ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DEL SISTEMA DE INFORMACIÓN DE BIENESTAR UNIVERSITARIO, PARA LA ADMINISTRACIÓN DE LOS IMPLEMENTOS Y ESPACIOS FÍSICOS DEPORTIVOS EN LA UNIVERSIDAD DEL MAGDALENA

HÉCTOR ÁLVARO MARTÍNEZ BAQUERO
EDUARD JAVIER PALACIOS MENDOZA

UNIVERSIDAD DEL MAGDALENA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
SANTA MARTA D.T.C.H.
2008

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN MÓDULO DEL SISTEMA DE INFORMACIÓN DE BIENESTAR UNIVERSITARIO, PARA LA ADMINISTRACIÓN DE LOS IMPLEMENTOS Y ESPACIOS FÍSICOS DEPORTIVOS EN LA UNIVERSIDAD DEL MAGDALENA

Proyecto de Memoria de Grado

HÉCTOR ÁLVARO MARTÍNEZ BAQUERO
EDUARD JAVIER PALACIOS MENDOZA

Director de Tesis

MARIA DEL PILAR SALES
Ingeniera de Sistemas

UNIVERSIDAD DEL MAGDALENA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
SANTA MARTA D.T.C.H.
2008

Nota de aceptación

_____

_____

_____

_____
Presidente del Jurado

_____
Jurado

_____
Jurado

Santa Marta, Marzo 2008

DEDICATORIAS

A Dios porque es en él que pongo toda mi confianza y me guía y me ilumina en todas y cada una de mis decisiones.

A mis padres, Claire Isabel Mendoza y Gilberto Palacio quienes gracia a su esfuerzo, dedicación y educación han hecho de mi una persona de bien y responsable de sus acciones.

A mis hermanos Juan, Mónica, Claire y Soraya quienes han sido mi apoyo y mi inspiración.

A mis tíos, tías, y primos por enseñarme el valor y el significado de la frase "Unión Familiar".

A las personas que de una u otra manera han llegado a mi existencia y me han dejado alguna enseñanza que ha enriquecido mi saber.

Y a todos mis amigos, los cuales han compartido junto a mi tantas alegrías y hemos sobrellevado las tristezas y salido adelante.

Eduard Javier Palacios Mendoza

A Dios.

A mis padres Jaime Martínez y Uldis Baquero.

A mis hermanos Jaime Alberto y Jaime Enrique.

A mi abuela Emilia Villar.

A toda mi familia.

A mis amigos.

Héctor Álvaro Martínez Baquero

AGRADECIMIENTOS

Los autores expresan sus sinceros agradecimientos a aquellas personas que de alguna u otra manera nos han inspirado a alcanzar nuestras metas:

A nuestros padres

Gilberto Palacio y Claire Isabel Mendoza.

Jaime Martínez y Uldis Baquero.

A nuestros hermanos y amigos

Juan, Mónica, Claire, Soraya, Jaime Alberto, Jaime Enrique, Kelly, Luz Dary, Milena, José, Carlos, Rodnel, Diego, Vilma, Mariano, Oscar.

A la Ing. Maria del Pilar Sales.

A los Ingenieros Inés Meriño y Julio Alcázar.

A la Universidad del Magdalena, Profesores del Programa de Ingeniería de Sistemas y a nuestros compañeros de estudio.

CONTENIDO

Pág.

LISTA DE TABLAS

LISTA DE FIGURAS

Pág.

INTRODUCCIÓN

El deporte es una de las necesidades para el desarrollo en la vida del ser humano, ya que es una forma sana de aprovechar el tiempo libre, evita el sedentarismo, fomenta las relaciones interpersonales a través del deporte en grupos, mejora la condición física general, evita o previene enfermedades y desarrolla el sentido de autoestima y superación personal.

La Universidad del Magdalena cuenta con una infraestructura física ideal para el desarrollo y fomento de todas las disciplinas deportivas, entre los cuales se encuentran: estadio de softbol, estadio de fútbol con pista atlética y cancha alterna, canchas de fútbol de salón, tenis y baloncesto, además se cuenta con una dotación de implementos (balones de fútbol, balones de baloncesto, balones de voleibol, balones de fútbol sala, raquetas y pelotas de tenis, bates y pelotas de softbol, raquetas y pelotas de ping pong, entre otros.), para el servicio de toda la comunidad universitaria.

El Análisis, Diseño e Implementación de un Módulo del Sistema de Información de Bienestar Universitario, para la Administración de los Implementos y Espacios Físicos Deportivos en la Universidad del Magdalena es un aporte informático con el fin de mejorar la calidad en el manejo de préstamos y reservas de los escenarios e implementos deportivos que pertenecen a la Universidad del Magdalena.

Para el área de deporte de la Universidad del Magdalena, quien está a cargo del manejo de los préstamos y reserva de escenarios e implementos deportivos, se hace necesaria una herramienta que le ayude a manejar de forma adecuada y precisa toda la información referente a los procesos mencionados anteriormente, como información acerca de los escenarios e implementos deportivos, entre los cuales se encuentra, horarios de reservas y préstamos, nombres, cantidad de existencias por cada implemento, estado (ocupado o desocupado) de los escenarios y los datos personales de los estudiantes que solicitan este servicio, es por esto que surge este proyecto, con el cual se creará un módulo para el manejo de esta información.

1. PLANTEAMIENTO Y FORMULACIÓN DEL PROBLEMA

A mediados de la década del 50, cuando las computadoras hicieron su aparición en las instituciones, surgió la necesidad de organizar de forma eficiente los procesos que se llevaban a cabo. Desde entonces, las soluciones computacionales irrumpieron en el mundo empresarial y educativo como una herramienta poderosa para facilitar la realización de las tareas fundamentales dentro de las organizaciones, tales como nóminas, informes, presentaciones y entre otras.

La Universidad del Magdalena, institución de estudios superiores, teniendo como principal responsabilidad el desarrollo social, cultural, económico, artístico y tecnológico en la comunidad magdalenense y en pro del mejoramiento de los servicios que presta a la comunidad, ha orientado sus esfuerzos hacia la modernización y reestructuración de las actividades que realizan cada una de las dependencias que la conforman; en este proceso, la implementación de tecnologías de información ha jugado un papel sumamente importante debido a las grandes ventajas que ofrece en el manejo de un recurso tan valioso como lo es la información.

De acuerdo al análisis de las situaciones actuales que enfrenta la Universidad del Magdalena y teniendo en cuenta el crecimiento progresivo en cuanto al número de programas, estudiantes y docentes vinculados a la institución (quienes constituyen los principales usuarios de los servicios ofrecidos por la dependencia de bienestar estudiantil), la Coordinación de Deportes, en su afán de mejorar los servicios que presta a la comunidad universitaria, ha expresado la necesidad de implementar una aplicación que facilite tanto el manejo de los recursos deportivos (implementos y espacios físicos) con los que se cuenta, como la sistematización de las actividades de tipo operativo que le son propias tales como: Los préstamos y reservas de implementos y espacios físicos deportivos, programar actividades deportivas, informes estadísticos de los programas que muestra tendencias al deporte y balances de la existencias de los implementos deportivos.

La Coordinación de Deportes, a través del tiempo que lleva funcionando, ha desarrollado su labor básicamente en forma manual; sin embargo, en los últimos tiempos viene implementando plantillas elaboradas en Excel; sin convertirse esto en una solución a todas sus necesidades.

El préstamo de los implementos deportivos se hace personalmente y de forma inmediata, dejando como único requisito el carné que lo acredita como estudiante de la Universidad del Magdalena hasta que devuelva el objeto prestado, imposibilitando al estudiante que haga uso de otras actividades que solo puede hacerse con este documento (carné), tales como préstamo de libros, hacer uso de beneficios de refrigerios, apartar turnos para ingresar a las salas de Internet e ingreso a la Universidad.

