la línea central en el béisbol. Producto a las ventajas que propician las aplicaciones existentes de seguimiento y a la gran aceptación que ha tenido la que se aplica en la actual serie nacional, la FCBA desea poder ampliar hacia otros departamentos, otras aplicaciones que propicien el desarrollo y calidad para nuestro deporte nacional. La UCI actualmente trabaja en la construcción de otros módulos con el mismo propósito y destino que el que ya esta usándose.

1.4 Conceptos asociados al dominio del problema

Béisbol*:* Juego entre dos equipos, donde los jugadores han de recorrer ciertos puestos o bases de un circuito, en combinación con el lanzamiento de una pelota desde el centro de dicho circuito.

Sistema*:* Conjunto de elementos que se interrelacionan para funcionar como un todo. Los sistemas tienen tanto entradas como salidas, a las cuales llamamos interfaz. Además tiene una localización, la cual son todos aquellos lugares donde se realiza un proceso.

Federación Cubana de béisbol: Es la encargada de la organización de la serie nacional de béisbol (SNB). La que decide si se puede o no la realización de un juego en un estadio determinado.

Anotador Oficial: Es el encargado de observar el juego de béisbol para tomar las decisiones oficiales concernientes a la aplicación de las reglas de anotación, tales como si el avance de un bateador a primera base es consecuencia de un hit o un error.

Estadísticas*:* Es un registro acumulativo de todos los récords de bateo, fildeo, corrido de bases y récords de lanzadores, para cada jugador que aparezca en un juego de campeonato de la liga o de post temporada.

Escauteo*:* Sinónimo de seguimiento**.**

1.5 Metodología

Con el avance tecnológico, se han incrementado la cantidad de metodologías a tener en cuenta para el desarrollo del software. En la actualidad es imprescindible el uso de dichos procesos de desarrollo, pues permiten a los desarrolladores un buen ambiente de trabajo y evitan de cierto modo que el proyecto fracase a largo plazo. Con el uso de las metodologías de desarrollo, cualquier equipo puede elaborar un producto de alta calidad que cumpla con los requisitos establecidos por el cliente en el tiempo acordado y con el costo esperado.

Como consecuencia de lo antes expuesto en la ingeniería de software actual se pueden encontrar varias metodologías entre las que se pueden mencionar: RUP, XP, Crystal, Scrum, Evo, FDD, ASD, LSD, AUP. A continuación se hará una descripción de algunas seleccionadas como parte del estudio realizado.

1.5.1 Programación Extrema (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. La metodología XP se basa en:

Pruebas unitarias

Son pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueden hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

Re fabricación

Consiste en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares

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

La misma se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Considera que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable para el desarrollo de proyectos. (3)

1.5.2 El Proceso Unificado de Desarrollo de Software (RUP)

Es una de las metodologías más generales y usadas que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto. Constituye además, una propuesta de proceso para el desarrollo de software orientado a objeto, utilizando Unified Model Lenguage (UML), como lenguaje para visualizar, especificar, construir, y documentar algunos artefactos que se crean durante el proceso.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Ver Figura 1.

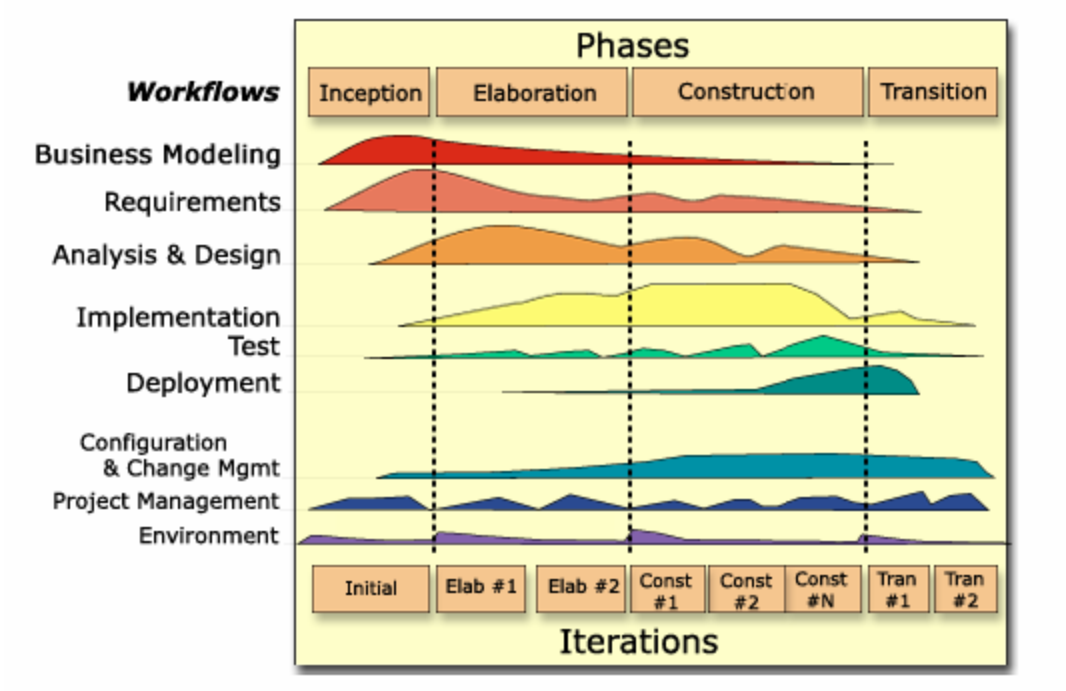


Figura 1. RUP en Dos Dimensiones.

El ciclo de vida de RUP se caracteriza por:

Dirigido por casos de uso: Los casos de uso (CU) reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo con la que el equipo del proyecto y los usuarios deben estar de acuerdo, ya que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones,

comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero irá creciendo incrementalmente en cada iteración.

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

RUP está basado en 5 principios de desarrollo, estos son:

Adaptar el proceso*:* el proceso debe adaptarse a las características particulares del proyecto, el tamaño del mismo, así como las regulaciones que condicionan su funcionamiento ya que estas influirán en su diseño específico.

Balancear Propiedades*:* puesto que los requerimientos de los participantes en el proyecto pueden ser diferentes o contradictorios, se debe encontrar un balance que satisfaga de alguna manera los deseos de todos los integrantes.

Demostrar valor iterativamente*:* la entrega del proyecto, aunque sea de un modo interno, se realiza en etapas iteradas. En cada una de las iteraciones se analizan las opiniones de inversores, la estabilidad y calidad del producto para poder refinar la dirección del proyecto e identificar riesgos.

Elevar el nivel de abstracción: este principio dominante motiva el uso de conceptos reutilizables tales como el patrón de software. Esto evita que los ingenieros de software vayan directo de los requisitos a la codificación, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos pensando en la reutilización de código.

Enfocarse en la calidad: el control de la calidad en el proyecto no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente. (4)

1.6 Lenguaje Unificado de modelado. (UML)

La modelación ha sido una parte esencial tanto en la ingeniería como en el arte y la construcción; tal es el caso del diseño de un software, que supuestamente complejo podría complicarse si no se hace pleno uso de diagramas que lo identifiquen.

La modelación proporciona tres beneficios clave: visualización, complejidad de gestión y una comunicación clara. En la actualidad el auge de desarrollo de software o aplicaciones complicadas ha crecido hasta el punto de que un equipo de desarrollo de software ve la necesidad de la utilización de diagramas que ayuden al entendimiento pleno del software a desarrollar. Para ello la investigación se apoya en UML, que no es más que un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software totalmente independiente del lenguaje de programación y de las características de los proyectos, ya que ha sido diseñado para poder realizar modelos de cualquier tipo de proyectos, tanto informáticos como de arquitectura.

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y debido a que es un lenguaje, cuenta con reglas para combinar tales elementos. Además es importante recalcar que 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. (5)

1.7 Fundamentación de la metodología seleccionada

Con lo expuesto anteriormente sobre los procesos de desarrollo de software y en vista de que el proyecto debe ser suficiente y adecuadamente documentado, producto de que el cliente desea integrarlo como solución a un sistema tecnológicamente superior; y teniendo en cuenta que aunque es un sistema pequeño, los requisitos deben ser estables ya que el usuario del sistema no interactúa en ningún momento con el equipo de desarrollo.

Y teniendo en cuenta que RUP:

- Define roles fundamentales que tienen a su cargo la generación de artefactos y documentación correctos por cada fase y flujo que tributan a un buen producto final.
- Es una metodología establecida y una de las más usadas mundialmente.
- Está respaldada por buenos resultados en proyectos en el mundo y en la UCI.
- Está diseñada de forma que exista un entendimiento continuo y gradual de todos los implicados en el proyecto.
- Centra su ciclo de vida en la arquitectura.

Se determinó adoptar esta metodología para guiar el desarrollo del proyecto, por todas las ventajas de organización que brinda y debido a que goza de un grupo de características y facilidades, que hacen más dinámico el desarrollo del trabajo.

1.8 Herramientas CASE

Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida del desarrollo de software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

1.8.1 Rational Rose Enterprise (Rational Software Corporation, 2002)

Esta herramienta CASE creada por los ingenieros (Booch, Rumbaugh y Jacobson) es de fácil utilización. Permite una modelación absoluta de los procesos del negocio y del sistema, además resulta de gran utilidad para los desarrolladores de proyectos, pues ella cubre todo el ciclo de vida del proyecto desde su fase inicial hasta su culminación.

Esta herramienta posibilita establecer una trazabilidad entre los modelos (análisis y diseño) y el código ejecutable, también permite el desarrollo de software en equipo basado en la metodología RUP. Cada rol

tiene su propia vista de arquitectura (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y funcionalidad del sistema en construcción.

Cada analista, desarrollador o diseñador puede usar Rational Rose para definir y comunicar el negocio, el diseño y la arquitectura de la aplicación que se esté desarrollando. Es una completa solución para mostrar de forma gráfica el análisis de los procesos del negocio y los requerimientos del sistema. (6)

1.8.2 Visual Paradigm

Visual Paradigm es una herramienta muy potente y fácil de utilizar para crear los artefactos necesarios en la confección de un software. Está diseñada para varios tipos de usuarios, incluyendo Ingenieros de Software, Analistas de Sistemas, Analistas de Negocio y Arquitectos de Sistemas. Es una suite completa de herramientas CASE que da soporte al modelado visual con UML 2.0 ofreciendo distintas perspectivas del sistema.

Resulta ser independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software así como garantizar la calidad del producto final. Es posible generar código desde Visual Paradigm para plataformas como .NET y lenguajes de programación como Java y PHP, así como obtener diagramas de clases a partir del código, siendo de gran utilidad pues ahorra tiempo a los desarrolladores y reduce las posibilidades de cometer errores.

Brinda la posibilidad de obtener una Base de Datos relacional y el código necesario para acceder a esta a partir de un diagrama Entidad – Relación; así como conectarse fácilmente a varios servidores de Base de Datos e integrarse con varios ambientes de desarrollo integrados (IDE) lo cual permite pasar del código al modelado y viceversa. Establece interoperabilidad con otras aplicaciones como el Rational Rose y documenta todo el trabajo y especificaciones de Casos de Uso sin necesidad de utilizar herramientas externas, por ejemplo editores de texto, utilizando plantillas que se encuentran o que pueden ser creadas por los usuarios. Además se encuentra disponible en múltiples lenguajes y plataformas: Microsoft Windows (98, 2000, XP, o Vista) Linux, Mac OS X, Solaris o Java. (7)

1.8.3 Fundamentación de la Herramienta seleccionada

Visual Paradigm debido a que ofrece un entorno de creación de diagramas para UML 2.0, tiene la disponibilidad en múltiples plataformas. Está soportada por una gama de lenguajes en la Generación de Código e Ingeniería Inversa en Java, C++, CORBA IDL, PHP, Esquema de XML, Ada y Python. También la Generación de Código soporta C #, VB .NET, Lenguaje de Definición de Objeto (ODL), Flash Action Script, Delphi, Perl, Objetivo-C y Ruby, adicionándole las ventajas que presenta en el diseño de sistemas y la agilidad que proporciona en la modelación. Se determinó emplear esta herramienta en la propuesta de solución por todas las características antes planteadas.

1.9 Lenguaje de programación

Un lenguaje de programación no es más que una construcción mental del ser humano para expresar programas, constituido por un grupo de reglas gramaticales, símbolos utilizables, términos monosémicos (con sentido único) y una regla principal que resume las demás. Para que esta construcción mental sea operable en un computador debe existir otro programa que controle la validez o no de lo escrito llamado traductor.

Para la solución del problema planteado en la introducción de este trabajo, se decidió realizar una aplicación de escritorio, cuyo objetivo será recoger los datos personales de un pelotero y las estadísticas defensivas que le pertenecen a este durante un juego, para luego obtener información estadística concreta. Dado que la infraestructura informática que presenta el Instituto Nacional de Deporte y Recreación (INDER) en sus estadios es insuficiente, por falta de equipos de cómputo y puesto que no están enlazadas en una misma red, se hace muy complejo enviar información de un estadio a otro en tiempo real.

Entre los lenguajes de programación más usados en la actualidad se analizaron los siguientes:

1.9.1 Java

A lo largo de la historia se han creado lenguajes de programación cada vez más especializados y sencillos, los cuales han ido mejorando con respecto a sus antecesores. Un ejemplo es el lenguaje Java; el mismo surgió mucho antes de la creación de la World Wide Web (WWW) como software para

dispositivos electrónicos de consumo. Este lenguaje de programación llevaba otro nombre en sus inicios (Oak) resurgiendo con el nombre de Java en el año 1995.

Una de las principales características que favoreció 0el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Esto significa que el mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además es un lenguaje orientado a objetos que resuelve los problemas en la complejidad de los sistemas, entre otras.

Java es un lenguaje que agrupa características fundamentales de otros lenguajes y las hace suyas, mejorando su funcionamiento y efectividad, brindando grandes ventajas al mismo:

- Es un lenguaje orientado a objetos: Facilita abordar la resolución de cualquier tipo de problema.
- Es un lenguaje sencillo, aunque sin duda potente.
- La ejecución del código Java es segura y fiable: Los programas no acceden directamente a la memoria del ordenador, siendo imposible que un programa escrito en Java pueda acceder a los recursos del ordenador sin que esta operación le sea permitida de forma explícita. De este modo, los datos del usuario quedan a salvo de la existencia de virus escritos en Java. La ejecución segura y controlada del código Java es una característica única, que no puede encontrarse en ninguna otra tecnología.
- Es totalmente multiplataforma: Es un lenguaje sencillo, por lo que el entorno necesario para su ejecución es de pequeño tamaño y puede adaptarse incluso al interior de un navegador. Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.
- Dinámico: El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas y se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red. (8)

1.9.2 C#

Microsoft Visual C# 2005 es un lenguaje de programación diseñado para crear una amplia gama de aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Con sus diversas innovaciones; permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C.

C# combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, Visual Basic o Delphi. De hecho, su creador Anders Heljsberg fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++. La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic.

Este lenguaje de programación elimina muchos elementos añadidos por otros lenguajes que facilitan su uso y compresión, como por ejemplo ficheros de cabecera, o ficheros fuentes. Es por ello que se dice que C# es auto contenido. Además, no se incorporan al lenguaje elementos poco útiles, como por ejemplo macros, herencia múltiple u operadores diferentes al operador de acceso a métodos (operador punto) para acceder a miembros de espacios de nombres.

Al ser C# un lenguaje de última generación, incorpora elementos que se ha demostrado a lo largo del tiempo que son muy útiles para el programador, como tipos decimales o booleanos, un tipo básico string, así como una instrucción que permita recorrer colecciones con facilidad (instrucción foreach). Estos elementos hay que simularlos en otros lenguajes como C++ o Java.

C# como lenguaje orientado a objetos soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo. En el mismo todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros. Sin embargo, a diferencia de Java, existen modificadores para saltarse esta restricción, pudiendo manipular objetos a través de punteros. Para ello basta identificar regiones de código con el identificador unsafe, y podrán usarse en ellas punteros de forma similar a como se hace en C++. Esta característica puede resultar de utilidad en situaciones en las que se necesite gran velocidad de procesamiento. (9)

1.9.3 Lenguaje de marcado XML

XML es un lenguaje de metamarcado que ofrece un formato para la descripción de datos estructurados facilitando unas declaraciones de contenido más precisas y unos resultados de búsquedas más significativos en varias plataformas. Además, XML habilitará una nueva generación de aplicaciones para ver y manipular datos basadas en el Web; ofrece una representación estructural de los datos que se pueden implementar ampliamente y es fácil de distribuir.

XML es un subconjunto de Standard Generalized Markup Language (SGML) optimizado para el Web. Definido por el WWW Consortium (W3C), garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes. La interoperabilidad resultante está creando rápidamente una nueva generación de aplicaciones de comercio electrónico en el Web.

Proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos. XML se propone como lenguaje de bajo nivel para intercambio de información estructurada entre diferentes plataformas. Se puede utilizar en bases de datos, editores de texto, hojas de cálculo, etc. (10)

1.9.4 Fundamentación del Lenguaje de programación

Como lenguaje de programación, se decidió utilizar Java por ser un lenguaje multiplataforma, ya que posibilita al software correr en cualquier tipo de sistema operativo, es una plataforma de software libre; o sea contiene código abierto.

Además de las características antes mencionadas, es un lenguaje que ha demostrado ser uno de los más eficientes en el mundo de la programación. C# es un buen lenguaje, pero no es realmente multiplataforma, solo es multiplataforma si se usa la máquina virtual de Mono. Finalmente se recomienda para la aplicación usar Java + Netbeans. No se tendrá que pagar a nadie por las herramientas de desarrollo y la aplicación correrá en cualquier plataforma: Windows, Linux (Ubuntu) o UNIX (PC-BSD).

Como el INDER actualmente no cuenta con una base de datos central que pueda recopilar toda la información que le pueda dar una aplicación de este tipo se decidió utilizar XML como almacenador permanente de información, ya que es un lenguaje orientado a identificar estructuras de datos en un

documento. La especificación XML define la manera estándar de cómo hay que realizar el marcado de expresiones en un documento no estructurado, para que con dicho marcado se defina una determinada estructura de datos; y se almacenaría permanentemente la información en forma de ficheros.

1.10 Herramientas de Desarrollo

Las herramientas de desarrollo es algo imprescindible en la producción de software. Ya que nos aporta el basamento tecnológico que necesitamos para implementar y probar la aplicación; normalmente una herramienta soporta uno o varios lenguajes de programación para varios tipos de aplicaciones.

Teniendo en cuenta el lenguaje de programación seleccionado se valoraron algunas herramientas que soportaban el lenguaje Java, entre las cuales se mencionan:

1.10.1 NetBeans

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Esta plataforma es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)

- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogo paso a paso).

1.10.2 Eclipse

Eclipse es como una tienda donde no solamente se hacen productos, sino que además se hacen las herramientas para hacer los productos. Eclipse contiene un equipo de instrumentos para desarrollo en Java o Java Development Toolkit (JDT) para escribir y depurar programas en Java; además se obtiene un ambiente de desarrollo de Plug-in

Development Enviroment para heredar de Eclipse. Si todo lo que se quiere es un IDE para Java, no se necesita nada además que el JDT. Esto es para lo que la mayoría las personas usan Eclipse.

Aunque Eclipse es escrito en Java y su principal uso es como IDE para Java, este es un lenguaje neutral. El soporte para desarrollo en Java es proveído por un componente enchufado o plug-in, pero además están disponibles plugins para otros lenguajes, como C/C++, Cobol, C#, PHP. En principio permite ejecutar un programa sobre cualquier plataforma. Es una extensible plataforma de código abierto para desarrollar herramientas.

1.10.3 Propuesta de la herramienta a utilizar

De las herramientas de desarrollo mencionadas anteriormente y analizando las necesidades de la presente investigación, se hace conveniente la selección de la herramienta NetBean debido a las simples características que presenta el sistema a desarrollar. Es preciso mencionar que el entorno de desarrollo Eclipse es mucho más robusto y eficiente, pero la aplicación que se desea desarrollar no necesita de la instalación de varios plug-in debido a que el acceso a datos no es contra un SGBD externo sino en manejo de ficheros XML.

1.11 Conclusiones

En este capítulo se realizó un resumen de los principales conceptos asociados al dominio del problema con el objetivo de lograr un mayor entendimiento del mismo. Se explica de manera argumentada la situación problémica existente. También se realizó un análisis detallado de algunas aplicaciones existentes en el mundo y en nuestro país que sirven de guía en los objetivos que persigue este trabajo.

Para la solución del problema planteado en la investigación se quiere lograr una aplicación de escritorio que sea capaz de calcular las estadísticas defensivas tanto de un jugador como de un equipo; además que brinde la posibilidad de guardar en ficheros XML los datos personales y las acciones defensivas de cada pelotero.