Además esta dependencia no posee la comodidad de apartar implementos o escenarios de manera fácil ni llevar un seguimiento a cada uno de éstos, con el ánimo de observar su vida útil o el estado en que se encuentra el inventario de los implementos deportivos (balones, raquetas, mallas, entre otros).

De esta misma forma la Coordinación de Deportes muestra una incapacidad para generar de manera exacta y oportuna informes estadísticos que brinden datos como la concurrencia de un determinado tipo de estudiantes (por facultades), aproximar el tiempo que dedican los estudiantes al deporte, entre otros.

Otra de las grandes dificultades que enfrenta la Coordinación de Deportes se evidencia claramente en la cantidad de errores de tipo humano en la determinación de las fechas, horas y usuarios que realizan solicitudes de servicios; a esta dificultad se suman otras como el alto volumen de documentos almacenados de forma irregular en carpetas que a su vez son organizadas en archivadores donde se corre el riesgo de que se extravíe esa información, debido que cualquier persona que llega a esa dependencia tiene acceso a dichos archivos sin vigilancia alguna.

Para el préstamo de los escenarios deportivos (espacio físicos deportivos), es necesario haber realizado anteriormente la reserva, para la cual es necesario presentar una carta dirigida a la coordinación de deportes, en la cual se debe informar la fecha en que se utilizará el escenario, quién solicita el préstamo, el número de personas que utilizarán el escenario, entre otros datos necesarios para este préstamo, hasta este punto no se ve ninguna clase de problema, pero, hay que tener en cuenta que el objetivo es masificar la práctica de las disciplinas deportivas por parte de los estudiantes, pero cada vez que algún estudiante desee utilizar algún escenario debería presentar una carta de solicitud, acá se estaría hablando de un problema mencionado anteriormente, porque todas las cartas de solicitud que llegan a la oficina se deben ir almacenando, lo cual ocasiona que hayan cantidades de hojas de papel en la oficina ocupando un espacio, que además, pueden generar demoras al buscar algún dato y la perdida de información, porque, al estar expuestas en la oficina y no estar totalmente aseguradas, corren el riesgo de extraviarse.

En la actualidad no existe ningún tipo de amonestación o sanción para los usuarios que hacen mal uso de los recursos facilitados por esta dependencia, esto conlleva a recurrentes solicitudes que nunca son utilizadas y que dejan a muchos usuarios que realmente necesitan el servicio sin la posibilidad de utilizarlo debido a que, en teoría, los recursos se encuentran ocupados por otros.

2. ANTECEDENTES

La Universidad del Magdalena entró en un proceso de refundación, el cual trajo como resultado la modernización de sus áreas y dependencias, este proceso en el área de Bienestar Universitario, trae como resultado la subdivisión de sus servicios en cuatro áreas que son: Área de Desarrollo Humano, Área de Salud, Área de Cultura y Área de Deportes.

Este proyecto está específicamente ligado al área de deportes, el cual con el fin de masificar la práctica de las diferentes disciplinas deportivas que se ofrecen en esta Alma Mater, ha implantado varias estrategias para crear de esta área un organismo activo, con planes deportivos y la convicción de integrar deportivamente y recreativamente a toda la comunidad universitaria, entre las cuales, sus principales objetivos están el desarrollo de los distintitos niveles deportivos (Nivel Recreativo, Nivel Formativo, Nivel Representativo y la Escuela de Formación Deportiva), junto con estas ideas de desarrollo nace la necesidad de una herramienta para manejar toda la información que se administra en esta área, lo cual trae como resultado el SISTEMA DE INFORMACIÓN PARA EL ÁREA DE DEPORTES DE LA UNIVERSIDAD DEL MAGDALENA – SIADUM.

SIADUM es un sistema de información para manejar los datos referentes a los niveles y las escuelas deportivas, pero desafortunadamente no tiene dentro de sus procesos el de administrar las reservas y prestamos de los escenarios e implementos deportivos, por lo que se hace necesario diseñar e implementar un módulo para administrar estos procesos.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Analizar, diseñar e implementar un módulo del sistema de información de Bienestar Universitario, para la administración de implementos y espacios físicos deportivos en la Universidad del Magdalena.

3.2. OBJETIVOS ESPECÍFICOS

- Diseñar e implementar un módulo que permita al usuario el préstamo de implementos deportivos.

- Diseñar e implementar un módulo de administración que facilite la asignación y control de turnos a los empleados de la dependencia.

- Diseñar e implementar un módulo que permita al usuario el préstamo de escenarios deportivos.

- Diseñar e implementar un módulo en línea que permita al usuario reservar implementos y escenarios deportivos.

- Diseñar e implementar medidas de control en los módulos de préstamo de implementos y espacios físicos deportivos en el periodo del tiempo asignado a cada uno de los usuarios, imponiendo sanciones en los cuales se excedan en dicho tiempo.

- Diseñar y construir un módulo para generar reportes de los préstamos de implementos y escenarios deportivos que facilite información oportuna acerca de los usuarios del sistema, tales como programa al que pertenece, número de préstamos realizados, el tiempo de uso del servicio y el implemento o espacio físico que utilizó, entre otras.

4. JUSTIFICACIÓN

Con el desarrollo de este módulo, mejorará el servicio que presta el área de Coordinación de Deportes en la Universidad del Magdalena, específicamente con lo relacionado al préstamo de implementos y espacios físicos deportivos, en la cual se ofrecerán los servicios de reservas, préstamos y reporte de informes.

Por otra parte, con el módulo que se pretende desarrollar se garantiza una distribución más equitativa de los recursos, ya que se respetarán las solicitudes y se amonestarán a quienes no hagan un buen uso de los servicios, además ya no será necesario dejar el carné estudiantil como requisito para realizar el préstamo, y así los estudiantes podrán utilizar este documento en otros servicios, mencionados anteriormente, que son propios de la Universidad del Magdalena.

En la actualidad no existe ningún tipo de amonestación o sanción para los usuarios que hacen mal uso de los recursos facilitados por esta dependencia, esto conlleva a recurrentes solicitudes que nunca son utilizadas y que dejan a muchos usuarios que realmente necesitan el servicio sin la posibilidad de utilizarlo debido a que, en teoría, los recursos se encuentran ocupados por otros usuarios. Esta dificultad quedaría superada a través de los controles que se ofrecerán en este módulo.

Es importante mencionar que para la formación integral de un profesional es fundamental la practica de un deporte, y como se mencionó anteriormente, con el sistema en funcionamiento, además de obtener reportes o informes acerca de las reservas y préstamos, también se pueden realizar estadísticas sobre los programas que utilicen estos servicios, esta información se podría utilizar de muchas maneras, entre las cuales, teniendo en cuenta los intereses del área de deportes, con esta se pueden conocer los programas que dediquen menos tiempo en practicar un deporte o que deporte practican más, y así se pueden crear estrategias para fomentar o masificar la práctica de las diferentes disciplinas deportivas, factor importante dentro de los procesos de acreditación de los programas de la Universidad.

5. MARCO TEÓRICO CONCEPTUAL

5.1. CLIENTE SERVIDOR

Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes'', resultan en un trabajo realizado por otros computadores llamados servidores" [MON_6].

5.1.1. Qué es un cliente: Es el que inicia un requerimiento de servicio.
El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

5.1.2. Qué es un servidor: Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente (ver ejemplo en la figura 1). Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, entre otros servicios.

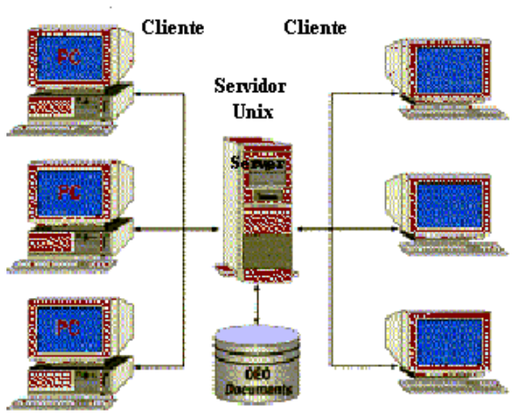

Figura 1: Ejemplo servidor.

5.1.3. Elementos de la arquitectura Cliente/Servidor: En esta aproximación, y con el objetivo de definir y delimitar el modelo de referencia de una arquitectura Cliente/Servidor, se deben identificar los componentes que permitan articular dicha arquitectura, considerando que toda aplicación de un sistema de información está caracterizada por tres componentes básicos:

- Presentación/Captación de Información
- Procesos
- Almacenamiento de la Información

Los cuales se suelen distribuir tal como se presenta en la figura 2:

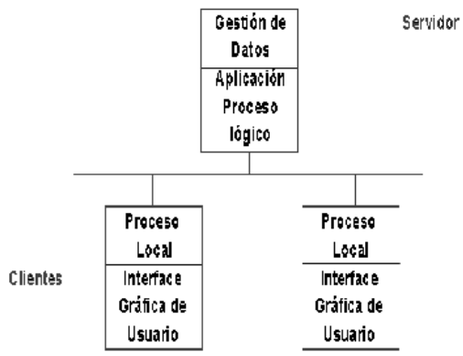

Figura 2: Aplicaciones Cliente/Servidor

Y se integran en una arquitectura Cliente/Servidor en base a los elementos que caracterizan dicha arquitectura, es decir:

- Puestos de Trabajo
- Comunicaciones
- Servidores

Tal como se presenta en la figura 3:

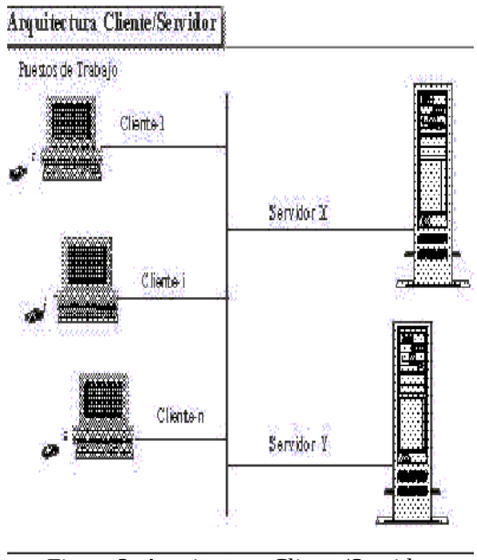

Figura 3: Arquitectura Cliente/Servidor

5.1.4. Características del modelo Cliente/Servidor: Como se muestra en la figura 4, **e**n el modelo Cliente/Servidor se pueden encontrar las siguientes características:

- El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
- Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.

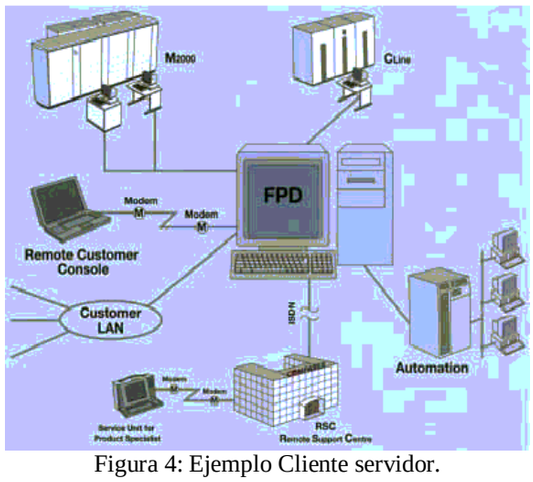

Figura 4: Ejemplo Cliente servidor.

- Un servidor da servicio a múltiples clientes en forma concurrente.
- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.
Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

- Además se constituye como el nexo de unión mas adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.
- Designa un modelo de construcción de sistemas informáticos de carácter distribuido.
- Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización.

En conclusión, Cliente/Servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en Token¬Ring, Ethernet, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

5.2. SISTEMA DE INFORMACIÓN

Es el conjunto de procesos que, operando sobre una colección de datos estructurada de acuerdo a una empresa, recopila, elabora y distribuye (parte de) la información necesaria para la empresa y para las actividades de dirección y control correspondientes, apoyando al menos en parte, la toma de decisiones necesarias para desempeñar las funciones y procesos de negocios de la empresa de acuerdo a su estrategia.

Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información [SEN_4].

5.2.1. Entrada de información: Es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas son datos o información que provienen o son tomados de otros sistemas o módulos. Esto último se denomina interfases automáticas.

5.2.2. Almacenamiento de información: El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos.

5.2.3. Procesamiento de información: Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados o un balance general de un año base.

5.2.4. Tipos de sistemas de información:

- Sistemas Transaccionales
- Sistemas de Apoyo de las Decisiones
- Sistemas Estratégicos

5.2.4.1. Sistemas transaccionales: Estos sistemas se dirigen principalmente a las áreas de ventas y mercadotecnia, administración y finanzas y al área de recursos humanos.
Sus principales características son:

- A través de éstos suelen lograrse ahorros significativos de mano de obra, debido a que automatizan tareas operativas de la organización.

- Con frecuencia son el primer tipo de Sistemas de Información que se implanta en las organizaciones. Se empieza apoyando las tareas a nivel operativo de la organización.

- Son intensivos en entrada y salid de información; sus cálculos y procesos suelen ser simples y poco sofisticados.

- Tienen la propiedad de ser recolectores de información, es decir, a través de estos sistemas se cargan las grandes bases de información para su explotación posterior.

- Son fáciles de justificar ante la dirección general, ya que sus beneficios son visibles y palpables.

5.2.4.2.Sistemas de apoyo de las decisiones: Los sistemas de apoyo a las decisiones (SAD) usan computadoras para el facilitar el proceso de toma de decisiones de tareas semiestructuradas.

Estos sistemas están diseñados no para reemplazar el criterio administrativo, sino para apoyarlo y hacer mas efectivo el proceso de toma de decisiones. Los sistemas de respaldo a las decisiones ayudan también a los gerentes a reaccionar rápidamente a los cambios de necesidades. Por lo tanto, queda claro que el diseño de un sistema efectivo requiere de un conocimiento profundo de cómo los gerentes toman las decisiones.

Las principales características de estos son:

- Suelen introducirse después de haber implantado los Sistemas Transaccionales más relevantes de la empresa, ya que estos últimos constituyen su plataforma de información.
- La información que generan sirve de apoyo a los mandos intermedios y a la alta administración en el proceso de toma de decisiones.

- Suelen ser intensivos en cálculos y escasos en entradas y salidas de información. Así, por ejemplo, un modelo de planeación financiera requiere poca información de entrada, genera poca información como resultado, pero puede realizar muchos cálculos durante su proceso.

- No suelen ahorrar mano de obra. Debido a ello, la justificación económica para el desarrollo de estos sistemas es difícil, ya que no se conocen los ingresos del proyecto de inversión.

- Suelen ser Sistemas de Información interactivos y amigables, con altos estándares de diseño gráfico y visual, ya que están dirigidos al usuario final.

- Apoyan la toma de decisiones que, por su misma naturaleza son repetitivos y de decisiones no estructuradas que no suelen repetirse. Por ejemplo, un Sistema de Compra de Materiales que indique cuándo debe hacerse un pedido al proveedor o un Sistema de Simulación de Negocios que apoye la decisión de introducir un nuevo producto al mercado.

- Estos sistemas pueden ser desarrollados directamente por el usuario final sin la participación operativa de los analistas y programadores del área de informática.