Se efectuó un estudio profundo de algunas herramientas y metodologías para la realización del sistema llegándose a la conclusión que para cumplir los objetivos se debe trabajar con la metodología RUP, como lenguaje modelado UML, la herramienta de modelado Visual Paradigm, el lenguaje de programación Java, la herramienta de desarrollo NetBean y el lenguaje XML para hacer archivos de carácter persistente donde se puedan almacenar los datos de cada jugador recogido durante el juego.

Capítulo 2 Características del Sistema

2.1 Introducción

En el siguiente capítulo se trabajará en los flujos de trabajo de negocio y requerimiento. Se desarrollará un modelo de dominio para una mejor comprensión del negocio. Se describirán los requisitos funcionales y no funcionales. Finalmente se podrá comprender el diagrama de casos de uso del sistema y la descripción de cada uno de ellos.

2.2 Reglas del negocio

En la aplicación que se quiere lograr existe reglas que especifican condiciones que deben ser ciertas para asegurarse que una operación se ejecute correctamente. Las reglas que se tienen son:

1. No pueden existir dos jugadores que poseen un mismo número que los identifique.
2. No puede existir un equipo con el mismo nombre, lo que traería error en el sistema.

2.3 ¿Qué es un modelo de dominio?

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema. El modelo de dominio se describe mediantes diagramas de UML (específicamente mediante diagramas de clases).

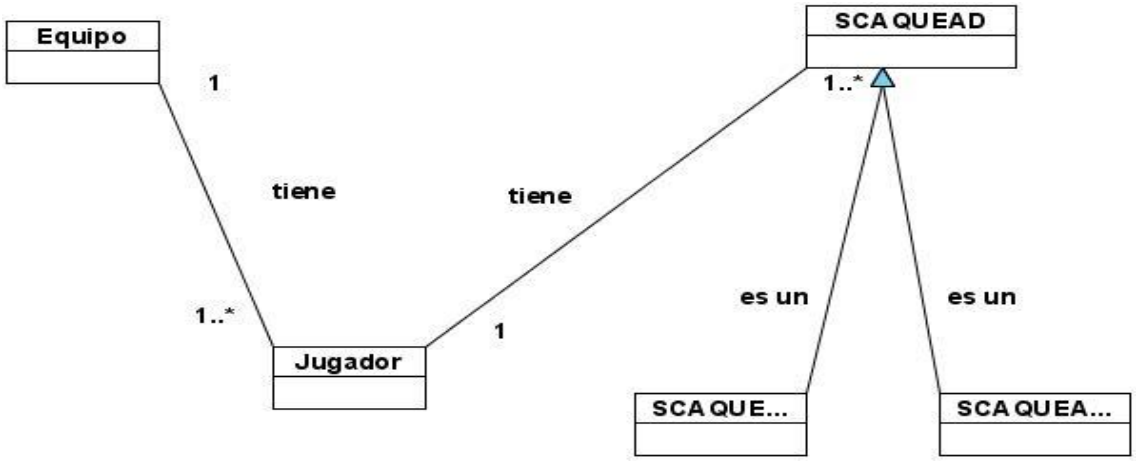


Figura 2.Modelo de Dominio

2.4 Glosario de términos del dominio

Scaque AD: Casilla donde se insertan las acciones defensivas de un jugador.

Scaque ADF: Casilla donde se insertan las acciones defensivas y fildeo de un jugador.

Scaque ADFT: Casilla donde se insertan las acciones defensivas y fildeo con tiro de un jugador.

Equipo: Un equipo comprende a cualquier grupo de personas unidas con un objetivo común.

Jugador: Que juega o participa en un juego o deporte, en este caso es el que juega béisbol.

2.5 Captura de Requisitos

A continuación se enumeran los requisitos funcionales y no funcionales del sistema.

2.5.1 Requisitos Funcionales

R1. Gestionar Jugador.

R1.1 Verificar que no se encuentre el jugador que se quiere insertar.

R1.2 Insertar los datos de un Jugador.

R1.3 Mostrar datos de un Jugador seleccionado.

R1.4 Modificar los datos un Jugador seleccionado.

R1.5 Eliminar un jugador seleccionado.

R2. Gestionar Equipo.

R2.1 Verificar que el equipo que se quiere crear no esté creado.

R2.2 Crear un Equipo.

R2.3 Mostrar datos un Equipo.

R2.4 Modificar un Equipo

R2.5 Eliminar un Equipo.

R3. Insertar SCAQUE.

R4. Consultar Jugador.

R5. Calcular coeficiente integrador de fildeo para un Jugador.

R6. Calcular coeficiente integrador de fildeo para un Equipo.

R7. Mostrar árbol defensivo de un Jugador seleccionado.

R8. Exportar a PDF el árbol defensivo de un jugador y consultar jugador.

R9. Autenticar.

2.5.2 Requisitos No Funcionales.

Interfaz externa

1. Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.

Portabilidad

2. El sistema deberá funcionar sobre plataforma Windows.

Hardware

3. Requiere estar instalada en una PC Pentium, 256 Mb de RAM.

Software

4. Se debe disponer del sistema operativo Windows, Linux (Ubuntu) o UNIX (PC-BSD).

Confidencialidad

5. La información manejada por el sistema deberá estar protegida de acceso no autorizado.

Seguridad

6. Garantizar que la información sea vista únicamente por quien tiene derecho a verla.

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

8. Verificación de la integridad de los archivos de datos que pueden trasladarse en ficheros XML.

9. Cuando se crea el archivo XML se le otorga a este permiso de lectura únicamente.

Confiabilidad

10. Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.

2.6 Modelo de casos de uso del sistema

En este epígrafe se enumeran los actores del sistema, y a partir de los requisitos funcionales se obtendrá los casos de uso con sus descripciones correspondientes.

2.6.1 Definición de los actores del sistema

Tabla 1 .CU Gestionar Jugador

Actor	Descripción

Anotador	Es la única persona encargada de trabajar con la aplicación, donde tiene acceso a todas las funcionalidades de la misma.

2.6.2 Casos de uso del sistema

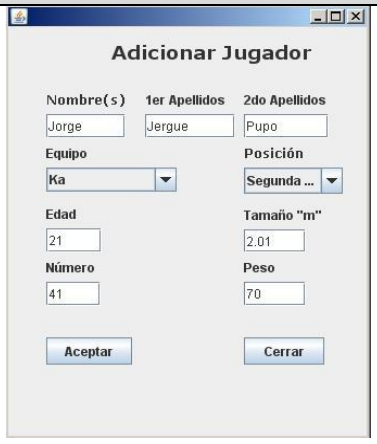
A continuación se presentan los casos de uso determinados para satisfacer los requerimientos funcionales del sistema:

1. Gestionar Jugador.
2. Gestionar Equipo.
3. Insertar SCAQUE.
4. Consultar Jugador.
5. Calcular coeficiente integrador de fildeo para un Jugador.
6. Calcular coeficiente integrador de fildeo para un Equipo.
7. Mostrar árbol defensivo del jugador.
8. Autenticar.

2.6.3 Descripciones de los casos de usos

Tabla 1 .CU Gestionar Jugador

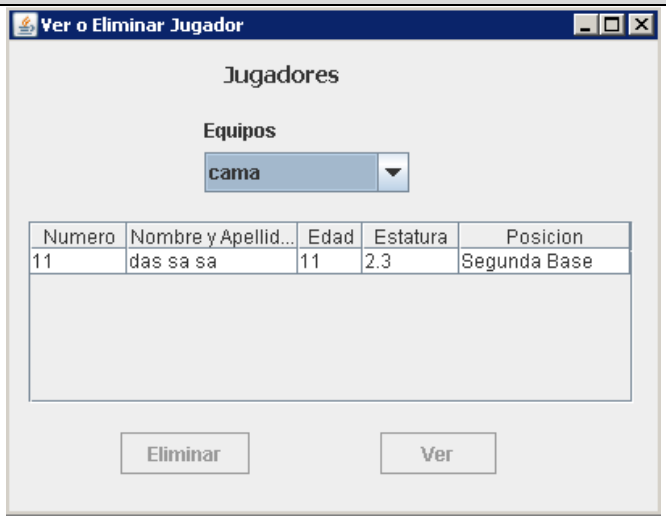
Caso de uso:	**Gestionar Jugador.**
Actores:	Anotador.
Propósito:	Insertar, Modificar, Mostrar y Eliminar los datos de un jugador.
Resumen:	El caso de uso se inicia cuando el anotador accede a una de las opciones de Insertar, Modificar, Mostrar y Eliminar los datos de un jugador. El caso de uso termina cuando el sistema devuelve la opción solicitada.
Referencia:	RF1
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. El anotador selecciona que desea hacer con el jugador:	2.a) Si desea registrar un nuevo jugador, ver sección: Insertar jugador b) Si desea eliminar un jugador , ver sección: Eliminar jugador c) Si desea modificar un jugador, ver sección: Modificar jugador. d) Si desea ver a un jugador, ver sección: Ver jugador.
Sección: "Insertar Jugador"	
1. El anotador escoge la opción de "Insertar jugador"	2. El sistema muestra un campo para insertar los datos del jugador.
3. El anotador ingresa los datos solicitados por el sistema.	4. El sistema verifica que los campos del formulario no estén vacíos.
	5. El sistema verifica que el jugador no esté registrado.
	6. El sistema Inserta al jugador y termina el CUS.
Flujos Alternos de los Eventos	
4. El anotador deja algún campo vacio.	4.1 Muestra un mensaje de error indicando que debe llenar todos los campos. Retorna a la acción 2.
5. Los datos del jugador se encuentran inscrito en la aplicación.	5.1. Muestra un mensaje de error indicando que el jugador ya existe.
Prototipo No Funcional	
	

Sección : “Eliminar Jugador”	
1. El anotador escoge la opción de “Eliminar jugador”	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos y a los jugadores que integran cada equipo.
3. El anotador selecciona el equipo de donde quiere eliminar al jugador.	4. El sistema automáticamente carga todos los jugadores que integran ese equipo
5. El anotador confirma que desea eliminar el jugador	6. El sistema elimina al jugador y muestra un mensaje: “El jugador fue eliminado” y termina el CUS.

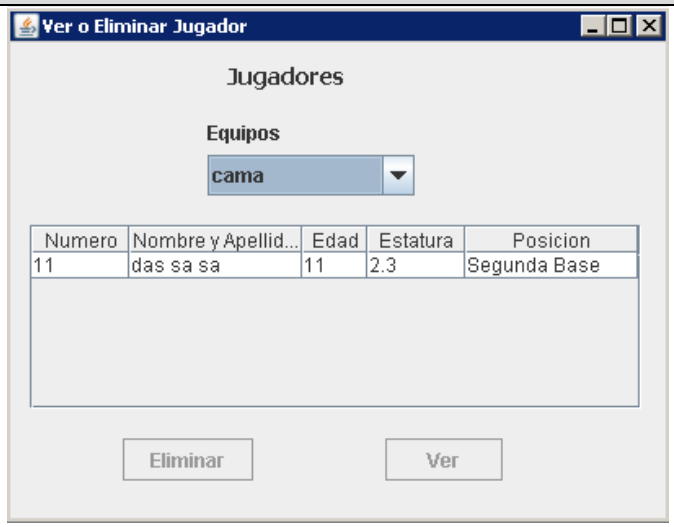
Flujos Alternos de los Eventos	
5. El anotador selecciona la opción cancelar.	5.1. Se cancela la acción y se culmina el CUS sin ejecutar ninguna acción.

Prototipo No Funcional



Sección : “Modificar Jugador”	
1. 1. El anotador escoge la opción de “Modificar Jugador”.	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos y a los jugadores que integran cada equipo.
3. El anotador selecciona el equipo de donde quiere modificar al jugador.	4. El sistema automáticamente carga todos los jugadores que integran ese equipo
5. El anotador selecciona el jugador que	6. El sistema muestra en ese campo los datos

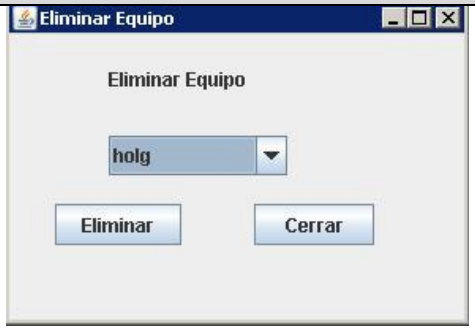
se quiere modificar	actuales del jugador.
7. El anotador modifica los datos necesarios y pincha el botón Aceptar.	8. El sistema los actualiza y muestra un mensaje "Los datos del jugador ya fueron actualizados" y termina el CUS.
Sección : "Mostrar Datos del Jugador"	
1. El anotador escoge la opción de "Mostrar Datos del Jugador".	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos y a los jugadores que integran cada equipo.
3. El anotador selecciona el equipo de donde quiere ver al jugador.	4. El sistema automáticamente carga todos los jugadores que integran ese equipo
5. El anotador selecciona al jugador	6. El sistema muestra los datos del jugador que fue seleccionado.
Prototipo No Funcional	



Precondiciones:	**-----**
Poscondiciones:	Se inserta, actualiza o elimina un jugador.
Prioridad: Crítico.	

Tabla 2.CU Gestionar Equipo

Caso de uso:	**Gestionar Equipo**
Actores:	Anotador.
Propósito:	Insertar, Modificar, Mostrar y Eliminar los datos de un equipo.
Resumen:	El caso de uso se inicia cuando el anotador accede a una de las opciones de Insertar, Modificar, Mostrar y Eliminar los datos de un equipo. El caso de uso termina cuando el sistema devuelve la opción solicitada.
Referencia	RF2
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El anotador selecciona que desea hacer con el Equipo:	2.a) Si desea registrar un nuevo Equipo, ver sección: Insertar Equipo b) Si desea eliminar un equipo , ver sección: Eliminar Equipo c) Si desea modificar un equipo, ver sección: Modificar Equipo. d) Si desea ver a un equipo, ver sección: Ver Equipo.
Sección: "Insertar Equipo"	
1. El anotador escoge la opción de "Insertar Equipo"	2. El sistema muestra un campo para insertar los datos del Equipo.
3. El anotador ingresa los datos solicitados por el sistema.	4. El sistema verifica que los campos del formulario no estén vacíos.
	5. El sistema verifica que el Equipo no esté registrado.
	6. El sistema Inserta al equipo y termina el CUS.
Flujos Alternos de los Eventos	
4. El anotador deja algún campo vacio.	4.1. Muestra un mensaje de error indicando que debe llenar todos los campos. Retorna a la acción 2.

5. El equipo se encuentra inscrito.	5.2. Muestra un mensaje de error indicando que el equipo ya existe.
Prototipo No Funcional	
	
Sección : "Eliminar Equipo"	
1. El anotador escoge la opción de "Eliminar Equipo"	2. El sistema muestra un campo donde están todos los equipos y permite seleccionar un Equipo.
3. El anotador selecciona al equipo confirma que desea eliminarlo.	4. El sistema elimina al equipo y a los integrantes de él, mostrando después de esta acción un mensaje diciendo "El Equipo fue eliminado" y termina el CUS
Flujos Alternos de los Eventos	
3. El anotador selecciona la opción cancelar.	3.1. Se cancela la acción y se culmina el CUS sin ejecutar ninguna acción.
Prototipo No Funcional	
	
Sección : "Modificar Equipo"	
1. El anotador escoge la opción de "Modificar Equipo".	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos.

3. El anotador selecciona el equipo que se quiere modificar.	4. El sistema muestra en ese campo los datos actuales del equipo.
5. El anotador modifica los datos necesarios y pincha el botón Aceptar.	6. El sistema los actualiza y muestra un mensaje "Los datos del Equipo ya fueron actualizados" y termina el CUS.

Prototipo No Funcional

Sección : "Mostrar Datos del Equipo"

1. El anotador escoge la opción de "Mostrar Datos del Equipo".	2 El sistema muestra un campo donde están todos los equipos.
3. El anotador selecciona al equipo.	4. El sistema muestra los datos del equipo seleccionado.

Prototipo No Funcional

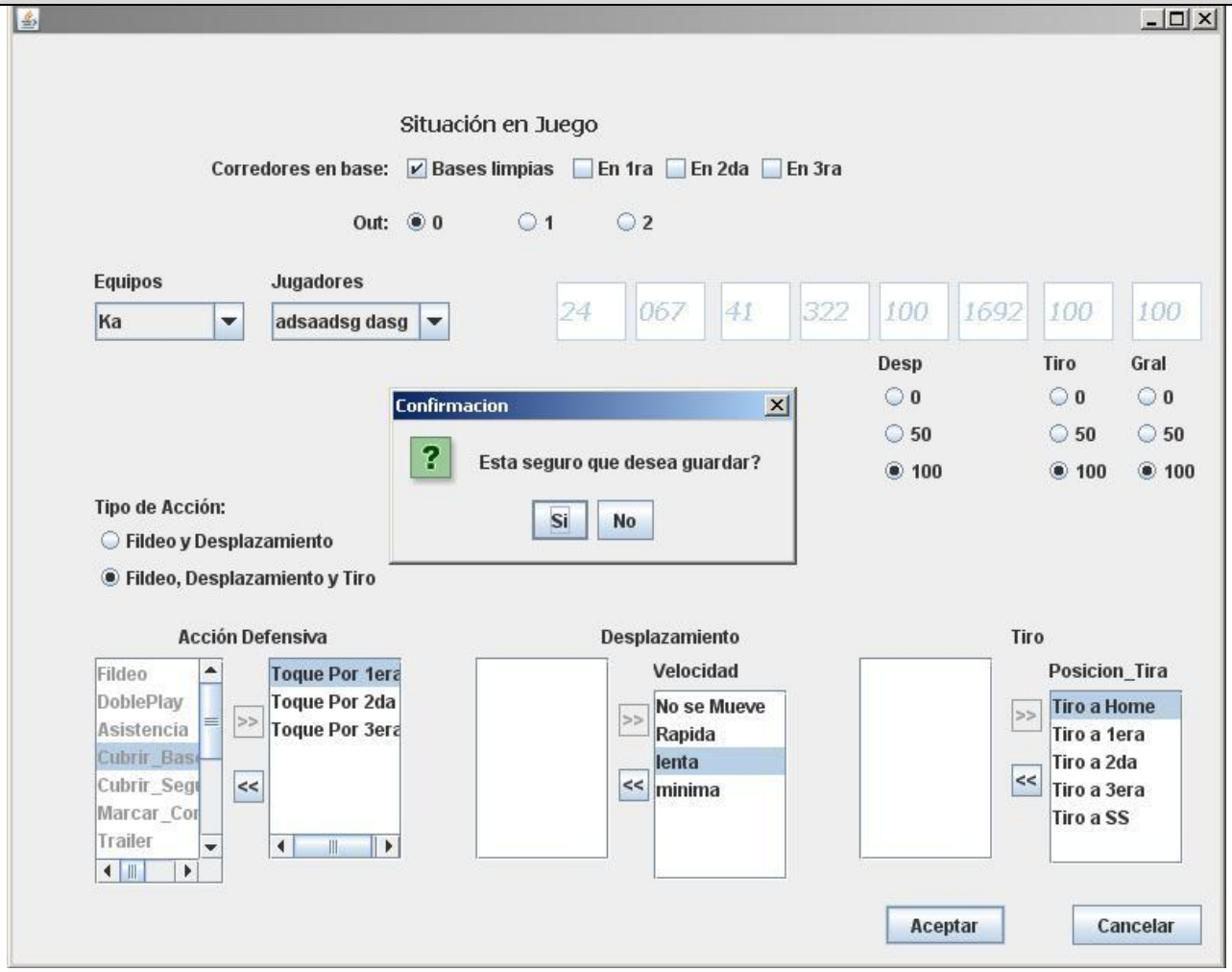
Precondiciones:	
Poscondiciones:	Se inserta, actualiza o elimina un Equipo.
Prioridad: Crítico.	

Tabla 3.CU Insertar SCAQUE

Caso de uso:	**Insertar SCAQUE.**
Actores:	Anotador.
Propósito:	Insertar la codificación perteneciente a un jugador en una acción defensiva.
Resumen:	El caso de uso se inicia cuando el anotador escoge la opción de Insertar SCAQUE para fijar la codificación perteneciente a un jugador.
Referencia	RF3
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El anotador selecciona la opción Insertar SCAQUE.	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos y a los jugadores que integran cada equipo.
3. El anotador selecciona el equipo de donde quiere escoger al jugador que se le quiere insertar el scaque.	4. El sistema automáticamente carga todos los jugadores que integran ese equipo.
5. El anotador selecciona al jugador que se le quiere insertar el scaque.	6. El sistema muestra un campo para insertar los datos del scaque de ese jugador.
7. El anotador Insertar las acciones defensiva realizada por el jugador.	8. El sistema muestra un mensaje de la descripción completa de la acción a la defensiva del jugador.
9. El anotador acepta si los datos puestos anteriormente están correctos.	10. El sistema guarda los datos, termina el CUS.
Flujos Alternos de los Eventos	

9. Los datos puesto por el anotador son incorrectos.	9.1 Cancela y retorna a la acción 2.

Prototipo No Funcional



Precondiciones:	
Poscondiciones:	Se Inserta un nuevo SCAQUE
Prioridad: Crítico.	