  Este tipo de sistemas puede incluir la programación de la producción, compra de materiales, flujo de fondos, proyecciones financieras, modelos de simulación de negocios, modelos de inventarios, entre otros.

5.2.4.3. Sistemas estratégicos: Es la necesidad de entender la forma en que la tecnología de la información es utilizada para dar forma a la estrategia competitiva de la empresa o para reducir la ventaja de sus rivales.

Sus principales características son:

- Su función primordial no es apoyar la automatización de procesos operativos ni proporcionar información para apoyar la toma de decisiones.

- Suelen desarrollarse in house, es decir, dentro de la organización, por lo tanto no pueden adaptarse fácilmente a paquetes disponibles en el mercado.

- Típicamente su forma de desarrollo es a base de incrementos y a través de su evolución dentro de la organización. Se inicia con un proceso o función en particular y a partir de ahí se van agregando nuevas funciones o procesos.

- Su función es lograr ventajas que los competidores no posean, tales como ventajas en costos y servicios diferenciados con clientes y proveedores. En este contexto, los Sistemas Estratégicos son creadores de barreras de entrada al negocio. Por ejemplo, el uso de cajeros automáticos en los bancos en un Sistema Estratégico, ya que brinda ventaja sobre un banco que no posee tal servicio. Si un banco nuevo decide abrir sus puertas al público, tendrá que dar este servicio para tener un nivel similar al de sus competidores.

- Apoyan el proceso de innovación de productos y proceso dentro de la empresa debido a que buscan ventajas respecto a los competidores y una forma de hacerlo es innovando o creando productos y procesos.

5.3. BASE DE DATOS

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California –USA.

Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos [JAM_2].

5.3.1. Requerimientos de las bases de datos: El análisis de requerimientos para una base de datos incorpora las mismas tareas que el análisis de requerimientos del software. Es necesario un contacto estrecho con el cliente; es esencial la identificación de las funciones e interfaces; se requiere la especificación del flujo, estructura y asociatividad de la información y debe desarrollarse un documento formal de los requerimientos.

5.3.2. Características de las bases de datos: Una base de datos contiene entidades de información que están relacionadas vía organización y asociación. La arquitectura lógica de una base de datos se define mediante un esquema que representa las definiciones de las relaciones entre las entidades de información. La arquitectura física de una base de datos depende de la configuración del hardware residente. Sin embargo, tanto el esquema (descripción lógica) como la organización (descripción física) deben adecuarse para satisfacer los requerimientos funcionales y de comportamiento para el acceso al análisis y creación de informes.

5.3.3. Ventajas en el uso de bases de datos: La utilización de bases de datos como plataforma para el desarrollo de Sistemas de Aplicación en las organizaciones se ha incrementado notablemente en los últimos años, se debe a las ventajas que ofrece su utilización, algunas de las cuales se comentarán a continuación:

- Globalización de la información: permite a los diferentes usuarios considerar la información como un recurso corporativo que carece de dueños específicos.
- Eliminación de información inconsistente: si existen dos o más archivos con la misma información, los cambios que se hagan a éstos deberán hacerse a todas las copias del archivo de facturas.
- Permite compartir información.

- Permite mantener la integridad en la información: la integridad de la información es una de sus cualidades altamente deseable y tiene por objetivo que sólo se almacena la información correcta.
- Independencia de datos: el concepto de independencia de datos es quizás el que más ha ayudado a la rápida proliferación del desarrollo de Sistemas de Bases de Datos. La independencia de datos implica un divorcio entre programas y datos.

5.3.4. El sistema manejador de bases de datos (dbms): Es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Una de las ventajas del DBMS es que puede ser invocado desde programas de aplicación que pertenecen a Sistemas Transaccionales escritos en algún lenguaje de alto nivel, para la creación o actualización de las bases de datos, o bien para efectos de consulta a través de lenguajes propios que tienen las bases de datos o lenguajes de cuarta generación.

5.3.5. Diseño de una base de datos: Es el proceso por el que se determina la organización de una base de datos, incluidos su estructura, contenido y las aplicaciones que se han de desarrollar. Durante mucho tiempo, el diseño de bases de datos fue considerado una tarea para expertos: más un arte que una ciencia. Sin embargo, se ha progresado mucho en el diseño de bases de datos y éste se considera ahora una disciplina estable, con métodos y técnicas propios.

Existen distintos modos de organizar la información y representar las relaciones entre los datos en una base de datos. Los Sistemas administradores de bases de datos convencionales usan uno de los tres modelos lógicos de bases de datos para hacer seguimiento de las entidades, atributos y relaciones. Los tres modelos lógicos principalmente de bases de datos son el jerárquico, de redes y el relacional. Cada modelo lógico tiene ciertas ventajas de procesamiento y también ciertas ventajas de negocios.

5.3.5.1. Modelo de jerárquico de datos: Una clase de modelo lógico de bases de datos que tiene una estructura arborescente. Un registro subdivide en segmentos que se interconectan en relaciones padre e hijo y muchos más. Los primeros sistemas administradores de bases de datos eran jerárquicos. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos

5.3.5.2. Modelo de datos en red: Es una variación del modelo de datos jerárquico. De hecho las bases de datos pueden traducirse de jerárquicas a en redes y viceversa con el objeto de optimizar la velocidad y la conveniencia del procesamiento. Mientras que las estructuras jerárquicas describen relaciones de muchos a muchos.

5.3.5.3. Modelo relacional de datos: Es el más reciente de estos modelos, supera algunas de las limitaciones de los otros dos anteriores. El modelo relacional de datos representa todos los datos en la base de datos como sencillas tablas de dos dimensiones llamadas relaciones. Las tablas son semejantes a los archivos planos, pero la información en más de un archivo puede ser fácilmente extraída y combinada.

5.3.6. Metodología de diseño de bases de datos: El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

5.3.6.1. Diseño conceptual: Parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la base de datos. Un esquema conceptual es una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para manipularla. Un modelo conceptual es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

5.3.6.2. Diseño lógico: Parte del esquema conceptual y da como resultado un esquema lógico. Un esquema lógico es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un modelo lógico es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

5.3.6.3. Diseño físico: Parte del esquema lógico y da como resultado un esquema físico. Un esquema físico es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos

5.4. MODELO DE DATOS

Es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para manipularlos. Hay dos tipos de modelos de datos: los modelos conceptuales y los modelos lógicos. Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción. Mediante los modelos conceptuales se puede construir una descripción de la realidad fácil de entender. En los modelos lógicos, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos [PRE_3].

En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad, y luego se transforma el esquema conceptual en un esquema lógico. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad. Un esquema es un conjunto de representaciones lingüísticas o gráficas que describen la estructura de los datos de interés.

Los modelos conceptuales deben ser buenas herramientas para representar la realidad, por lo que deben poseer las siguientes cualidades:

- Expresividad: deben tener suficientes conceptos para expresar perfectamente la realidad.
- Simplicidad: deben ser simples para que los esquemas sean fáciles de entender.
- Minimalidad: cada concepto debe tener un significado distinto.
- Formalidad: todos los conceptos deben tener una interpretación única, precisa y bien definida.

En general, un modelo no es capaz de expresar todas las propiedades de una realidad determinada, por lo que hay que añadir aserciones que complementen el esquema.

5.5. MODELO ENTIDAD-RELACIÓN

Es el modelo más utilizado para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976. Como se ve en la figura 5, el modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas [PRE_3].

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado *modelo entidad-relación extendido*.

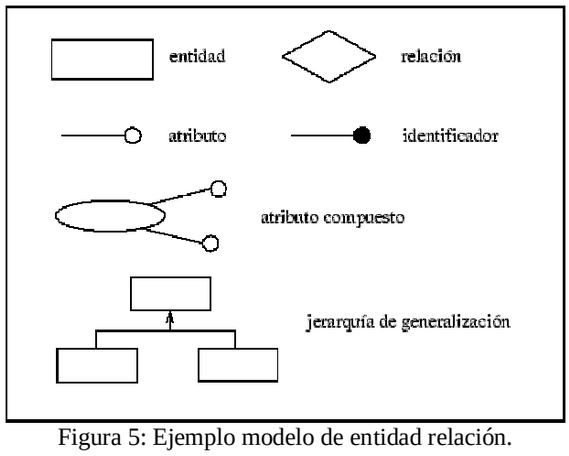

Figura 5: Ejemplo modelo de entidad relación.

5.5.1. Entidad: Cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Por ejemplo: coches, casas, empleados, clientes, empresas, oficios, diseños de productos, conciertos, excursiones, etc. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual.

Hay dos tipos de entidades: fuertes y débiles. Una *entidad débil* es una entidad cuya existencia depende de la existencia de otra entidad. Una *entidad fuerte* es una entidad que no es débil.

5.5.2. Relación (interrelación): Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior.

Las entidades que están involucradas en una determinada relación se denominan *entidades participantes*. El número de participantes en una relación es lo que se denomina *grado* de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación *binaria*; si son tres las entidades participantes, la relación es *ternaria* y así sucesivamente.

5.5.3. Atributo: Es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. Toda la información extensiva es portada por los atributos. Gráficamente, se representan mediante bolitas que cuelgan de las entidades o relaciones a las que pertenecen.

Cada atributo tiene un conjunto de valores asociados denominado *dominio*. El dominio define todos los valores posibles que puede tomar un atributo. Puede haber varios atributos definidos sobre un mismo dominio.

5.5.4. Identificador: Un identificador de una entidad es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad.

5.6. CREACIÓN DE UNA BASE DE DATOS

Para crear una base se deben realizar dos ejercicios de diseño: un diseño lógico y uno físico. El diseño lógico de una base de datos es un modelo abstracto de la base de datos desde una perspectiva de negocios, mientras que el diseño físico muestra como la base de datos se ordena en realidad en los dispositivos de almacenamiento de acceso directo. El diseño físico de la base de datos es llevado a cabo por los especialistas en bases de datos, mientras que el diseño lógico requiere de una descripción detallada de las necesidades de información del negocio de los negocios actuales usuarios finales de la base. Idealmente, el diseño de la base será una parte del esfuerzo global de la planeación de datos a nivel institucional [JAM_2].

El diseño lógico de la base de datos describe como los elementos en la base de datos han de quedar agrupados.

El proceso de diseño identifica las relaciones entre los elementos de datos y la manera más eficiente de agruparlos para cumplir con los requerimientos de información. El proceso también identifica elementos redundantes y los agrupamientos de los elementos de datos que se requieren para programas de aplicaciones específicos. Los grupos de datos son organizados, refinados y agilizados hasta que una imagen lógica general de las relaciones entre todos los elementos en la base de datos surja.

5.6.1. Bases de datos documentales: Son las derivadas de la necesidad de disponer de toda la información en el puesto de trabajo y de minimizar los tiempos del acceso a aquellas informaciones que, si bien se utilizan con frecuencia, no están estructuradas convenientemente. Esto se debe a que la procedencia de la información es muy variada (informes, notas diversas, periódicos, revistas, y muchos más).

5.6.2. Bases de datos distribuidas: Es aquella que se almacena en más de un lugar físico. Partes de la base de datos se almacena físicamente en un lugar y otras partes se almacenan y mantienen en otros lugares. Existen dos maneras de distribuir una base de datos. La base de datos central puede ser particionada de manera que cada procesador remoto tenga los datos necesarios sobre los clientes para servir a su área local. Los cambios en los archivos pueden ser justificados en la base de datos central sobre las bases de lotes, en general por la noche. Otra estrategia también requiere de la actualización de la base central de datos en hojas no laborables.

Aun otra posibilidad (una que se emplea en bases de datos grandes) es mantener solo un índice central de nombres y almacenar localmente los registros completos.
El procesamiento distribuido y las bases de datos distribuidas tienen beneficios e inconvenientes. Los sistemas distribuidos reducen la vulnerabilidad de un lugar único central y voluminoso. Permiten incremento en la potencia de los sistemas al adquirir mini computadoras que son más pequeñas y baratas. Finalmente incrementan el servicio y la posibilidad de respuesta de los usuarios locales. Los sistemas distribuidos, sin embargo, dependen de la alta calidad de las líneas de telecomunicaciones, las cuales a su vez son vulnerables. Además, las bases de datos locales pueden algunas veces alejarse de las normas y las definiciones de los datos centrales y hacen surgir problemas de seguridad al distribuir ampliamente el acceso a datos de alta sensibilidad.

5.6.3. Bases de datos orientadas a objetos: Estas son capaces de almacenar tanto procesos como datos. Por este motivo las bases orientadas al objeto deben poder almacenar información no convencional (como imágenes estáticas o en movimiento, colecciones de sonidos, entre otros). Este tipo de bases de datos deriva directamente de la llamada programación orientada a objetos, típica por ejemplo del lenguaje C/C++.

Entre las ventajas de las bases de datos orientadas al objeto destaca la posibilidad de tratar los casos excepcionales, que suelen ser la mayoría en la práctica cotidiana, en lugar de tratar de insertar la realidad en unos patrones rígidos que violentan para hacerla coincidir con los esquemas utilizados. Además, nadie pone en duda que es más cómodo manejar objetos de entorno que no es familiar, que trabaja, por ejemplo, con tablas, esquemas, cuadros, muchos más.

5.7. SISTEMA DE GESTIÓN DE BASE DE DATOS

Sistema desarrollado que hace posible acceder a datos integrados que atraviesan los límites operacionales, funcionales u organizacionales de una empresa [JAM_2].

5.7.1. Objetivos en el uso de un sistema de gestión de base de datos:

- Oportunidad, asociado a la eficiencia y eficacia.
- Disponibilidad, permitiendo la accesibilidad de datos
- Consistencias (oportunidad + disponibilidad), como calidad de datos
- Evolución, para adaptarse al entorno
- Integridad, en el nivel de los datos así como el sistema.

5.7.2. Objetivos del sistema de gestión de base de datos que podemos identificar son:

- Independencia de datos
- Accesibilidad limitada
- Datos al día y sin redundancias
- Consistencia
- Interfaz única
- Entrada directa a los datos
- Recuperación por diferentes accesos
- Función completa de interrogantes
- Estandarización
- Seguridad

5.8. LENGUAJE DE MODELAMIENTO UNIFICADO (UML - UNIFIED MODELING LANGUAGE)

Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables [FIG_5].

5.8.1. Modelo del desarrollo orientado a objetos: Se definen distintas dimensiones del modelo, como son el Modelo estático y el Modelo dinámico.

5.8.1.1. Modelo Estático: describe la estructura estática del sistema

- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Componentes

5.8.1.1.1. Diagrama de clases: Representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas.

Los diagramas de Clases por definición son estáticos, y representan las partes que interactúan entre sí.

Son utilizados durante el proceso de análisis y diseño de los sistemas informáticos, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Entre otras definiciones relacionadas están:

- **Propiedades:** también llamados atributos o características, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso.

- **Operaciones:** son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula,

a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrirPuerta, cerrarPuerta, buscarPuerta, etc.

- **Interfaz:** es un conjunto de operaciones y/o propiedades que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto.

- **Herencia:** se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una persona puede subdividirse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona, pero además tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, entre otra información.

En la figura 6 se muestra un ejemplo de cómo puede ser un diagrama de clases.