Tabla 4.CU Calcular coeficiente integrador de fildeo para un Jugador

Caso de uso:	**Calcular coeficiente integrador de fildeo para un jugador.**
Actores:	Anotador.
Propósito:	Calcular coeficiente integrador de fildeo para cada jugador.
Resumen:	El caso de uso se inicia cuando el anotador escoge la opción de Calcular coeficiente integrador de fildeo para fijar la codificación perteneciente a un jugador y termina cuando esta acción se le aplica a un pelotero específico.
Referencias	**RF5**

Flujo Normal de Eventos

Acción del Actor	Respuesta del Sistema
1. El anotador selecciona la opción calcular coeficiente integrador de fildeo de un jugador	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos y a los jugadores que integran cada equipo
3. El anotador selecciona el equipo de donde quiere escoger al jugador que se le quiere aplicar dicha acción.	4. El sistema automáticamente carga todos los jugadores que integran ese equipo
5. El anotador selecciona al jugador y presiona el botón calcular coeficiente de fildeo.	6. El sistema automáticamente muestra en un mensaje el cálculo efectuado y termina el CUS.

Prototipo No Funcional

Precondiciones:	
Poscondiciones:	Se calcular coeficiente integrador de fildeo para un nuevo jugador.
Prioridad: Crítico.	

Tabla 6.CU Calcular coeficiente integrador de fildeo para un Equipo

Caso de uso:	**Calcular coeficiente integrador de fildeo para un Equipo.**
Actores:	Anotador.
Propósito:	Calcular coeficiente integrador de fildeo para un Equipo.
Resumen:	El caso de uso se inicia cuando el anotador escoge la opción de Calcular coeficiente integrador de fildeo para fijar la codificación perteneciente a un equipo.
Referencia	RF6.

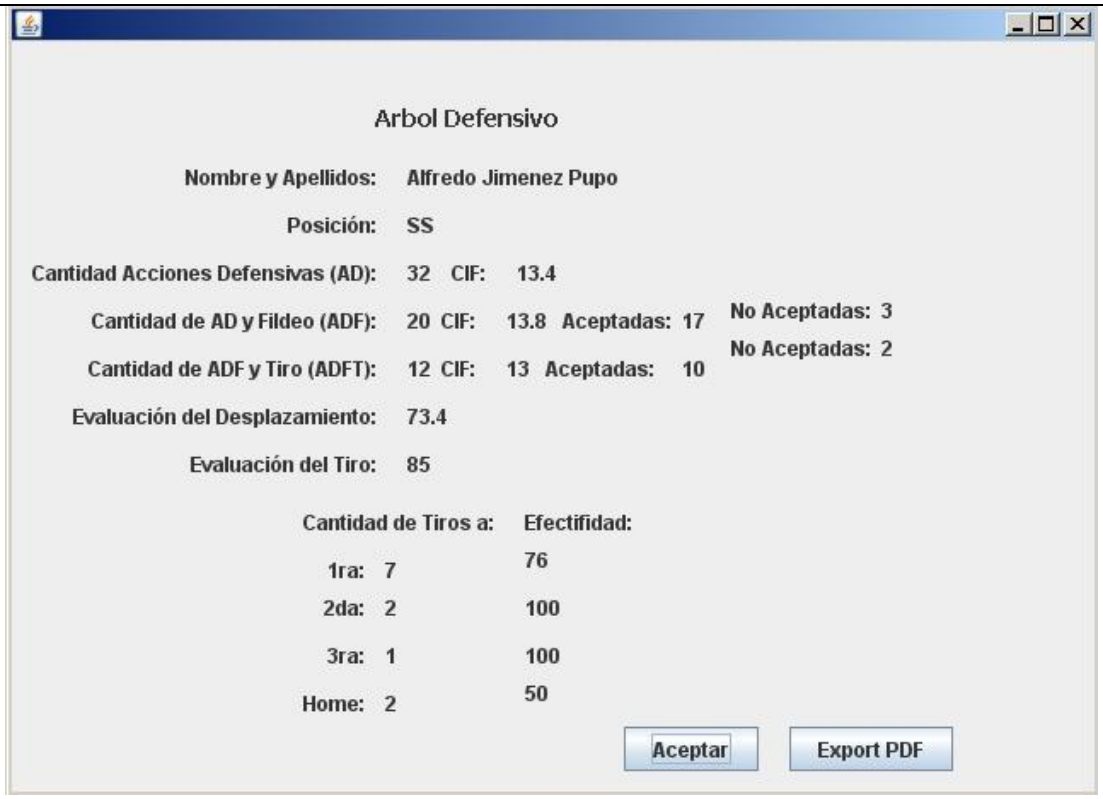
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El anotador selecciona la opción calcular coeficiente integrador de fildeo de un Equipo.	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos.
3. El anotador selecciona el equipo y presiona el botón calcular coeficiente de fildeo.	4. El sistema automáticamente muestra en un mensaje el cálculo efectuado y termina el CUS.

Prototipo No Funcional

Precondiciones:	
Poscondiciones:	Se calcular coeficiente integrador de fildeo para un nuevo equipo.
Prioridad: Crítico.	

Tabla 7 Mostrar Árbol defensivo del jugador

Caso de uso:	**Mostrar Árbol defensivo del jugador**.
Actores:	Anotador.
Propósito:	Mostrar todas las estadísticas de las acciones defensivas existentes de un jugador.
Resumen:	El caso de uso se inicia cuando el anotador escoge la opción de Mostrar Árbol defensivo del jugador y termina cuando el sistema muestra las estadísticas de las acciones defensivas que tiene un jugador.
Referencias	RF7, RF8.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El anotador selecciona la opción Mostrar Árbol defensivo del jugador.	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos y a los jugadores que integran cada equipo.
3. El anotador selecciona el equipo de donde quiere ver el árbol defensivo del jugador.	4. El sistema automáticamente carga todos los jugadores que integran ese equipo.
5. El anotador selecciona al jugador que se le quiere hacer la acción propuesta.	6. El sistema muestra el Árbol defensivo del jugador y da la opción de exportar a PDF terminando así el CUS.
7. El anotador escoge la opción exportar a PDF.	8. El sistema exporta los datos y termina el CUS.
Prototipo No Funcional	



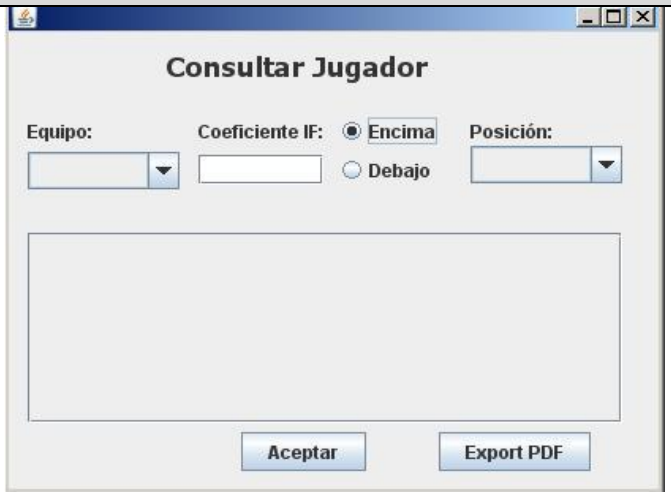
Precondiciones:	
Poscondiciones:	Se Muestra el Árbol defensivo de un nuevo jugador.
Prioridad: Crítico.	

Tabla 8.Cu Consultar Jugador

Caso de uso:	**Consultar Jugador.**
Actores:	Anotador.
Propósito:	Consultar un jugador determinado.
Resumen:	El caso de uso se inicia cuando el anotador escoge la opción de consultar jugador para que se pueda saber los datos de un determinado pelotero.
Referencias	RF4, RF8.
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El anotador selecciona la opción Consultar Jugador	2. El sistema muestra un campo donde el anotador tiene acceso a todos los equipos y a los jugadores que integran cada equipo.
3. El anotador selecciona el equipo de donde quiere consultar al jugador.	4. El sistema automáticamente carga todos los jugadores que integran ese equipo.
5. El anotador selecciona al jugador que se le quiere hacer la acción propuesta.	6. El sistema muestra automáticamente los datos que se quieren del jugador.
7. El anotador escoge la opción exportar a PDF.	8. El sistema exporta los datos y termina el CUS.

Prototipo No Funcional



Precondiciones:	
Poscondiciones:	Consultar información de un jugador
Prioridad: Crítico.	

Tabla 9.Cu Autenticar

Caso de uso:	**Autenticar**
Actores:	Anotador
Propósito:	Permitir al administrador del sistema entrar a la aplicación.
Resumen:	El caso de uso se inicia cuando el anotador introduce usuario y contraseña para entrar a la aplicación.
Referencias	RF9.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El anotador introduce usuario y contraseña.	2. El sistema verifica los datos entrados.
	3. El sistema muestra la aplicación.
Flujos Alternos de los Eventos	
2. El usuario entra mal el usuario o la contraseña.	2.1 Muestra un mensaje de error indicando que el usuario o la contraseña están mal. Retorna a la acción 1.
Prototipo No Funcional	
	
Precondiciones:	
Poscondiciones:	Escoger una de las opciones de la aplicación.
Prioridad: Crítico.	

2.6.4 Diagrama de casos de usos

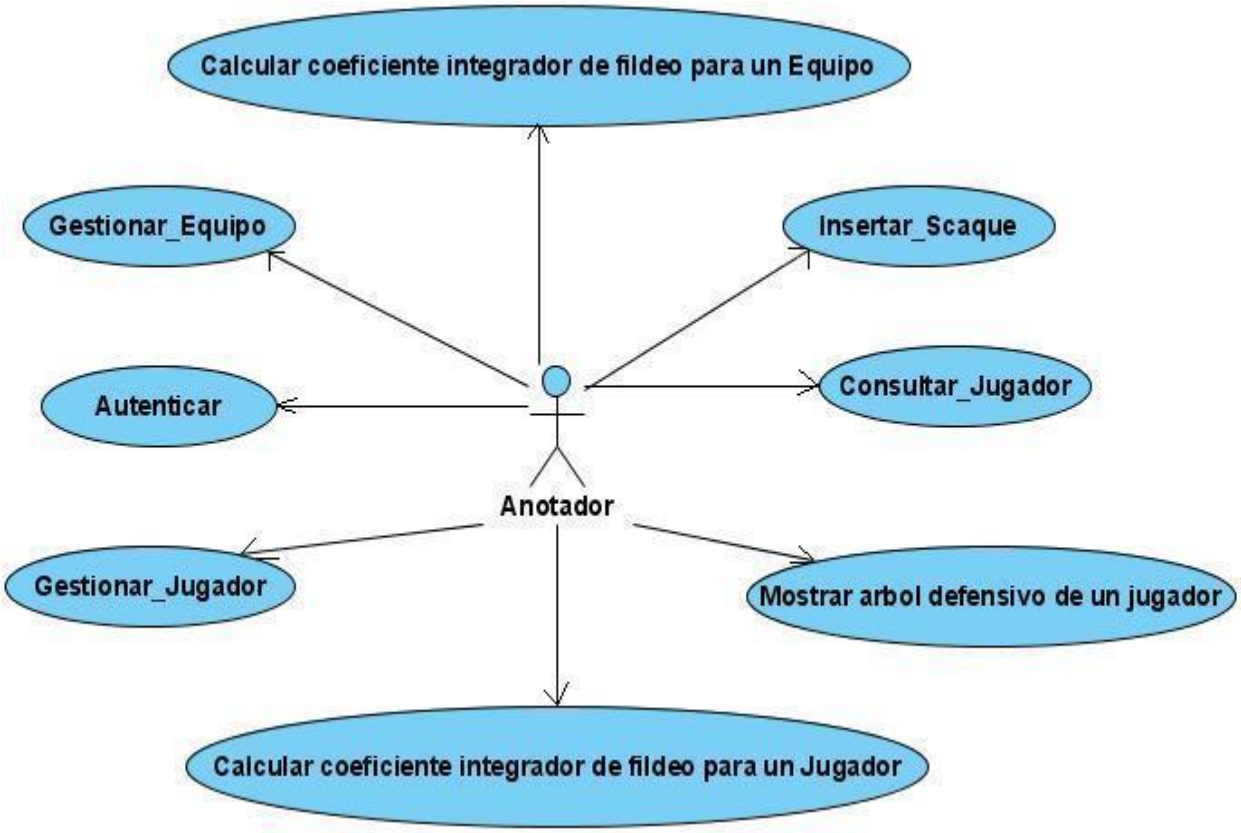


Figura 3.Diagrama de Caso de uso

2.7 Conclusiones

En el presente capítulo se presentaron los artefactos generados en los flujos de trabajo de modelamiento de negocio y requerimientos, a partir de un modelo de dominio del negocio, y se describieron los requisitos funcionales y no funcionales, realizándose una descripción de los casos de uso del sistema en formato expandido para tener una visión más clara de ellos.

Capítulo 3 Análisis y Diseño

3.1 Introducción

En este capítulo se trabajará en los flujos de trabajo de análisis y diseño, para lograr un mejor entendimiento y así facilitar la comprensión de la aplicación a desarrollarse. Finalmente se realizarán los diagramas pertenecientes en cada uno de ellos.

3.2 Análisis

El análisis es la primera representación técnica del sistema; durante esta parte de flujo, se analizan los requisitos que se describieron anteriormente, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requerimientos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero.

Unas de las tareas que se realizan es el análisis del comportamiento de los requisitos que significa:

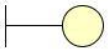
1. Identificar las clases de análisis que son necesarias para el cumplimiento de los requisitos. Conocidas también como las clases de análisis que intervienen para realizar un Caso de Uso.

2. Determinar la ubicación de estas clases de análisis dentro de la estructura lógica del sistema, se refiere, a como estas se agrupan en subsistemas de análisis.

3. Diseñar un prototipo de Interfaz de usuario.

Identificar las clases de análisis.

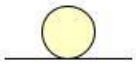
Las clases de análisis representan una abstracción de una o varias clases del diseño. Estas se agrupan en tres estereotipos básicos.

Clase Interfaz 

Las clases Interfaz se utilizan para modelar la interacción entre los actores (usuarios o sistemas externos) y el sistema, esta interacción está vinculada al traspaso de información desde cualquiera de las partes implicadas. Estas representan abstracciones de ventanas, formularios, interfaces de comunicaciones, interfaces de impresoras etc.

Clase Controladora 

Son las clases que efectúan el flujo de eventos en la realización de un caso de uso. Estas representan coordinación, secuencia, transacciones y control entre los objetos; son usadas para encapsular la lógica del negocio.

Clase Entidad. 

Las clases Entidad se utiliza para modelar información que posee larga vida en el negocio, por lo que a menudo describen abstracciones de una o varias entidades persistentes. Describen objetos del mundo real como personas, objeto etc. Son usadas para almacenar la información.

3.3 Realización de Casos de Usos en el análisis.

La realización de los casos de usos en el análisis es una colaboración como parte del modelo de análisis en el que se muestra cómo se ejecuta un determinado caso de uso en términos de clases del análisis y de sus objetos del análisis en interacción.

3.4 Diagramas de clases del análisis.

Una clase de análisis y sus objetos normalmente participan en varias realizaciones de casos de uso, y algunas de las responsabilidades, atributos y asociaciones de una clase concreta suelen ser sólo relevantes para una única realización de caso de uso. Es por esto que es importante coordinar todos los requisitos sobre una clase y sus objetos que pueden tener diferentes casos de uso.

Las clases del análisis representan un primer modelo conceptual de cosas en el sistema que tienen responsabilidades y comportamientos.

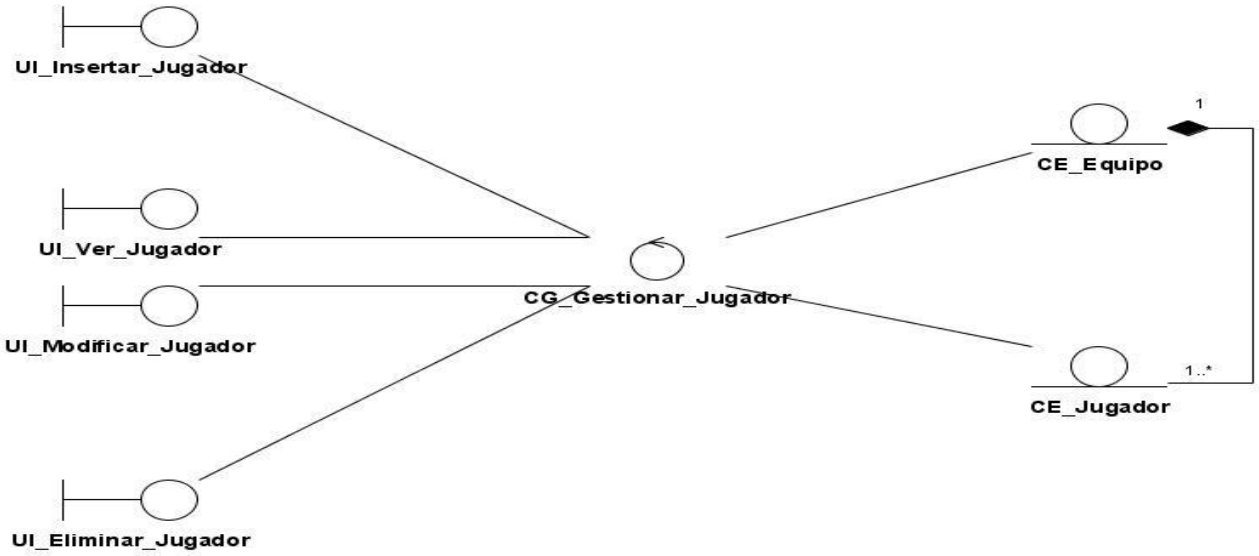


Figura 4.Análisis.CU Gestionar Jugador. Diagrama de Clases.