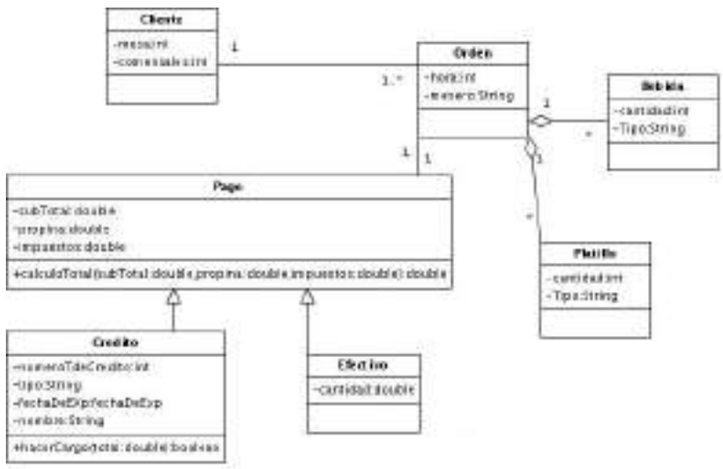

Figura 6: Ejemplo diagrama de clases.

5.8.1.1.2. Diagrama de objetos:

- En el análisis se usan los diagramas de objetos para indicar la semántica de los escenarios primarios y secundarios que proporcionan la traza del comportamiento del sistema
- En el diseño se usan diagramas de objetos para ilustrar la semántica de los mecanismos del diseño lógico
- Cada objeto se denota con su nombre y opcionalmente con sus atributos
- Los objetos interaccionan a través de enlaces con otros objetos
- Objeto cliente es el objeto que invoca una operación
- Objeto servidor es el objeto que suministra la operación
- Con una flecha se indica la dirección de la invocación, así como la operación, y un número de orden (opcional)

Un ejemplo grafico de estos objetos se puede observar en la figura 7 que se muestra a continuación.

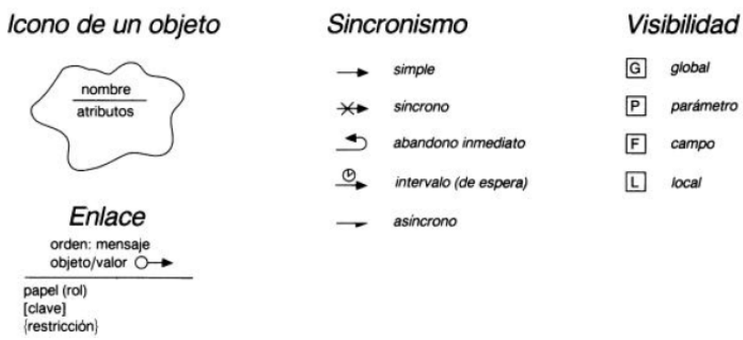

Figura 7: Ejemplo diagrama de objetos.

5.8.1.1.3. Diagrama de componentes: Describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.
Ver ejemplo en la figura 8.

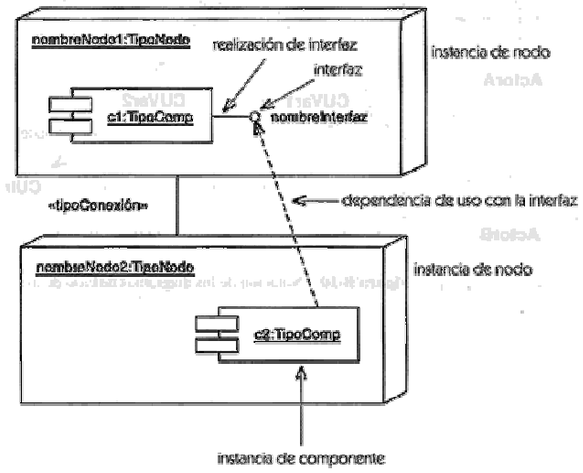

Figura 8: Ejemplo diagrama de componentes.

5.8.1.2. Modelo dinámico: describe la evolución dinámica y las interacciones entre objetos

- Diagramas de Casos de Uso
- Diagramas de interacción o de seguimiento de sucesos
- Diagrama de Secuencias
- Diagrama de colaboración
- Diagrama de Estado
- Diagrama de Actividades

5.8.1.2.1. Diagramas de casos de uso: Como se ve en el ejemplo de la figura 9, un diagrama de Casos de Uso es un diagrama sencillo que tiene como finalidad dar una visión global de toda la aplicación de forma que se pueda entender de una forma rápida y gráfica tanto por usuarios como por desarrolladores.

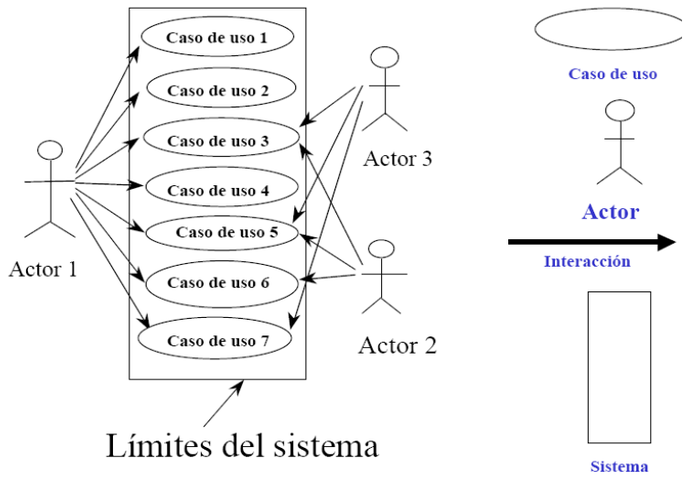

Figura 9: Ejemplo casos de uso.

5.8.1.2.2. Diagramas de interacción o de seguimiento de sucesos: Un diagrama de interacciones se usa para realizar una traza de la ejecución de un escenario en el mismo contexto que un diagrama de objetos de la notación Booch.

Los diagramas de interacción toman los elementos esenciales de los diagramas de objetos y los reestructuran para enfocarlos a la lectura de los mensajes en orden.

En la figura 10 se muestra un ejemplo de cómo puede ser un diagrama de iteración.

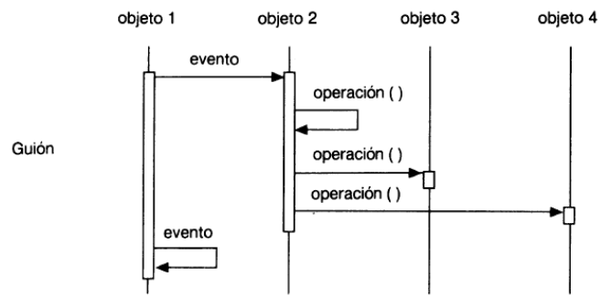

Figura 10: Ejemplo diagrama de iteración.

5.8.1.2.3. Diagrama de secuencias: Permiten en las fases iniciales de análisis y diseño:

- Razonar más en detalle como es el comportamiento de un escenario
- Obtener nuevas clases y objetos en el escenario (enriquecimiento del diccionario de datos)
- Detectar cuales son los métodos de las clases, al observar como se relacionan los objetos entre sí para llevar a cabo la tarea encomendada en el escenario

Se utilizan en las fases de prueba para validar el código y si se desea más detalle se utilizan los diagramas de Colaboración.

En la figura 11 se muestra un ejemplo de cómo puede ser un diagrama de secuencia.

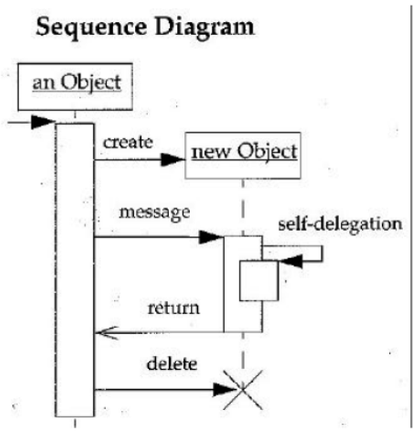

Figura 11: Ejemplo diagrama de secuencias

5.8.1.2.4. Diagrama de colaboración: Permiten profundizar en el nivel de detalle en los diagramas de Secuencia y expresan las colaboraciones de los objetos en tiempo de ejecución.

En la figura 12 se muestra un ejemplo sencillo de cómo puede ser un diagrama de colaboración.

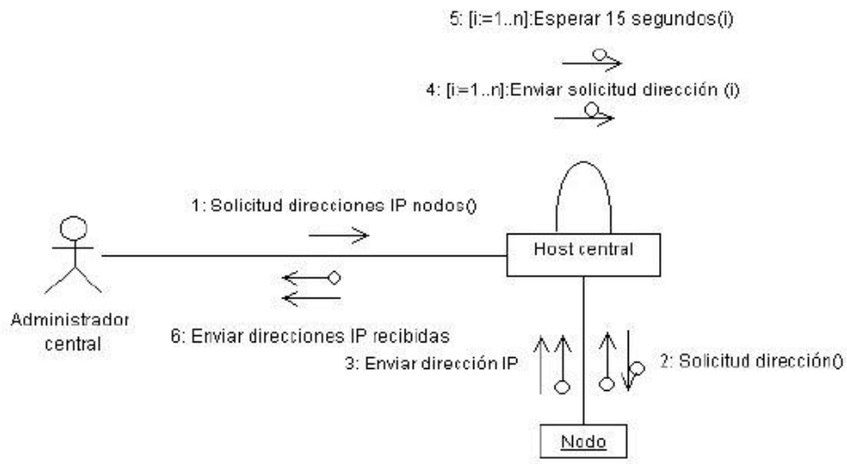

Figura 12: Ejemplo diagrama de colaboración.

5.8.1.2.5. Diagrama de estado: Los diagramas de estados muestran la secuencia de estados por los que pasa un objeto durante su vida y que se corresponden con los estímulos recibidos, junto con sus respuestas, acciones y cada diagrama de estados se corresponde con una clase o con un método.

En la figura 13 se muestra un ejemplo de un diagrama de estados.

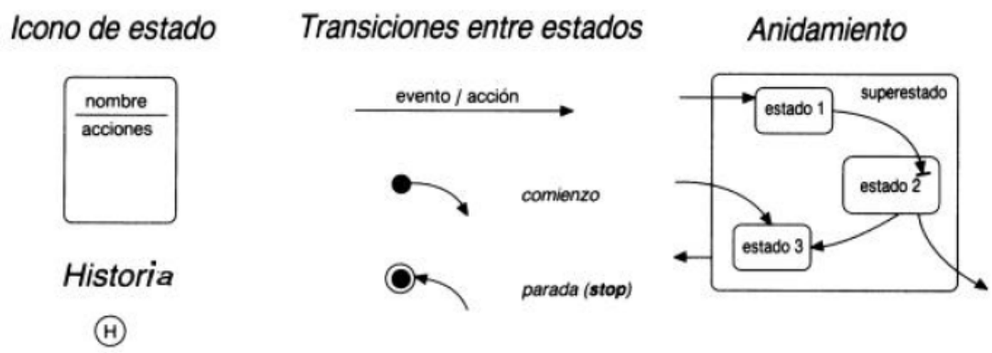

Figura 13: Ejemplo diagrama de estados.

5.8.1.2.6. Diagrama de actividades: Un diagrama de Actividad demuestra la serie de actividades que deben ser realizadas en un uso-caso, así como las distintas rutas que pueden irse desencadenando en el uso-caso.

Es importante recalcar que aunque un diagrama de actividad es muy similar en definición a un diagrama de flujo (típicamente asociado en el diseño de Software), estos no son lo mismo. Un diagrama de actividad es utilizado en conjunción de un diagrama uso-caso para auxiliar a los miembros del equipo de desarrollo a entender como es utilizado el sistema y como reacciona en determinados eventos. Lo anterior, en contraste con un diagrama de flujo que ayuda a un programador a desarrollar código a través de una descripción lógica de un proceso. Se pudiera considerar que un diagrama de actividad describe el problema, mientras un diagrama de flujo describe la solución.

En la figura 14 se muestra un ejemplo de cómo puede ser un diagrama de actividades.

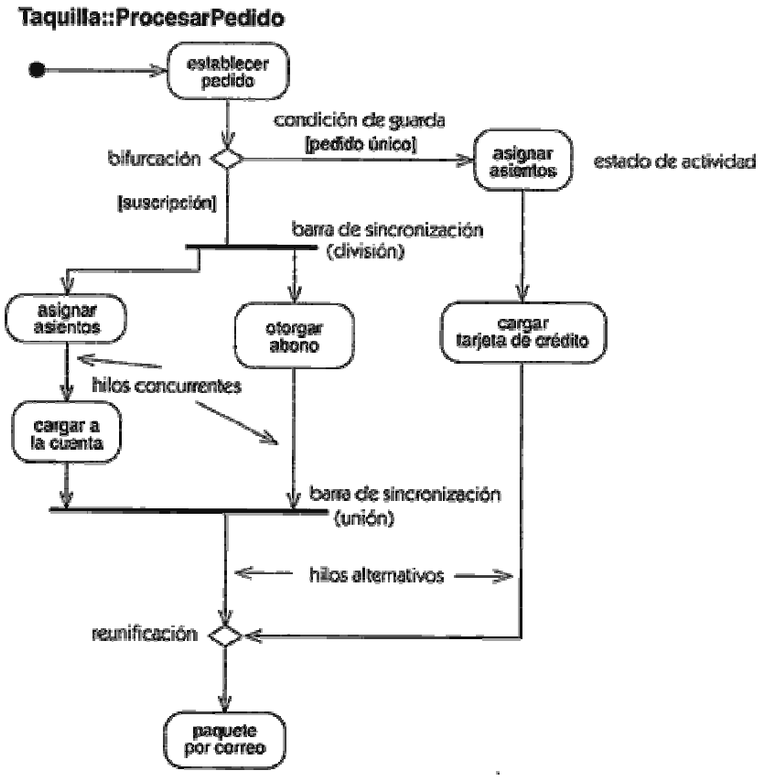

Figura 14: Ejemplo diagrama de actividades.

5.9. METODOLOGÍA

La metodología que se utilizó para el desarrollo de este módulo, es una simplificación del método CDM (Custom Development Method) de Oracle Corporation, al que se le han incorporado algunos modelos más recientes, con la notación estándar de UML (Unified Modeling Language) propuesta por Booch, Rumbaugh y Jacobson [ORA_11].

El método consiste en ejecutar una serie de actividades que se especifican como componentes de un entregable. Un entregable, por lo tanto, es un documento donde se consignan los resultados del trabajo correspondiente a una fase del desarrollo de software.

La premisa fundamental de este método, consiste en impedir la generación de código para la construcción del software hasta que no se tenga una idea muy precisa de lo que se necesita de él. Por lo tanto, todo desarrollo o adaptación de software debe partir de una clara definición de los requerimientos al nuevo sistema informático; tanto de información, como de comportamiento. A partir de una buena descripción de esos requerimientos, se sigue con la construcción de modelos que se irán transformando en otros, a medida que se va bajando de nivel de abstracción; como lo muestra la Figura 15.