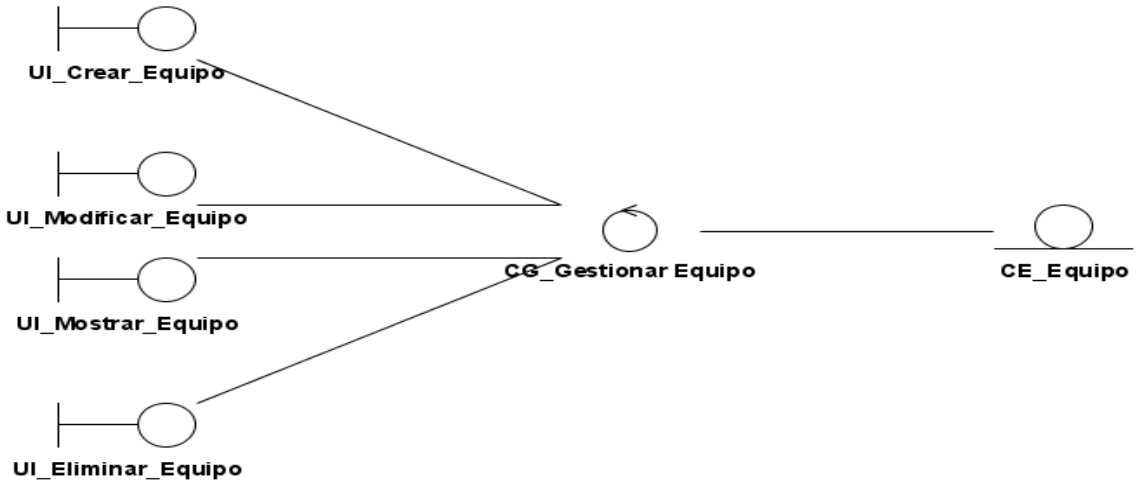


Figura 5. Análisis. CU Gestionar Equipo. Diagrama de Clases

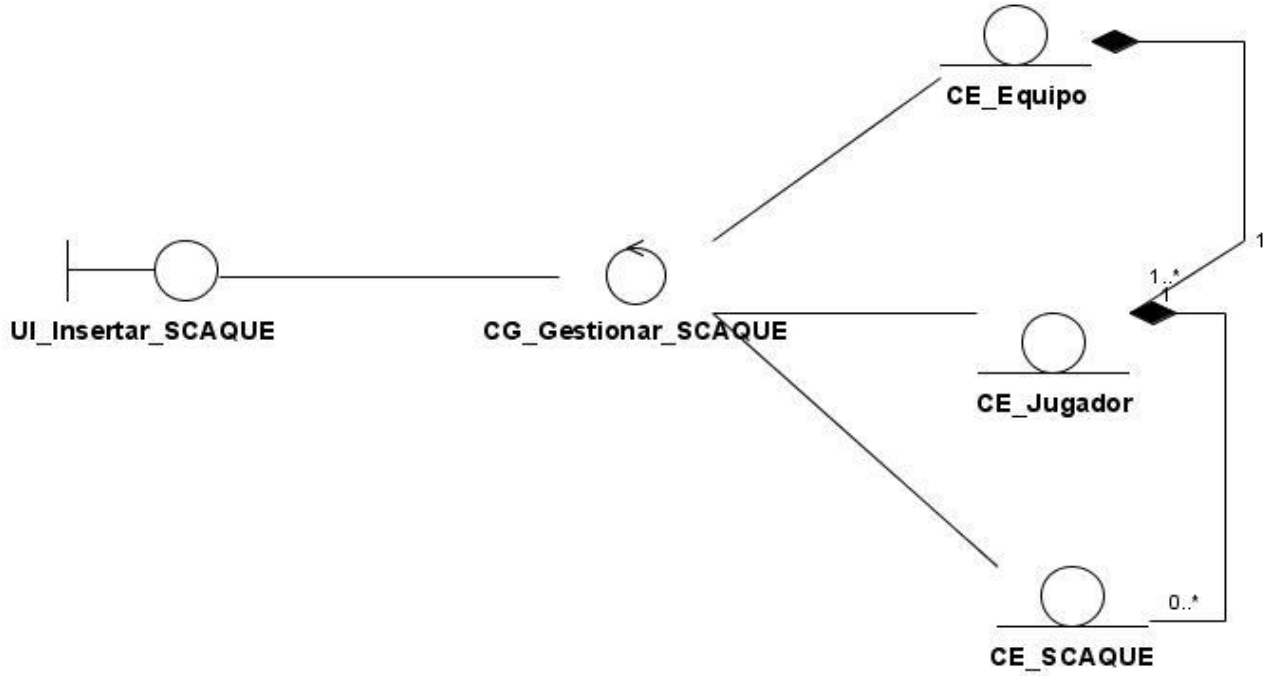


Figura 6. Análisis. CU Insertar SCAQUE. Diagrama de Clases

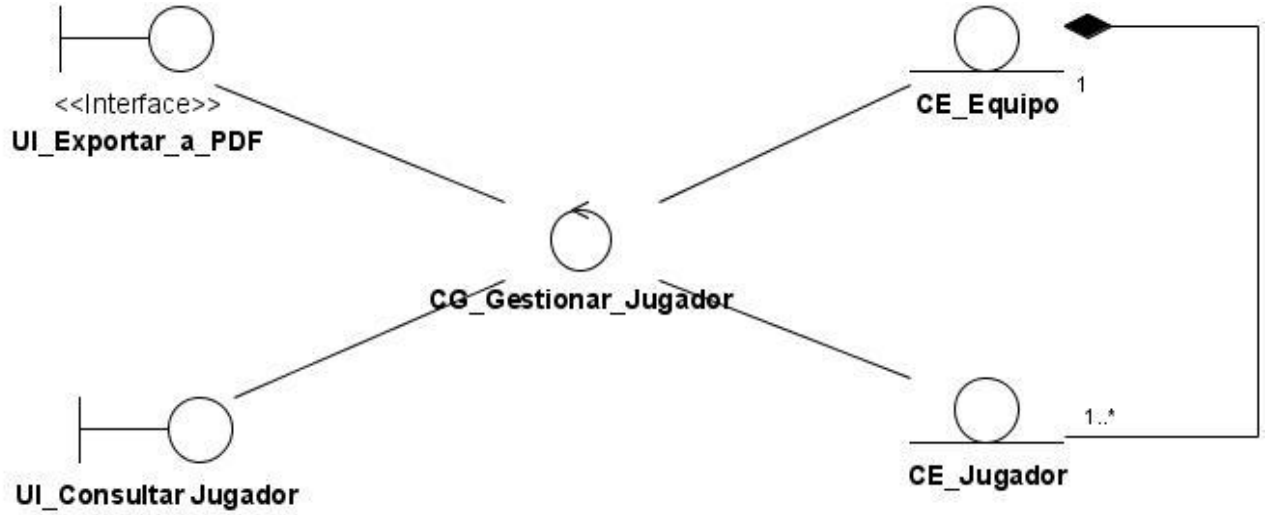


Figura 7. Análisis. Consultar Jugador. Diagrama de Clases

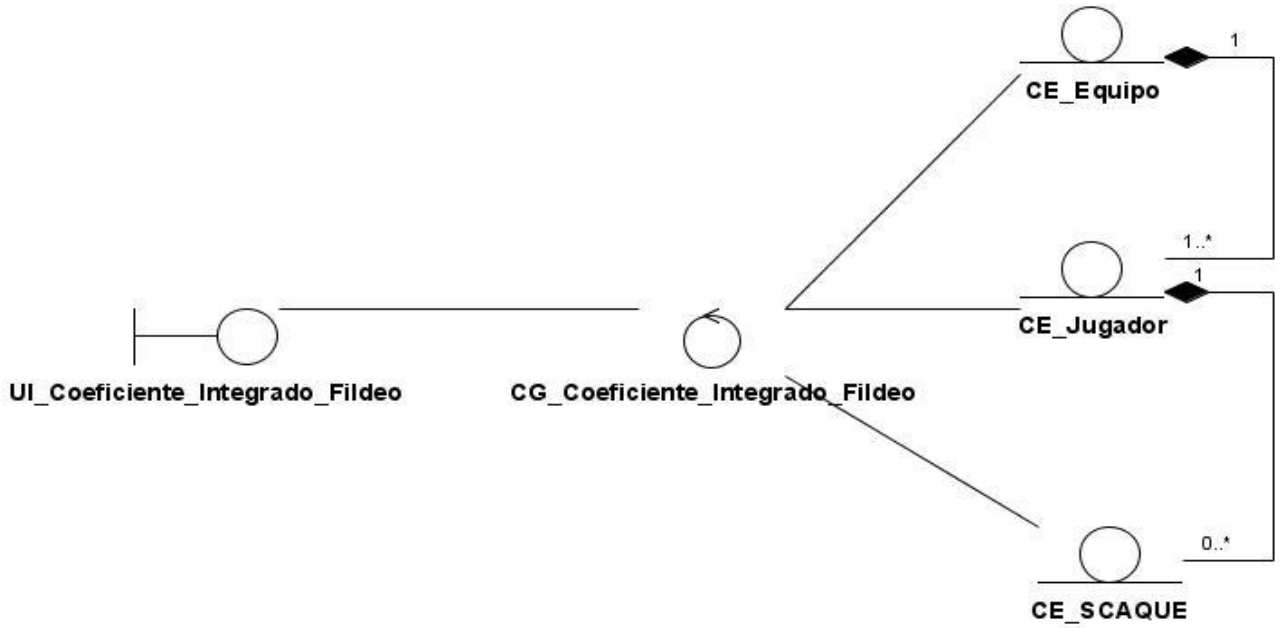


Figura 8. Análisis. CU Calcular coeficiente Integrador de Fildeo para un jugador. Diagrama

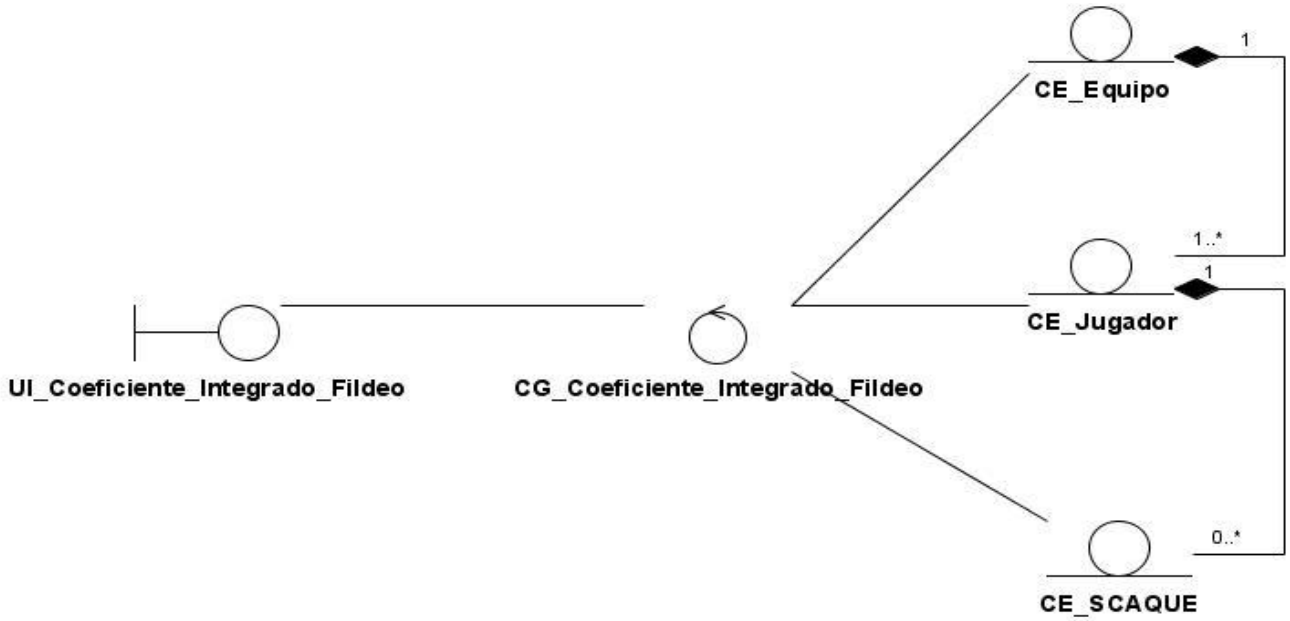


Figura 9. Análisis. CU Calcular coeficiente Integrador de Fildeo para un Equipo. Diagrama de Clases

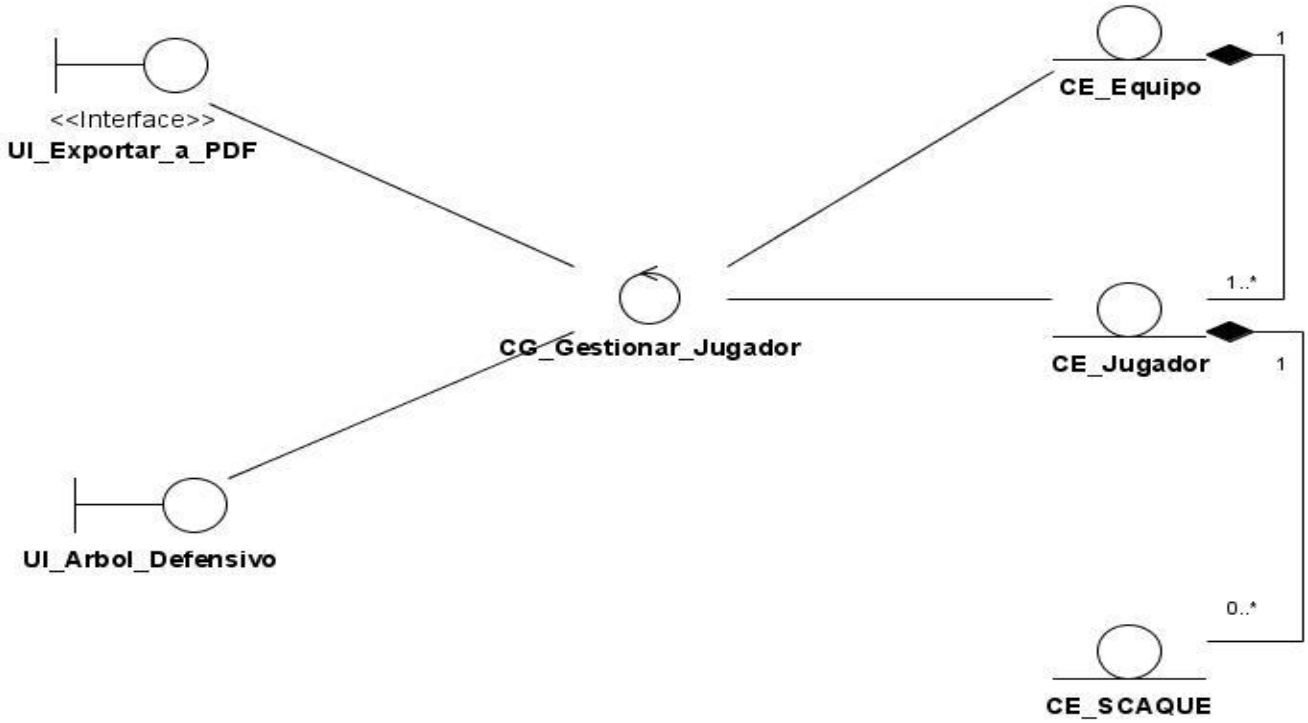


Figura 10. Análisis. CU Mostrar Árbol Defensivo del Jugador. Diagrama de Clases

3.5 Diagramas de colaboración

Un diagrama de colaboración es una representación gráfica que refleja los objetos que participan en la realización de un caso de uso o un escenario de este, además de sus interacciones a través de enlaces y los mensajes que se envían unos a otros.

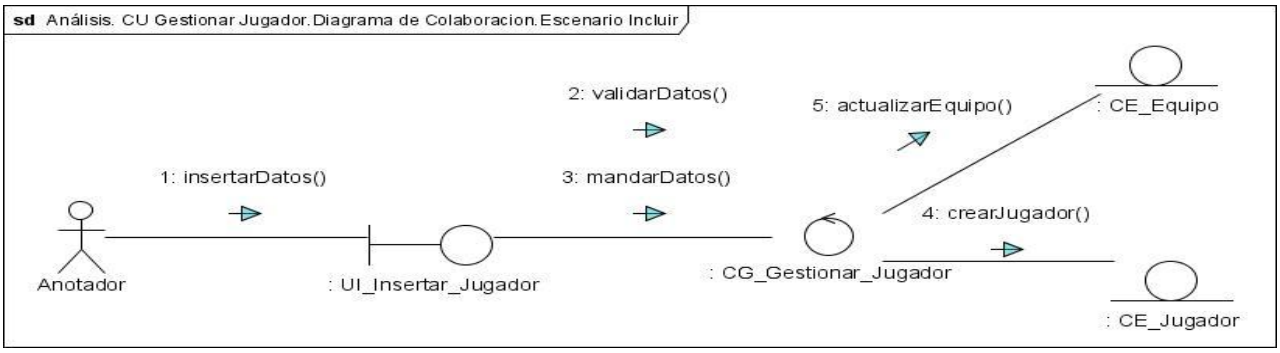


Figura 11. Análisis. CU Gestionar Jugador. Diagrama de Colaboración. Escenario Incluir

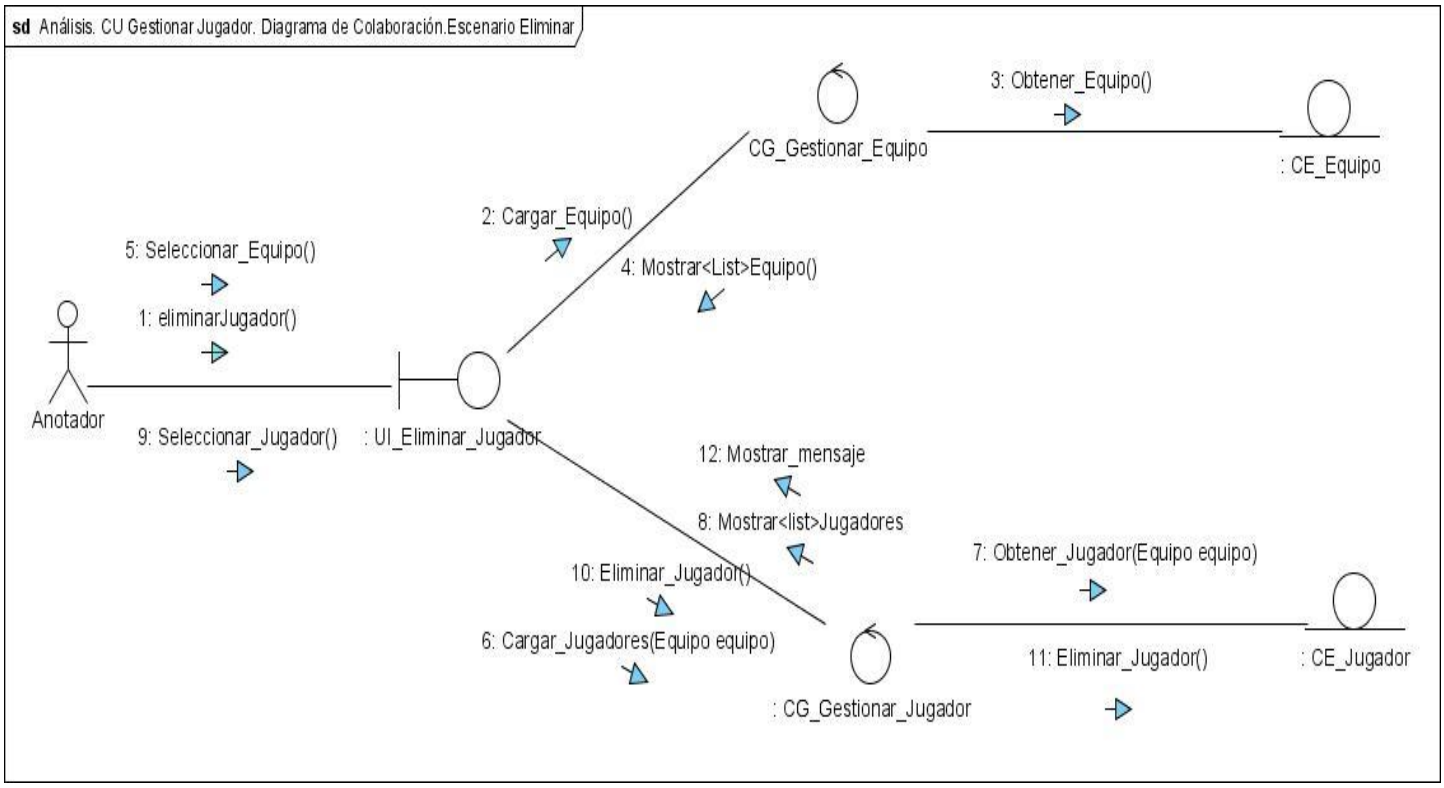


Figura 12. Análisis. CU Gestionar Jugador. Diagrama de Colaboración. Escenario Eliminar

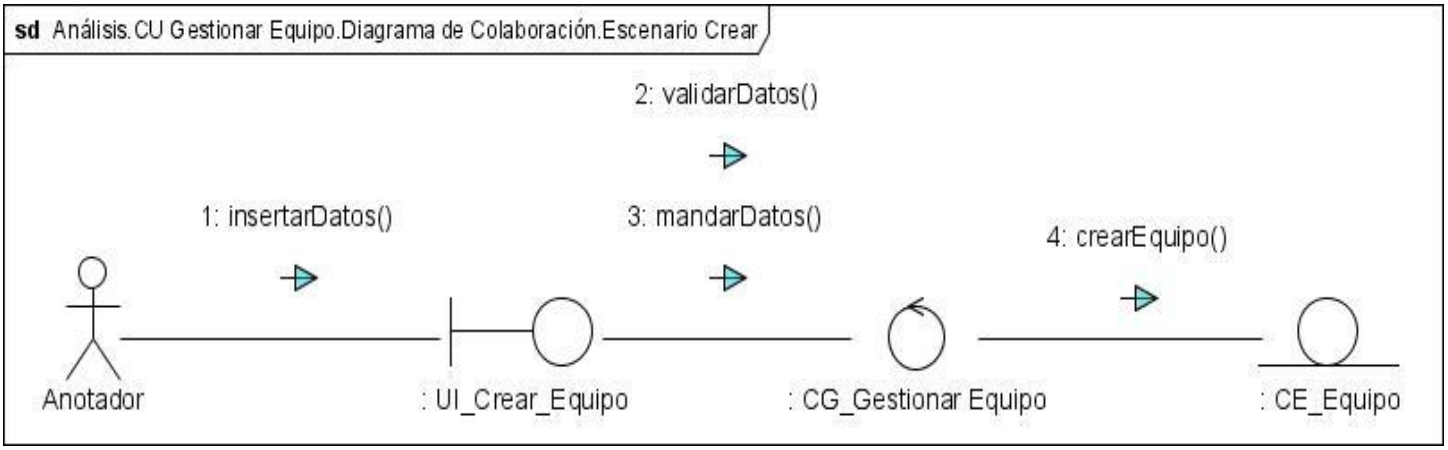


Figura 13. Análisis. CU Gestionar Equipo. Diagrama de Colaboración. Escenario Crear

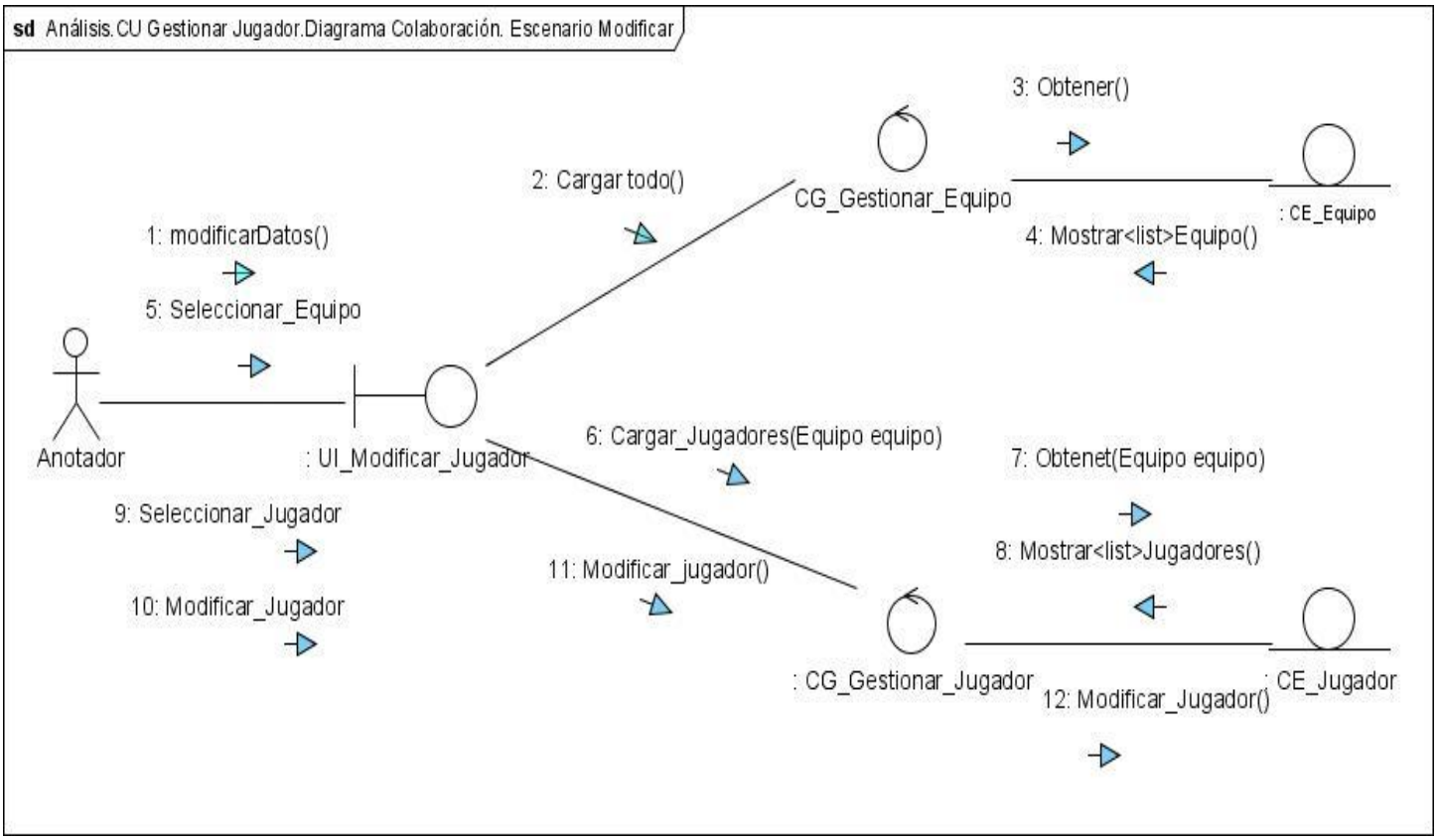


Figura 14. Análisis. CU Gestionar Jugador. Diagrama Colaboración. Escenario Modificar.

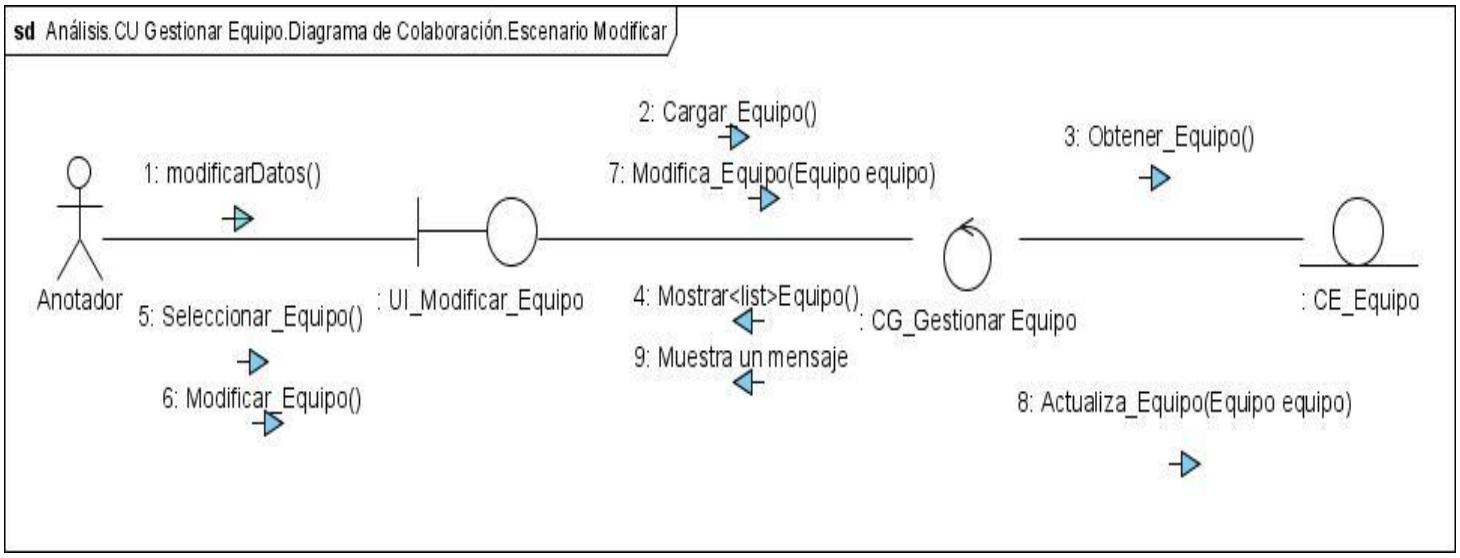


Figura 15. Análisis. CU Gestionar Equipo. Diagrama de Colaboración. Escenario Modificar.

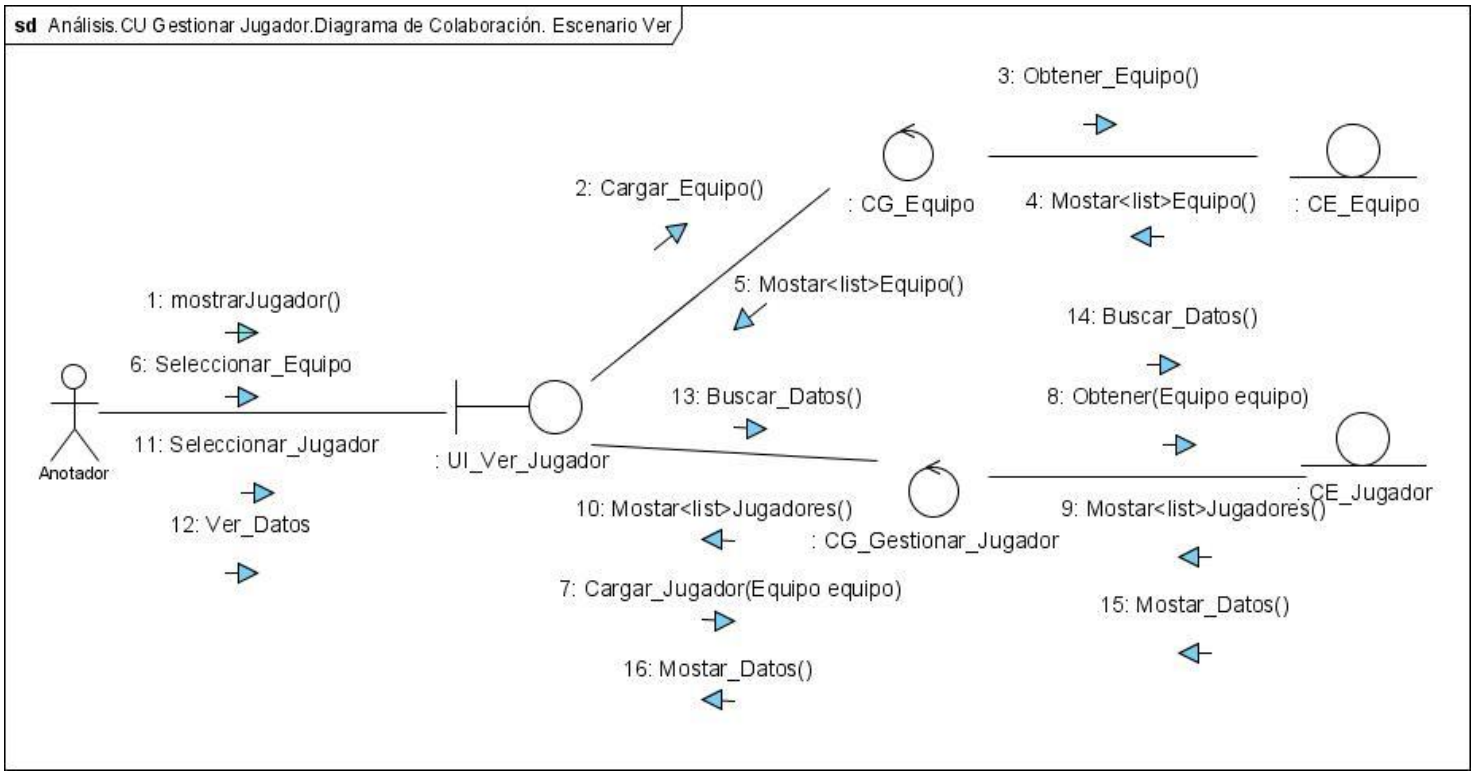


Figura 16. Análisis. CU Gestionar Jugador. Diagrama de Colaboración. Escenario Ver.

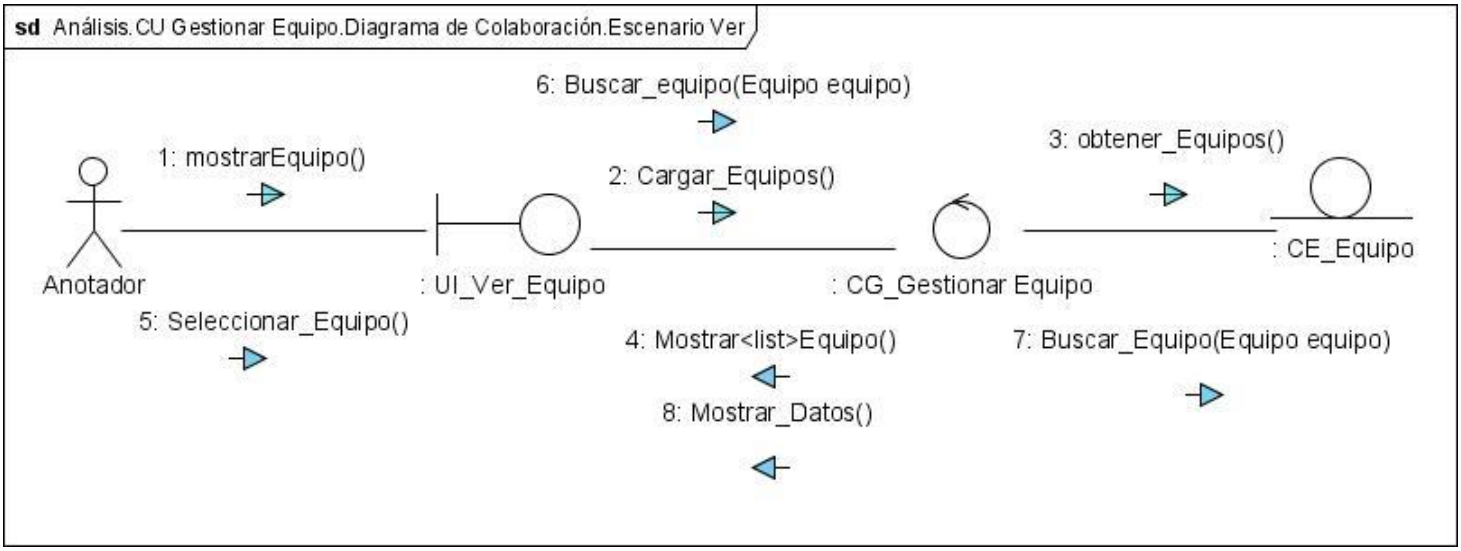


Figura 17. Análisis. CU Gestionar Equipo. Diagrama de Colaboración. Escenario Ver.

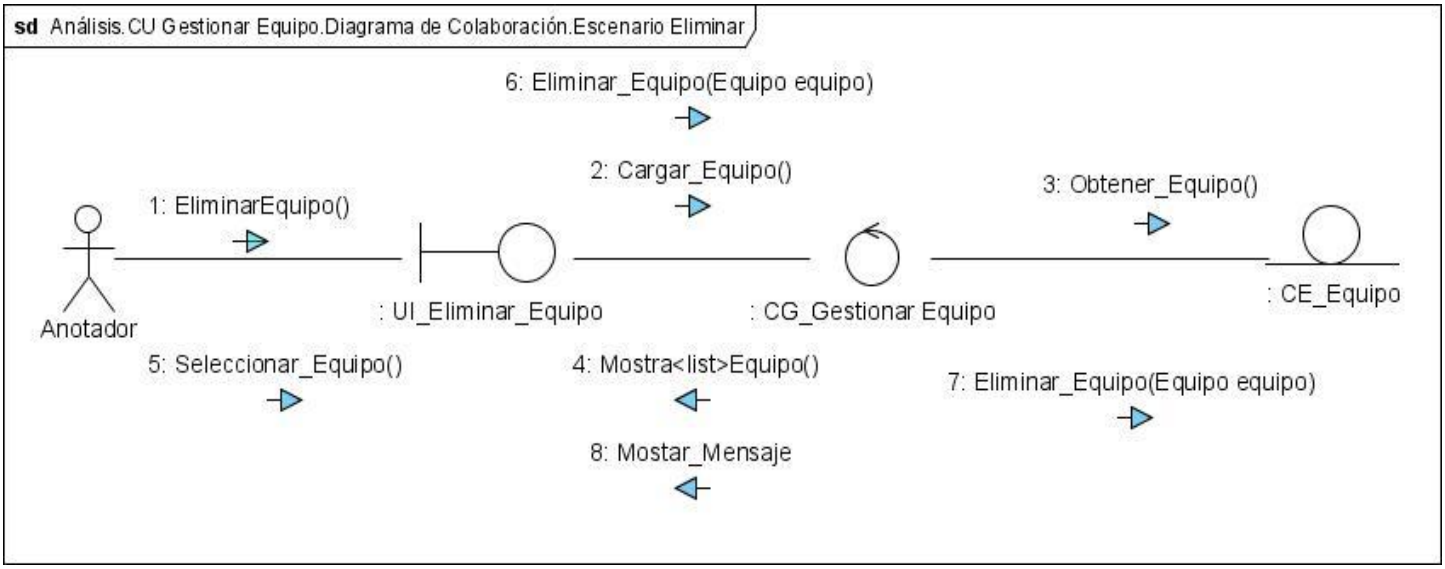


Figura 18. Análisis. CU Gestionar Equipo. Diagrama de Colaboración. Escenario Eliminar.

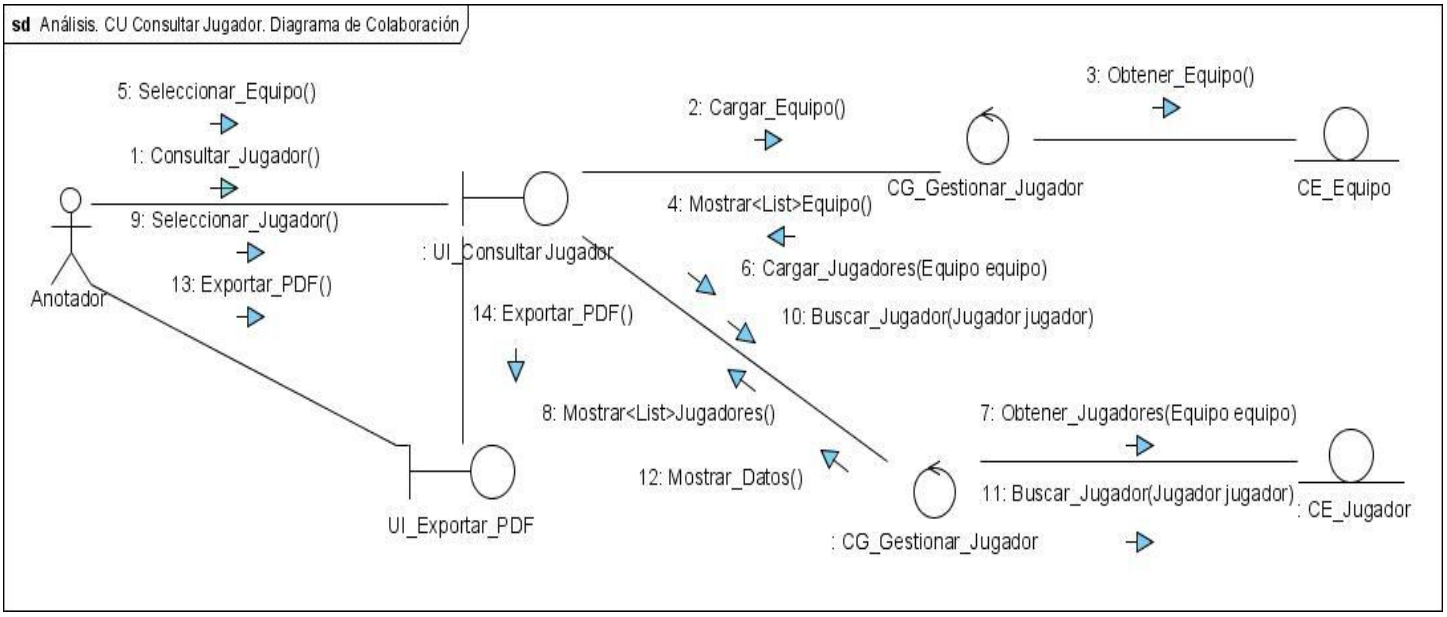


Figura 19. Análisis. CU Consultar Jugador. Diagrama de Colaboración.

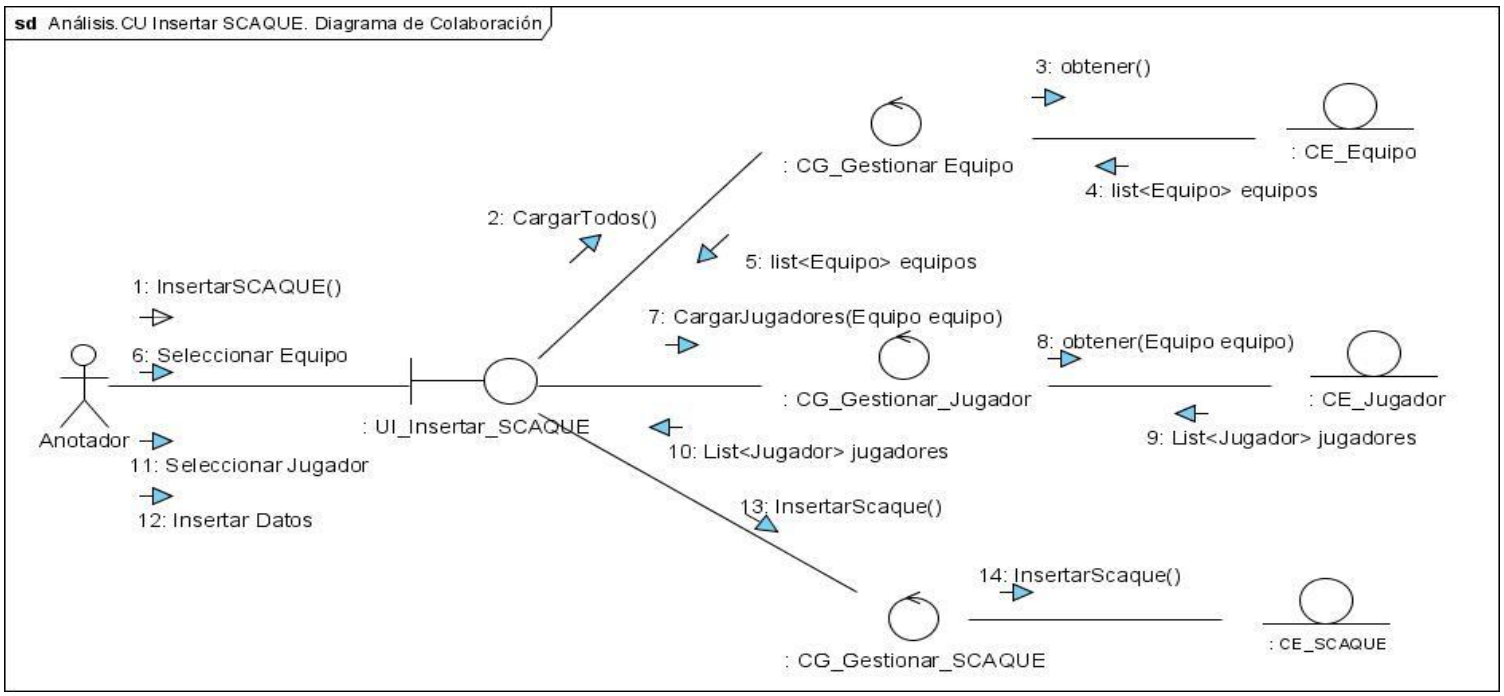


Figura 20. Análisis. CU Insertar SCAQUE. Diagrama de Colaboración.

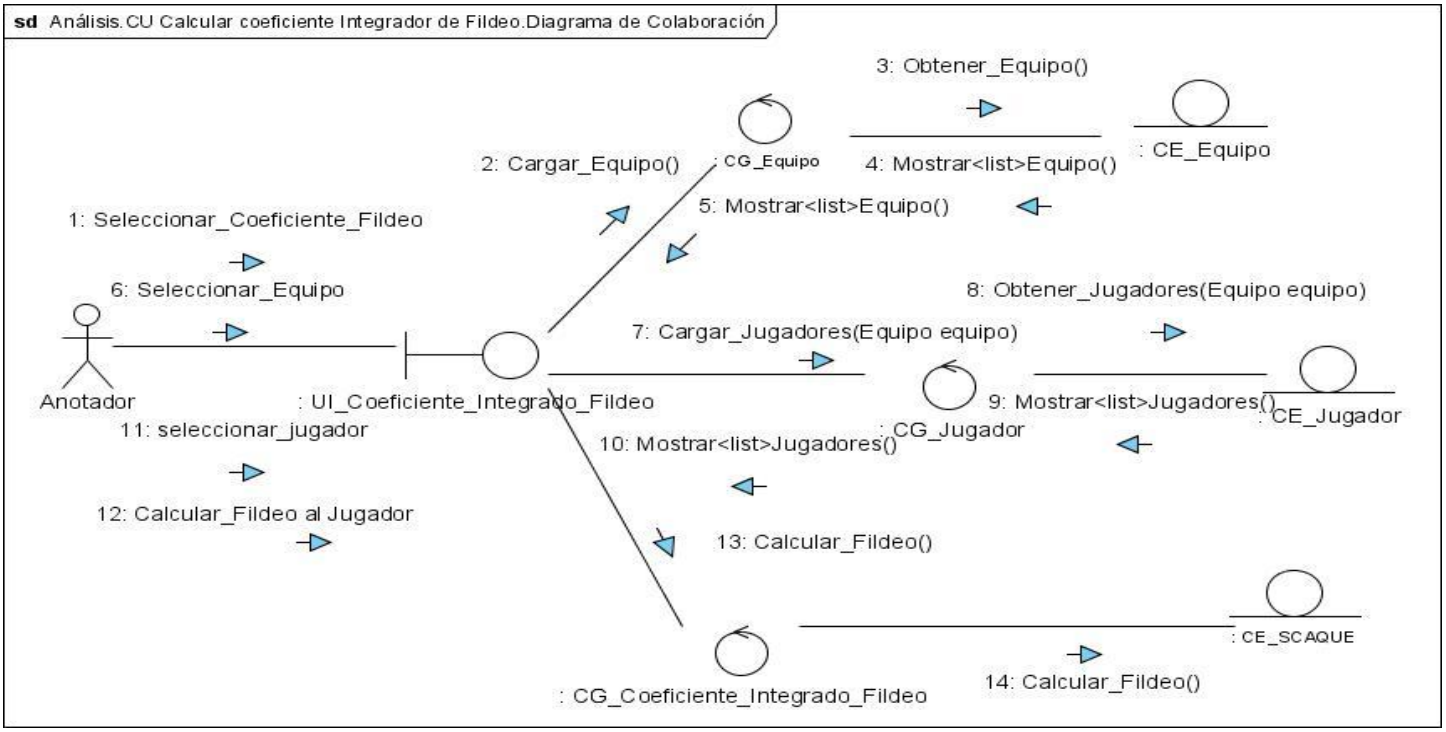


Figura 21. Análisis. CU Calcular coeficiente Integrador de Fildeo. Diagrama de Colaboración.

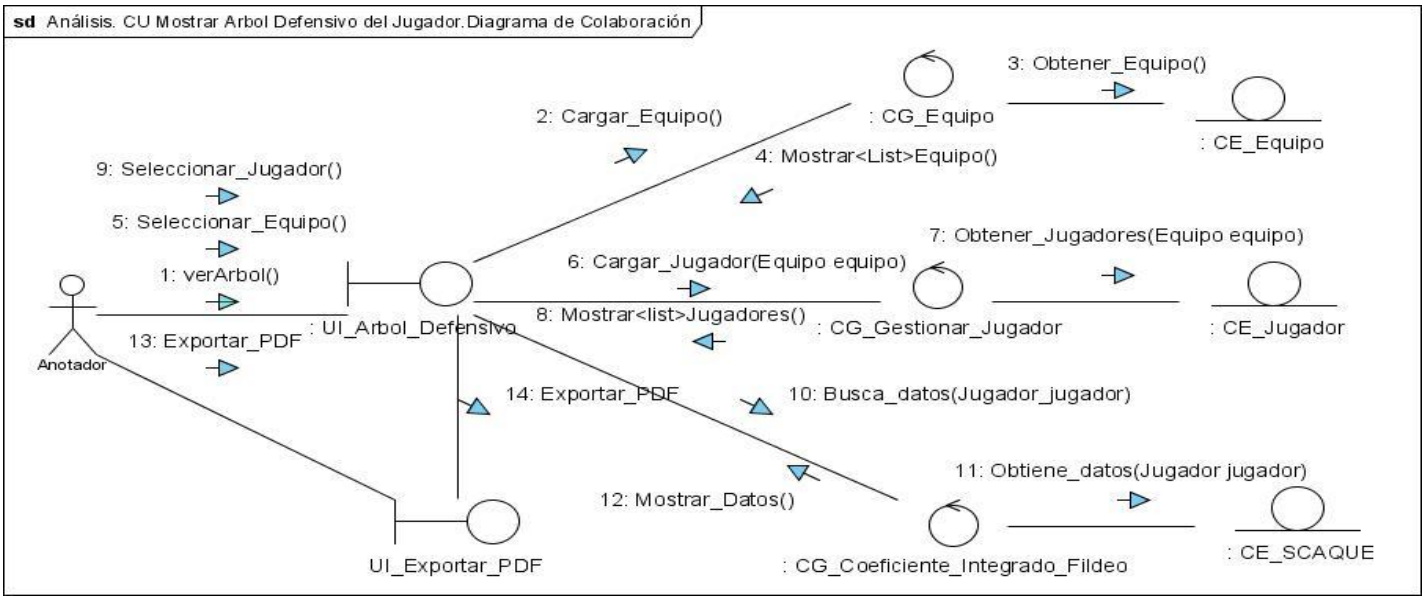


Figura 22. Análisis. CU Mostrar Árbol Defensivo del Jugador. Diagrama de Colaboración.

3.6 Diseño

El diseño consiste en transformar los requisitos tanto funcionales como no funcionales en un diseño de clases, viendo las relaciones e interacción entre ellas, y teniendo en cuenta en el proceso una arquitectura robusta que permita adaptar al sistema a un entorno de implementación donde se está trabajando y al mismo tiempo que permita la reutilización y la actualización de lo que se haga.

3.7 Arquitectura

La arquitectura es el esqueleto o base de una aplicación, en esta se analiza la aplicación desde varios puntos de vista que ayudan a caracterizar estructuralmente el sistema según su naturaleza y complejidad.

La arquitectura en capas define cómo organizar el modelo de diseño a través de capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, ayuda a identificar qué puede reutilizarse, y proporciona una estructura que ayuda a tomar decisiones sobre qué partes comprar y qué partes construir.

Existen tres capas distribuidas de la siguiente forma:

Capa de Aplicación

Esta capa contiene todas las clases de interfaz de usuarios que representan las pantallas de la aplicación que el usuario ve. Esta capa depende de la capa de Lógica de Negocio y de la capa de Acceso a Datos.

Capa de Lógica de Negocio

La capa de Lógica de Negocio tiene todas las clases controladoras que representan el comportamiento de la aplicación, según los casos de uso. Esta capa representa la frontera del cliente con la capa de Acceso a Datos. La capa de Lógica de Negocio depende de la capa de Acceso a Datos.

Capa de Acceso a Datos

La capa de Acceso a Datos apoya el acceso a un archivo XML.

En la siguiente figura se muestra cómo está estructurada la arquitectura en capas de la aplicación que se quiere lograr. Ver Figura.23

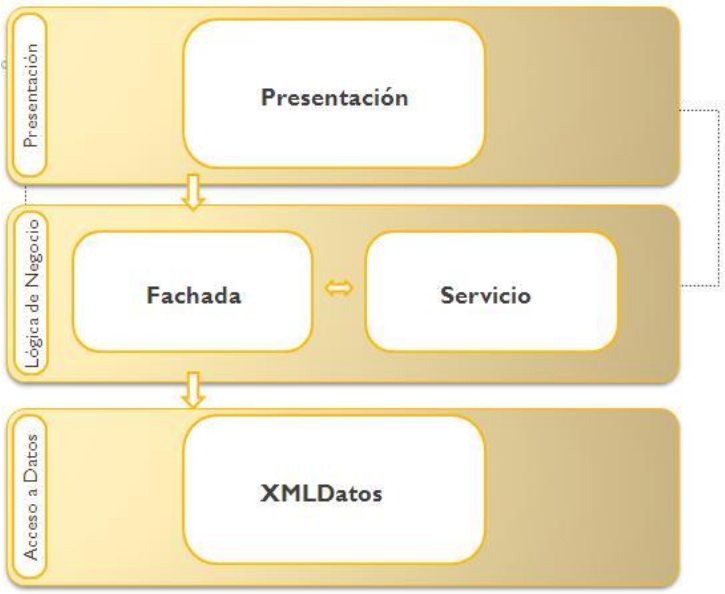


Figura 23. Arquitectura en capas.

3.8 Patrones de Diseño a utilizar

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

Un patrón de diseño es una solución a un problema de diseño, para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores; otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto

esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Se pueden destacar 5 patrones principales que son:

- **Experto**
- **Creador**
- **Alta Cohesión**
- **Bajo Acoplamiento**
- **Controlador**

3.9 Framework para la generación de reportes

Hoy en día existen diversos framework para la generación de reportes; pero como sucede siempre resaltan unos más que otros tal como lo son JFreeReport, CrystalReport, BIRT y JasperReport.

JasperReports.

JasperReports es ampliamente la presentación de reportes en Java. Presenta un acelerado desarrollo en esta especialidad, informes de alto rendimiento y escalabilidad masiva. JasperReports Professional incluye iReport que es un diseñador de informes gráfico muy fácil de usar y proporciona una cobertura completa de todas las capacidades de JasperReports. Los administradores y diseñadores de informes pueden apreciar la capacidad de utilizar el diseñador para obtener, almacenar y modificar los informes, es el más popular, potente y fácil de usar.

JasperReports en la actualidad es el más popular del mundo en sistemas Java de información gracias a las siguientes características:

- 100% puro Java, lo que asegura la portabilidad.
- Paneles, cuadros, y gráficos.
- Pixel-perfecto, complejos diseños de pantalla a imprimir.
- Flexibles y extensibles fuentes de datos, dando amplia gama de formatos de salida.
- De alto rendimiento.

- Solución abierta y basada en standards: Java y XML.
- Más grande del mundo, en cuanto a la comunidad más activa de diseñadores y desarrolladores de informes.

3.10 Diagramas de clases del diseño

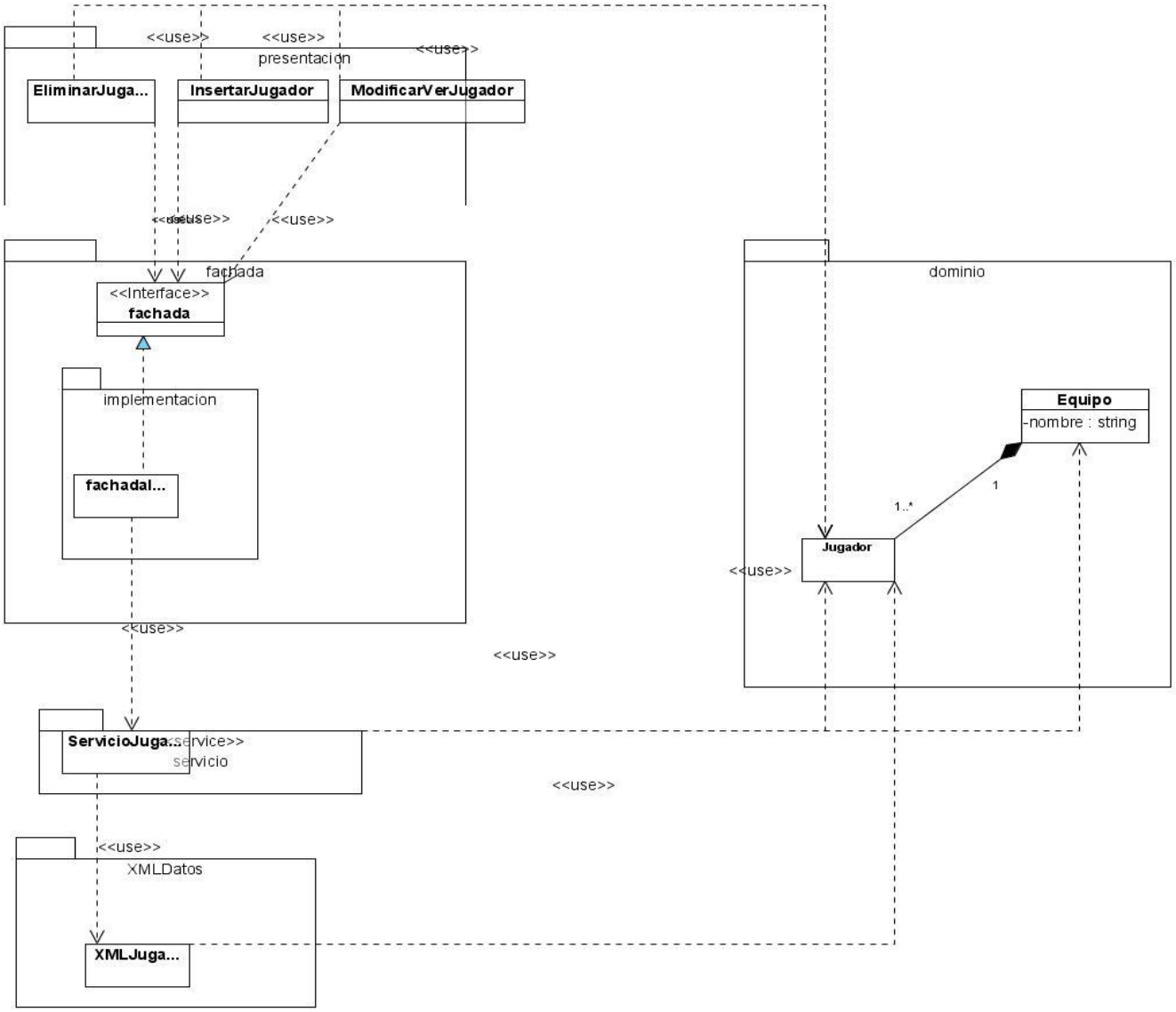


Figura 24. Diseño. CU Gestionar Jugador. Diagrama de Clases.

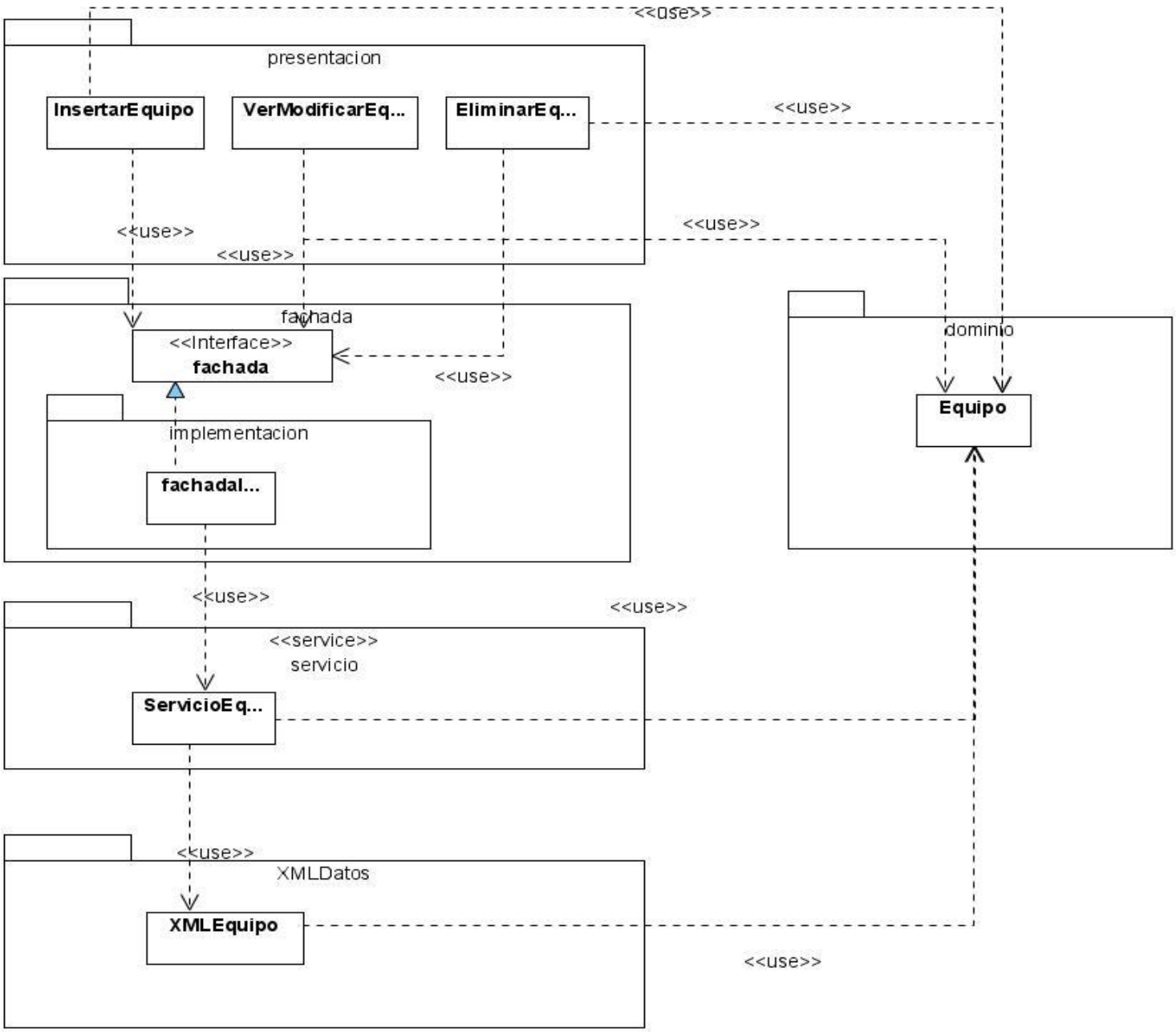


Figura 25. Diseño. CU Gestionar Equipo. Diagrama de Clases.

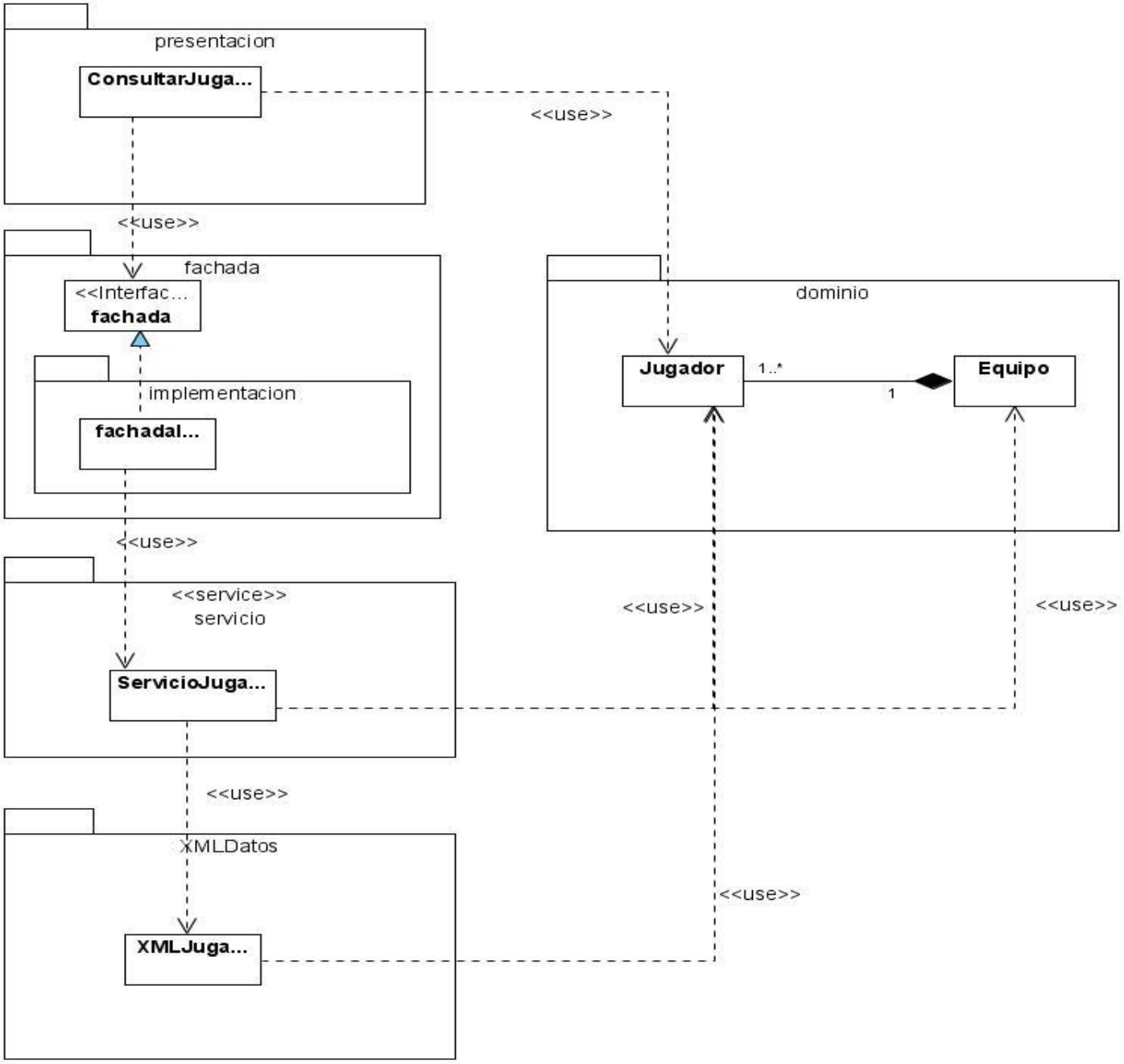


Figura 26. Diseño. CU Consultar Jugador. Diagrama de Clases.

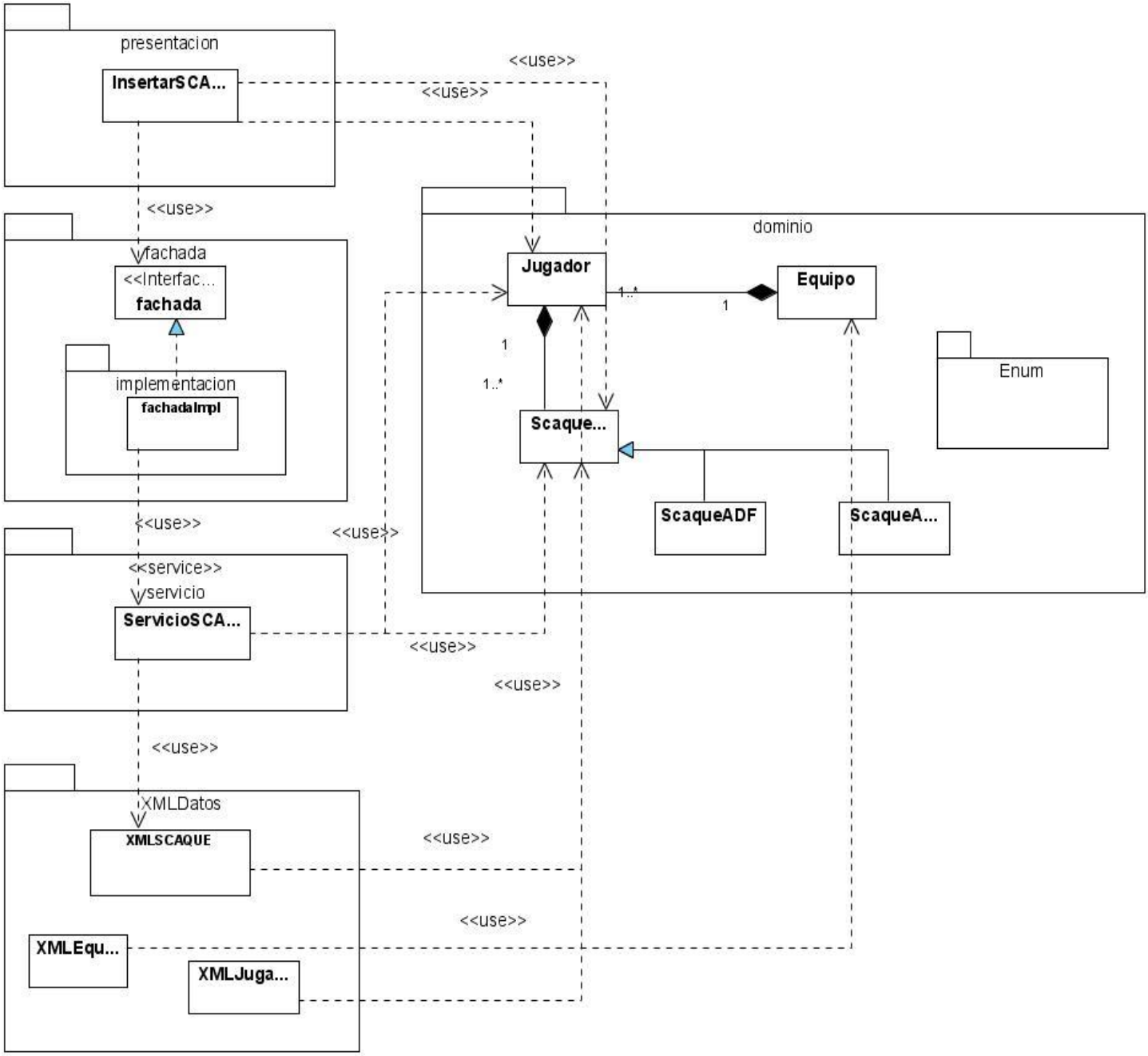


Figura 27. Diseño. CU Insertar SCAQUE. Diagrama de Clases.

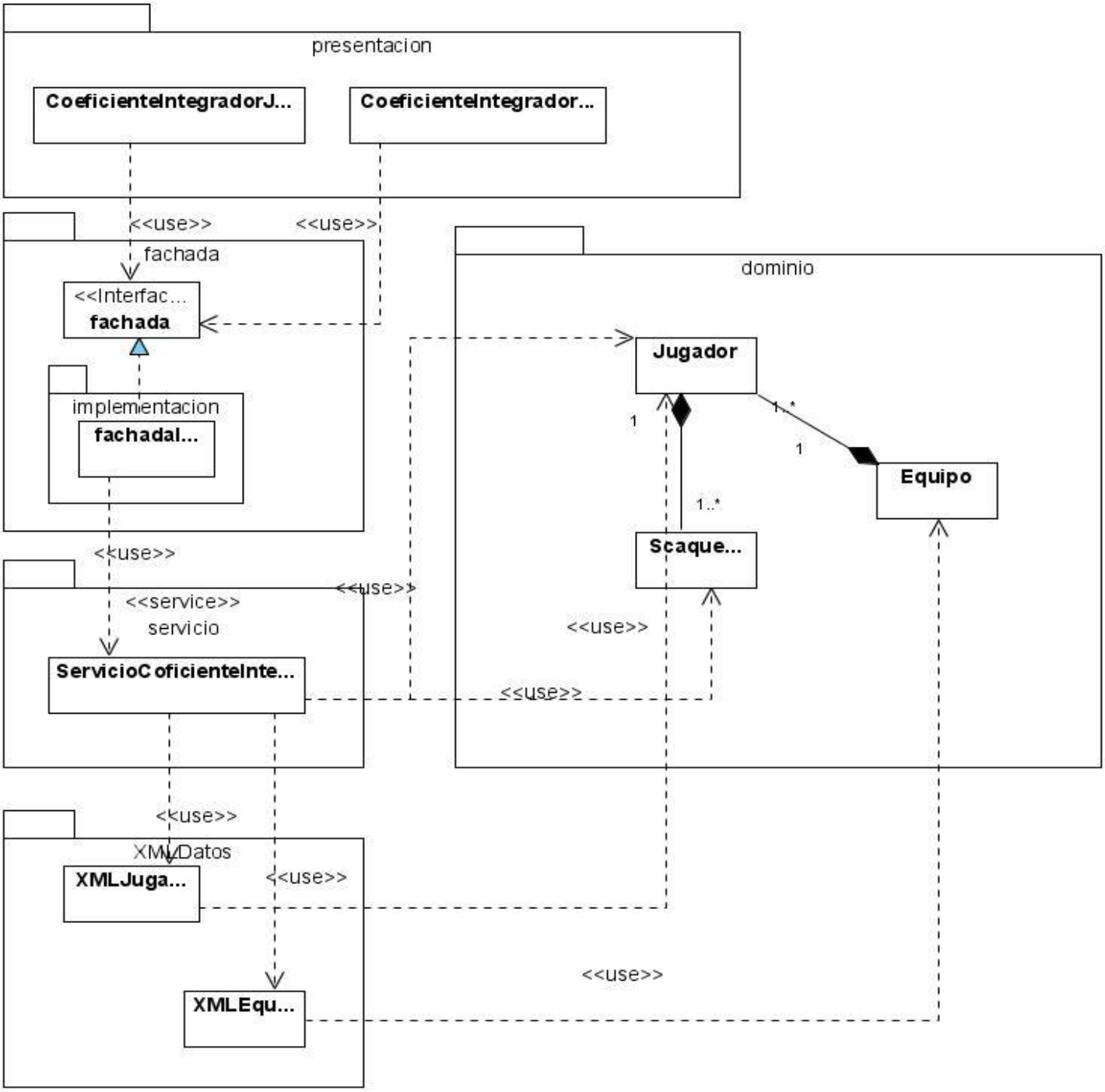


Figura 28.Diseño. CU Calcular Coeficiente Integrador de Fildeo. Diagrama de Clases.

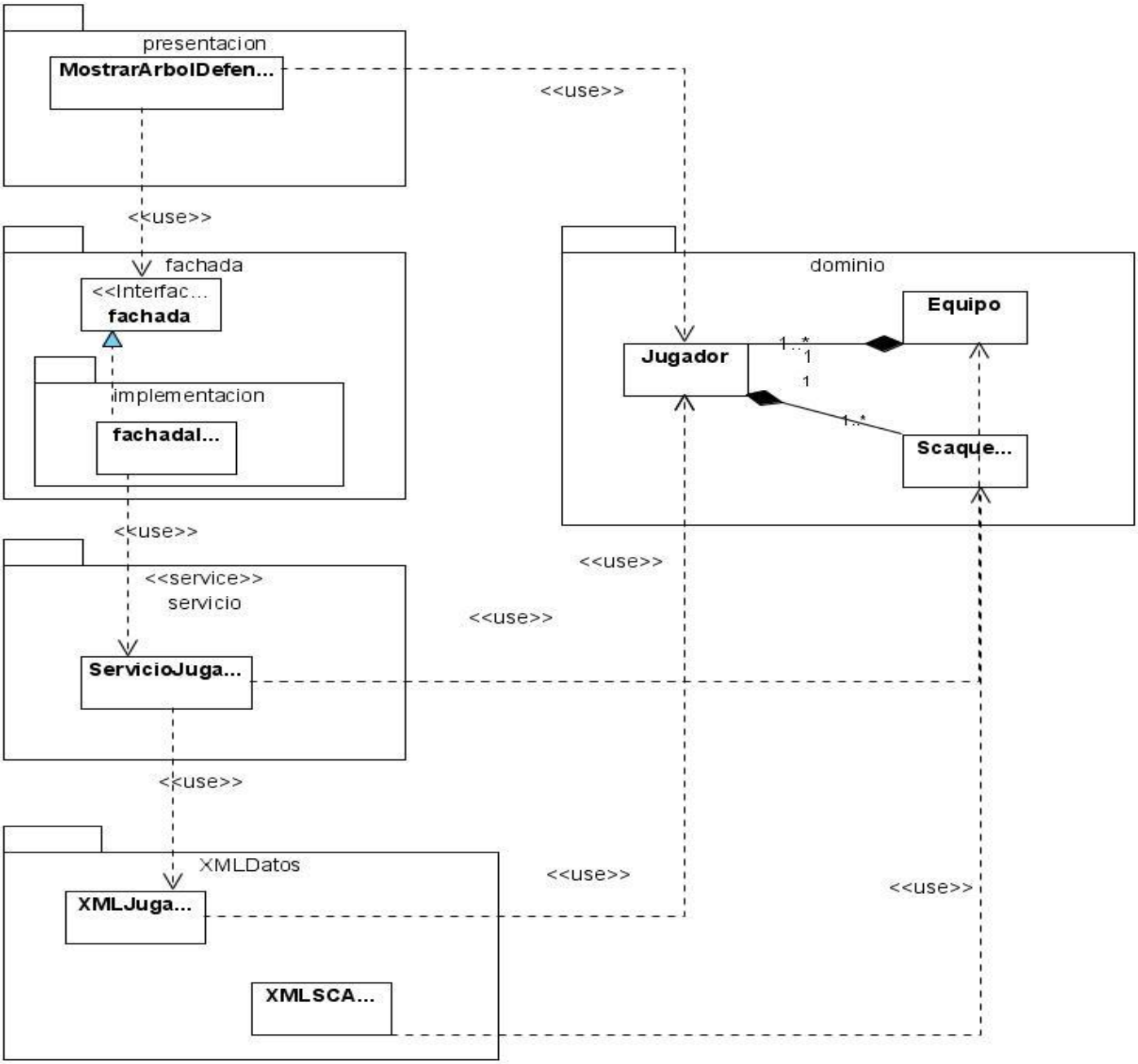


Figura 29. Diseño. CU Mostrar Árbol Defensivo. Diagrama de Clases.

Para un mejor entendimiento y vista de los paquetes a continuación se mostraran por las imágenes por paquetes.

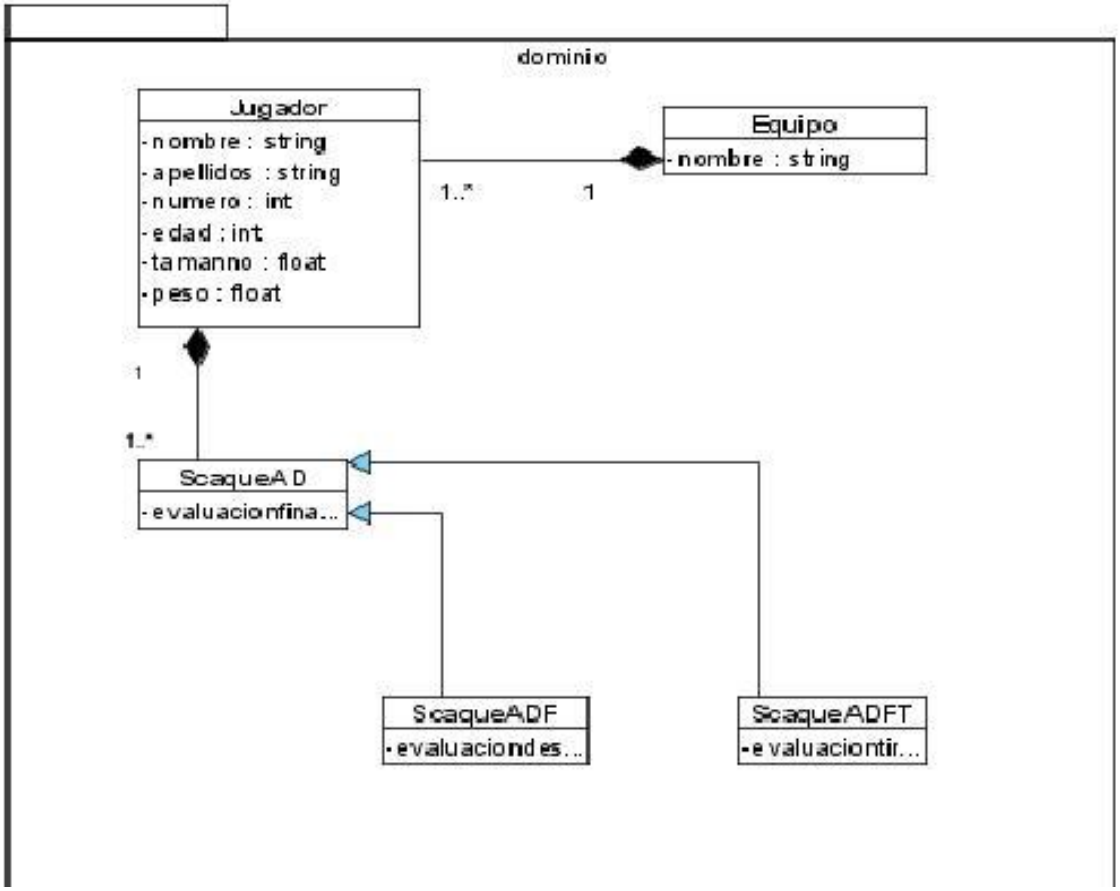


Figura 30. Diseño. Dominio.

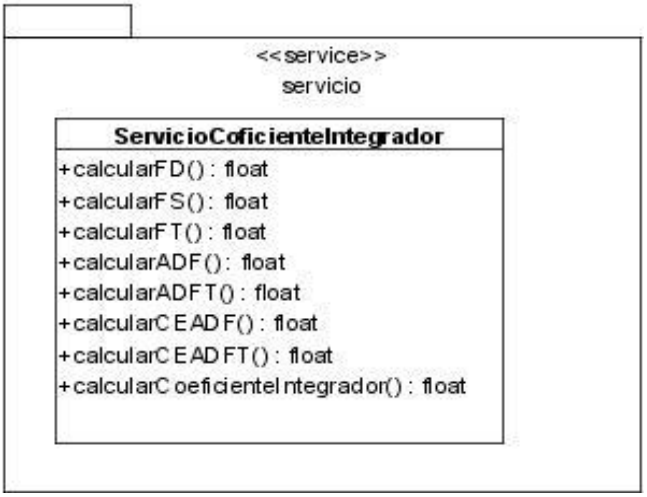


Figura 31. Diseño. Sevicio.Coeficiente_Integrador.

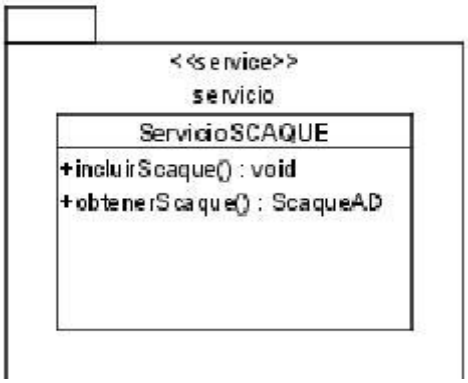


Figura 32. Diseño.Servicio.SCAQUE.

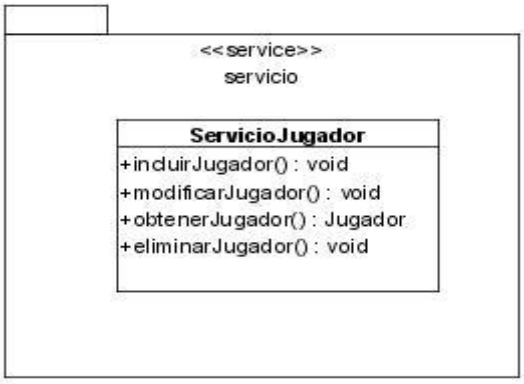


Figura 33. Diseño. Jugador.

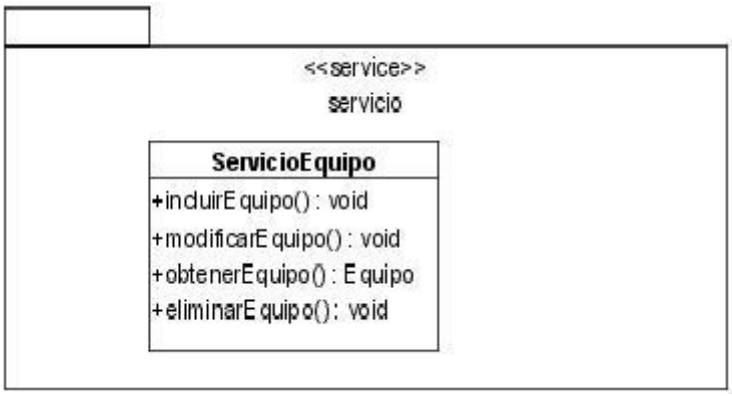


Figura 34. Diseño. Equipo.

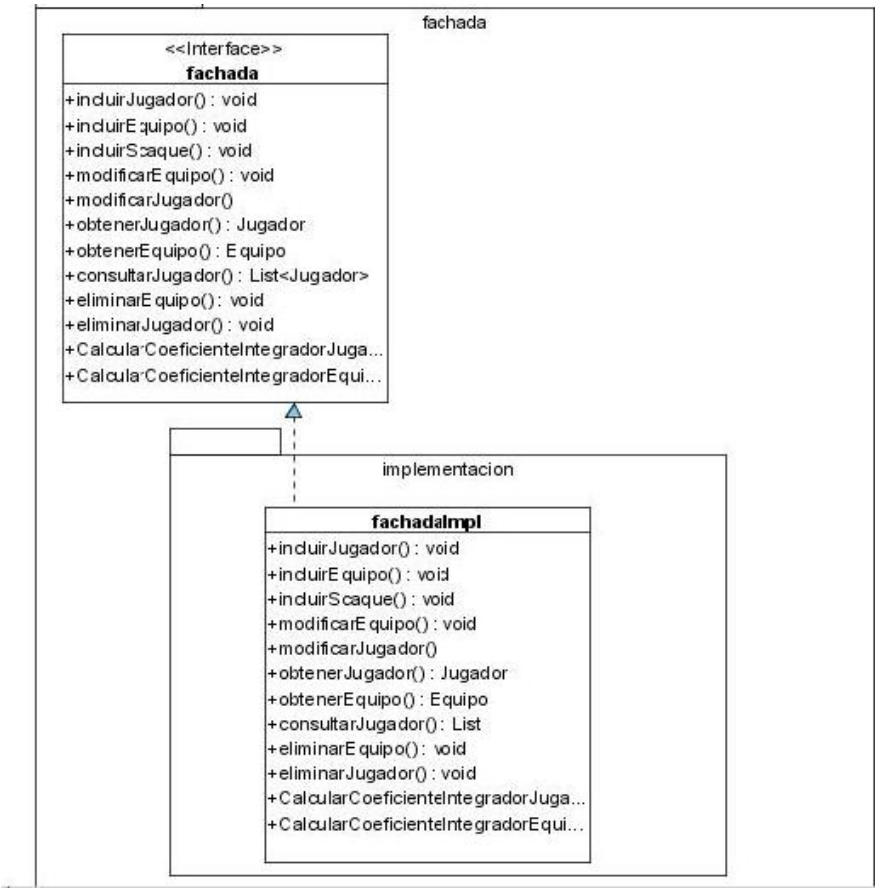


Figura 35.Diseño.Fachada.

3.11 Conclusiones

En este capítulo se realizaron los flujos de trabajo de análisis y diseño. Se desarrollaron los diagramas pertinentes en cada etapa, describiendo así el funcionamiento de cada caso de uso para su implementación.

Capítulo 4 Implementación

4.1 Introducción

En este capítulo se abordará el tema de la implementación, donde se hablará de los estándares de codificación, mostrando las dependencias entre las partes de código del sistema (diagrama de componentes), así como los diagramas de componentes que se utilizan para modelar la vista de implementación estática de un sistema.

4.2 Tratamientos de Excepciones

El tratamiento de errores posibilita el buen funcionamiento de una aplicación dándole una mejor apariencia ante los clientes. Para prevenir errores por parte del usuario, sólo se le brindan las opciones mínimas necesarias a la hora de efectuar cualquier operación, por ejemplo: se deshabilitan los botones, si el usuario no se ha loggeado. Una vez determinado su rol, se le habilitan las opciones a las cuales tiene acceso. Otro tipo de error que puede ocurrir son los que no pueden ser detectados en la parte del cliente, pues ocurren internamente en la aplicación. En este caso se muestra un mensaje de error donde se especifica el tipo de error claramente.

4.3.Estándares de la codificación

Resulta muy ventajoso utilizar un estándar para escribir código, pues se reducen considerablemente los errores, y los códigos resultan más compresibles y fáciles de leer. Con vistas a garantizar la homogeneidad de dicho código, se establece el estilo descrito a continuación:

4.3.1 Principios Generales

- Los nombres de cada uno de los elementos del programa deben ser significativos; su nombre debe explicar en lo posible el uso del elemento.

- La mayoría de los elementos se deben nombrar usando sustantivos (posiblemente compuestos), o formas verbales en imperativo.

- La forma de construir los nombres será colocando primero el verbo o el sustantivo, seguido de cada uno de sus complementos con la primera letra en mayúscula.

EJEMPLOS

Juego	es nombre sustantivo
numeroJugadas	es nombre sustantivo.
pilaVacia	es nombre sustantivo.
hagaJugada	es un nombre verbal en imperativo
almacenePartida	es un nombre verbal en imperativo

NOMBRAMIENTO DE ELEMENTOS

Elemento	Reglas de nombramiento
Clases, interfaces y archivos fuente	Nombre sustantivo singular, con la primera letra en mayúscula y las demás en minúsculas.
Constantes	Nombre sustantivo en mayúsculas. Para separar palabras se usará el guión bajo.
Atributos	Nombre sustantivo en minúsculas.
Métodos	Nombre sustantivo en minúsculas, si retorna un valor. Nombre verbal en minúsculas, si no retorna valor. Para los métodos analizadores y modificadores de atributos, alternativamente puede usarse el estándar getAtributo() y setAtributo(). Si se utiliza esta alternativa, debe usarse consistentemente en todos estos métodos.
Paquetes y directorios	Nombre sustantivo singular en minúsculas.

Estructura de Archivos

Todos los archivos fuente deben tener la siguiente estructura básica general:

```
//===================================================================
```

```
// ARCHIVO
// FECHA CREACIÓN:
// AUTOR:
// …. Comentarios generales
//=========================================================================

package xxxx;

//=========================================================================
//  BIBLIOTECAS REQUERIDAS
//=========================================================================

import xxx;

....
```

Cada clase o interfaz debe tener la siguiente estructura básica general:

```
//=========================================================================
//  CLASE NombreDeLaClase
//=========================================================================
….
//-------------------------------------------------------------------------
//  ATRIBUTOS DE INSTANCIA
//-------------------------------------------------------------------------
….
//-------------------------------------------------------------------------
//  ATRIBUTOS DE CLASE (ESTATICOS)
//-------------------------------------------------------------------------
….
//-------------------------------------------------------------------------
//  VALIDACIÓN DE INVARIANTE
//-------------------------------------------------------------------------
private bolean invarianteOk ()
//-------------------------------------------------------------------------
//  METODOS CONSTRUCTORES
//-------------------------------------------------------------------------
….
//-------------------------------------------------------------------------
//  MÉTODOS DE INSTANCIA
//-------------------------------------------------------------------------
….
```

```
//--------------------------------------------------------------------
//  METODOS DE CLASE (ESTATICOS)
//--------------------------------------------------------------------
….
//--------------------------------------------------------------------
//  INICIALIZADOR DE CLASE (ESTATICO)
//--------------------------------------------------------------------
….
```

Si alguna de las secciones no aparece, las líneas de encabezado pueden eliminarse.

Comentarios

GENERALIDADES

Se empleará el formalismo de precondición, poscondición e invariante. Éstas son afirmaciones sobre el estado antes y después de ejecutarse un método para las primeras dos; o durante toda la vida de un objeto para el último.

El código debe comentarse utilizando la sintaxis apropiada para uso de javadoc, teniendo en cuenta que para la producción de la documentación deben incorporarse los tags particulares que no hagan parte del estándar. En el caso de las poscondiciones, utilizaremos la notación prima (x') para referirnos al valor de una variable al finalizar el método, y sin prima (x) para su valor antes de iniciarse la ejecución de la rutina. Se asume que los atributos que no figuran en la poscondición, no sufren modificaciones. En el caso de funciones analizadoras, éstas no deben afectar atributos, y su poscondición se especifica con el tag @return de javadoc.

Comentarios de las Clases Interfaces

```
/**
 * Descripción de la clase
 * @author Nombre del desarrollador 1
 * @author Nombre del desarrollador 2
 * @version número versión y fecha
 * @inv Invariante de clase
*/
```

Comentarios de Atributos

```
/** Descripción del atributo */
```

Comentarios de Métodos Constructores

```
/**
  * Descripción de lo que hace el método
  * @param p Descripción del parámetro.
  * @pre precondición uno
  * @pre precondición dos
  * @pos poscondición uno
  * @pos poscondición uno
  * @exception TipoExcepción, Condiciones que causan la excepción
*/
```

Comentarios de Métodos analizadores

```
/**
  * Descripción de lo que hace el método
  * @param p Descripción del parámetro.
  * @pre precondición uno
  * @return valor a retornar
  * @exception TipoExcepción, Condiciones que causan la excepción
*/
```

Comentarios de Métodos Modificadores

```
/**
  * Descripción de lo que hace el método
  * @param p Descripción del parámetro.
  * @pre precondición uno
  * @pos poscondición uno
  * @exception TipoExcepción, Condiciones que causan la excepción
*/
```

SANGRÍA Y ABLOCAMIENTO

Todos los archivos fuentes deben seguir el estándar de sangría. Cada entrada corresponde a 4 espacios. No debe usarse el carácter de tabulación, pues es dependiente de la configuración del editor. Todos los

constructores que admiten bloques de instrucciones delimitados por {…}, deben usarlos aún si estos tienen una sola instrucción.

Las funciones deben tener un solo punto de retorno, y el flujo normal de las estructuras de iteración no debe alterarse mediante instrucciones break o continúe.

Definición de Clases o Interfaces

```
......... {
    ... ..... ;
    ... ..... ;
    ... ..... ;
    ... ..... ;
} ......  ;
```

Estructuras Condicionales

```
if ( ..... ) {
    ..... ;
    ..... ;
    ..... ;
} else if ( ..... ) {
    ..... ;
    ..... ;
    ..... ;
} else {
    ..... ;
    ..... ;
    ..... ;
}
```

```
switch ( ..... ) {
    case ..... :
        ..... ;
        ..... ;
        ..... ;
    break;
    case ..... :
        ..... ;
        ..... ;
        ..... ;
```

```
    break;
    case ..... :
        ..... ;
        ..... ;
        ..... ;
   default:
        ..... ;
        ..... ;
        ..... ;
}
```

Las proposiciones break siempre deben estar en el nivel de case.

ESTRUCTURAS DE ITERACIÓN

```
while ( ..... ) {
    ..... ;
    ..... ;
    ..... ;
}
```

```
for ( ..... ; ..... ; ..... ) {
    ..... ;
    ..... ;
    ..... ;
}
```

```
do {
    ..... ;
    ..... ;
    ..... ;
} while ( ..... ) ;
```

ESTRUCTURAS DE EXCEPCIÓN

```
try {
   .....;
   .....;
} catch (....) {
```

```
   ....;
} catch (....) {
   ....;
} finally {
   .... ;
}
```

4.4 Diagrama de Componentes

El diagrama de componente da una visión de los que se está programando, el mismo muestra los distintos componentes agrupados por paquetes. A continuación se ilustra el diagrama de componentes referente a nuestro sistema.

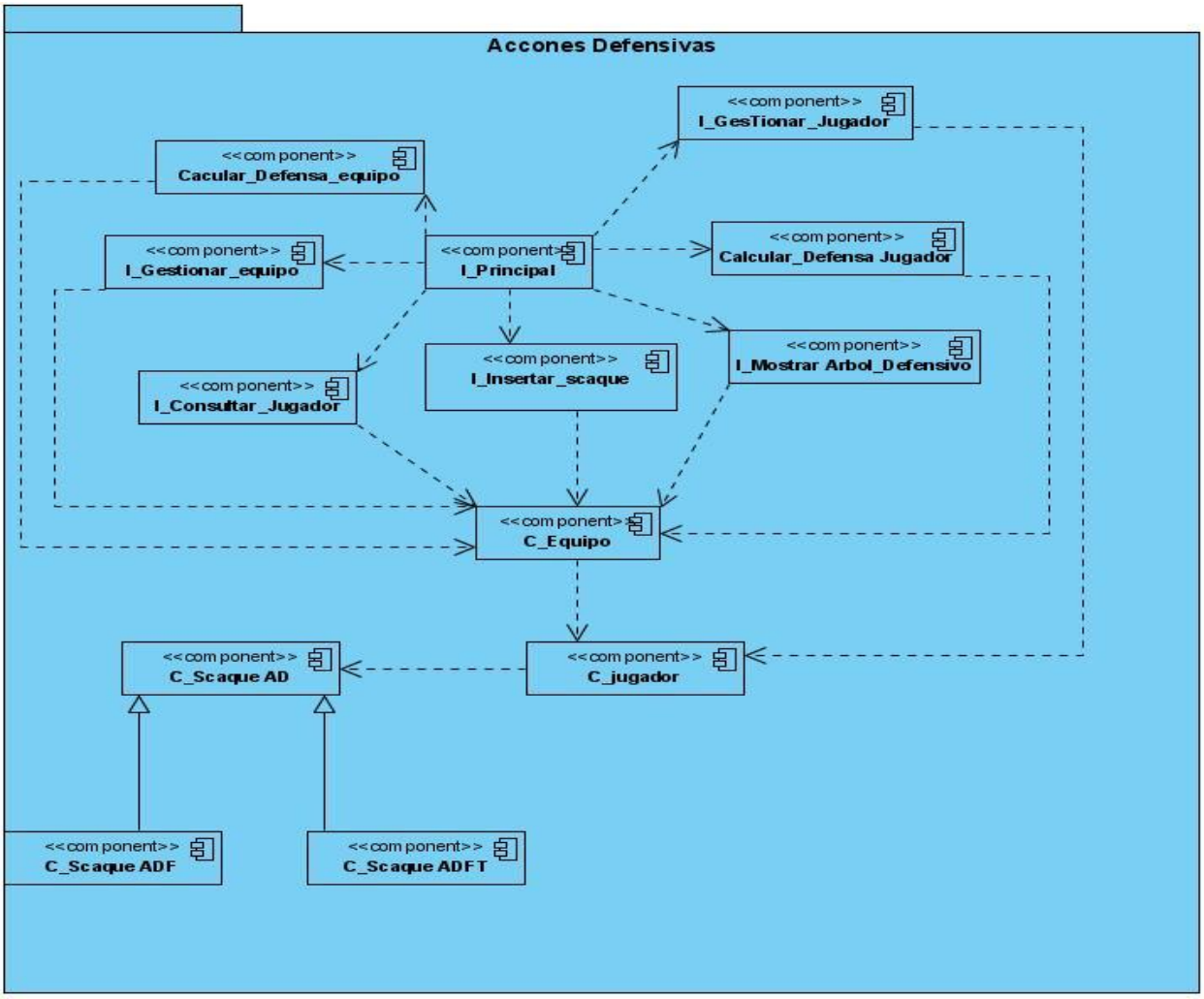


Figura 36. Diagrama de Componentes.

4.5 Conclusiones

En este capítulo se presentaron los diferentes tipos de estándares de codificación que ayudará a un mejor entendimiento con respecto al código. Quedó plasmado el diagrama de componentes representando las clases e interfaces que están presentes en la aplicación.

Conclusiones Generales

Con el desarrollo del Sistema Calidad y Efectividad Defensiva para los jugadores de la Línea Central en el Béisbol (CADEBA), se da cumplimiento a los objetivos de este trabajo, pues da camino a la obtención de una aplicación en el que se aplican los resultados de todo el proceso investigativo realizado a lo largo de las etapas del proyecto, lográndose:

- Un mecanismo de autenticación de usuarios para identificarlos y establecer el control de acceso a la información, así como a las diferentes operaciones a las que puede acceder el usuario.
- La creación de un sistema usable, funcional y accesible para los usuarios que mejora las condiciones del software existente.

Recomendaciones

De acuerdo a los resultados de todo el proceso de investigación realizado y basados en la experiencia acumulada se proponen las siguientes recomendaciones:

- Ampliar las funcionalidades del Sistema CADEBA de manera que se pueda realizar la codificación de las jugadas que hoy por hoy no pueden ser recogidas en el béisbol cubano.
- Desarrollar la implementación de un módulo para la televisión que visualice la mayor cantidad de información posible al usuario, utilizando como base para su implementación el sistema confeccionado.

Bibliografías

1. **Craig Larman. 1999.** *UML y Patrones. Introducción al Análisis y Diseño orientado a objetos.* México: Prentice Hall, 1999. 970-17-0261-1
2. **Introducción al lenguaje XML.** Geneura. [En línea] http://geneura.ugr.es
3. **Jaime E. Villate.** Introducción al XML. quark. [En línea] [Citado el: 4 de 3 de 2008.] www.quark.fe.up.pt.
4. **vico open modeling, s.l.** Vico.org: Open Modeling. Vico.org. [En línea] 2008. [Citado el: 20 de Enero de 2009.] http://www.vico.org.
5. **V P Company.** Visual Parading. [Online] 2002[Cited: Mayo 4, 2008] http://www.visualparadigm.com.
6. **Desarrollo de aplicaciones en C#.** http://geneura.ugr.es/CUR/C/
7. **Álvarez, Sofía, Hernández Anaisa.** Metodología para el desarrollo de aplicaciones con tecnología Orientada a Objetos utilizando notación UML. La Habana, 2000.
8. **Jacobson, Ivar; Booch, Grady y Rumbaugh, James.** El Proceso Unificado de Desarrollo de Software. Editorial Félix Varela, La Haba (1)na, 2004.
9. **Barrientos Enríquez, Aleida Mirian. El proceso Unificado de Modelado (RUP).** http://www.monografias.com/trabajos16/lenguaje-modelado-unificado/lenguaje-modelado-unificado.shtml#PROCESO (Noviembre-2005)
10. **Visual Paradigm. Visual Paradigm.** http://www.visual-paradigm.com/.
11. **http://www.sld.cu/libros/distancia/cap1.html.**
12. **http://www.netbeans.org/kb/trails/platform.html.**

Referencias bibliográficas

1. *Sparx.* [Online] [Cited: Febrero 20, 2009.] http://www.sparxsystems.com.ar/products/ea.html..

2. Trabajadores. *Trabajadores.* [Online] Trabajadores. Órgano de la Central de Trabajadores de Cuba., Febrero 01, 2009. [Cited: Febrero 15, 2009.] http://www.trabajadores.cu/.

3. **Innova, Grupo Soluciones.** GSInnova. [Online] 2007. [Cited: febrero 20, 2009.] http://www.rational.com.ar.

4. **Pérez, Agustin.** Scielo. *Scielo.* [Online] 2006. [Cited: Febrero 15, 2009.] http://www.scielo.cl.

5. **Schumller, J.** Aprendiendo UML en 24 horas. México : Pearson Educación, 2000.

6. IBM Rational. [Online] [Cited: Febrero 21, 2009.] http://www.rational.com.

7. **Company, VP.** Visual Parading. [Online] 2002. [Cited: Febrero 21, 2009.] http://www.visualparadigm.com.

8. JavaHispano. [Online] 2002. [Cited: Febrero 22, 2009.] http://www.javahispano.org.

9. **Corporation, Microsoft.** msdn. [Online] 2009. [Cited: Febrero 22, 2009.] http://msdn.microsoft.com.

10. Curso XML. [Online] [Cited: Febrero 22, 2009.] http://www.geneura.ugr.es.