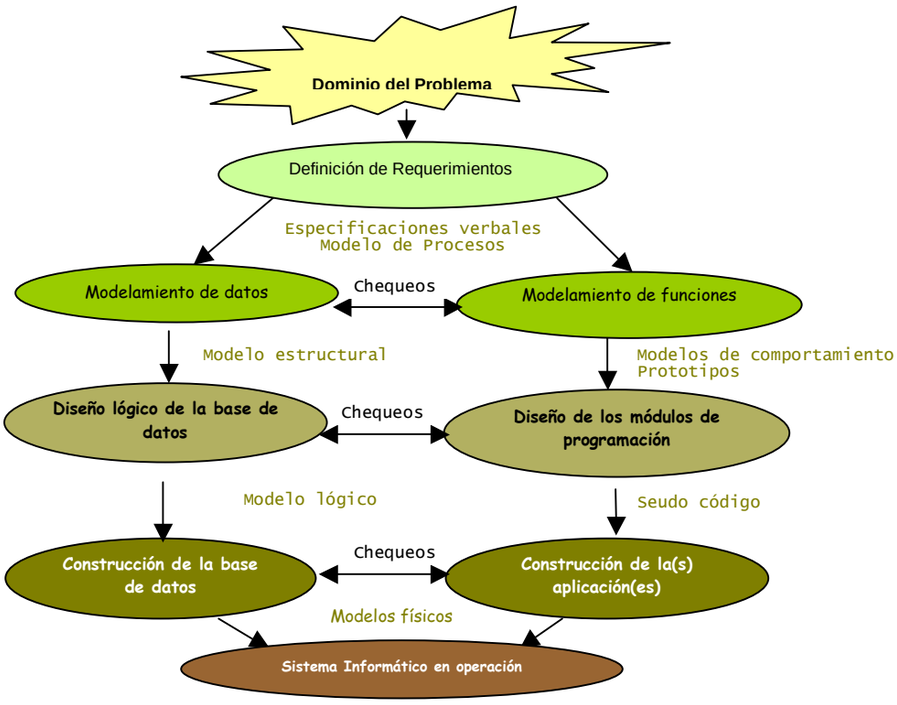

Figura 15: Desarrollo de un sistema informático.

5.9.1. Fases de la metodología CDM (Custom Development Method):

Entre las fases de este método están:

- Fase de definición
- Fase de análisis
- Fase de diseño
- Fase de construcción
- Fase de transición
- Fase de producción

5.9.1.1. Fase de definición: El modelo en prosa y el modelo de procesos, son los dos frutos del trabajo principales en la primera fase del desarrollo de software, denominada la fase de definición.

La principal ventaja del Modelo de Procesos es la posibilidad de representar pasos de un proceso que son realizados exclusivamente por personal humano y no por un computador. También, permite representar flujos de materiales, además de los flujos de datos, entre pasos de un proceso, o entre los pasos y los actores del sistema.

5.9.1.2. Fase de análisis: Se subdivide en dos procesos principales: el modelamiento conceptual de datos y el modelamiento conceptual de funciones.

En esta fase, y en las dos fases siguientes, a medida que se trabaja en el modelamiento de datos, conviene hacer chequeos cruzados con los modelos funcionales para que no se omita la representación de datos relevantes para una funcionalidad dada y, por otro lado, no se represente información que no se utiliza por ninguna función del sistema.

5.9.1.3. Fase de diseño: Consiste en la transformación de los modelos conceptuales construidos en la fase de análisis, el modelo conceptual de datos y el modelo conceptual de funciones, para obtener los modelos lógicos; lo que significa que se baja de nivel de abstracción y los modelos ya quedan atados a una tecnología de desarrollo de software, específica. Por ejemplo, en esta fase, se puede optar por transformar el Diagrama de Clases de UML al modelo relacional por ser éste el modelo lógico de datos elegido para la solución informática.

5.9.1.4. Fase de construcción: Esta fase es conocida como la fase de construcción del sistema informático.

Viene la transformación e integración de los modelos lógicos en un solo modelo físico. El modelo físico, contiene todas las especificaciones de la arquitectura, del almacenamiento de datos y del comportamiento del sistema, en los lenguajes de programación elegidos desde los inicios del desarrollo o de la adaptación de software.

5.9.1.5. Fase de transición: El modelo físico del desarrollo de un nuevo producto software, o de la mejora de uno ya existente, debe pasar por varios ambientes que simulan al ambiente real y que sirven para probar el sistema, primero por el proveedor, personal informático, y por último el usuario final. Por lo tanto, sólo se puede pasar a instalar el nuevo software en el ambiente real de operación, cuando la nueva aplicación o la mejora del software haya sido aprobada por los usuarios finales y la fase de transición se da por finalizada, únicamente, cuando se abandonan definitivamente los sistemas viejos, sin importar si éstos son manuales, o no.

5.9.1.6. Fase de producción: Ocurre cuando el producto software está operando tal como se esperaba y está ofreciendo todos sus servicios a los usuarios finales. En esta fase, se debe monitorear el sistema permanentemente para detectar problemas ocasionados por el trabajo concurrente con múltiples usuarios y el manejo de seguridad del sistema. También, en esta etapa o fase se debe iniciar el proceso de afinamiento de la base de datos; haciendo todos los correctivos demandados, para un óptimo desempeño del sistema. Por último, en esta fase, se debe medir cómo afectó el nuevo aplicativo, en forma positiva o negativa, a los otros ya existentes y por eso en la fase de producción deben hacerse pruebas de integración de los sistemas informáticos.

5.9.2. Plan de desarrollo del proyecto: Se divide en las fases del proyecto que son:

5.9.2.1. Fase de definición:

- Descripción del dominio del problema:
  Descripción del entorno de la aplicación y de los procesos que pretenden ser mejorados con el apoyo del nuevo software.

- Modelo de procesos:
  Descripción de los procesos automatizados, o no, necesitados para cumplir con los objetivos de la organización en el dominio del problema, sus responsables, su duración y los eventos que provocan su ejecución.

- Servicios y usuarios:
  Se extraen del modelo de procesos, los servicios o las funciones que debe ofrecer el nuevo sistema informático para cumplir con los objetivos de la organización.

- Riesgos, impactos y factores críticos del éxito:

  - Riesgos Organizacionales: Se especifican los riesgos en que está incurriendo la organización sino lleva a cabo la automatización.

  - Impacto Esperado: Se describen los beneficios, tanto en lo económico como en el bienestar de los usuarios, que se esperan obtener cuando el nuevo sistema esté en operación.

  - Factores críticos de éxito: La especificación de los factores críticos de éxito para implantar el nuevo sistema, es necesaria para tenerlos presentes en las fases siguientes.

- Diagrama jerárquico de funciones:
  Esta descomposición se representa mediante un diagrama jerárquico de funciones donde se resaltan las funciones comunes o compartidas por otras.

- Interfaces:
  Consiste en la colección de formas, informes, cuadros de diálogo y menús que el sistema debe ofrecer para cumplir con la funcionalidad especificada en el diagrama jerárquico de funciones.

- Arquitectura técnica inicial:
  Define el hardware, el software, las herramientas y la configuración necesaria para el éxito del proyecto; ajustado todo a las posibilidades de la organización.

- Interacción con sistemas existentes:
  Consiste de las descripciones de alto nivel para cada fuente o destino de datos y los requerimientos funcionales para importar, exportar o crear vínculos de acceso a los datos.

- Glosario de términos:
  Se inicia la documentación, definiendo la terminología empleada en las áreas involucradas de la organización y que tienen que ver con el aplicativo.

5.9.2.2. Fase de análisis:

- Modelo de datos:
  Se presenta una definición detallada de la información, con una representación gráfica de su estructura en un diagrama.

- Modelo de funciones:
  Se presenta una descripción detallada de las funciones que deben ser provistas por el sistema informático.

- Diagrama de transición de estados:
  El diagrama de transición de estados, muestra el conjunto de posibles vidas de los objetos importantes del sistema originadas por los eventos que ocurren en el sistema real.

- Matriz CRUD:
  Esta matriz muestra el uso de los datos por las funciones del sistema.

  En las columnas de la matriz se especifican las funciones y en las filas se especifican las entidades, o más específicamente los atributos de éstas. En las celdas de la matriz van las letras iniciales de los tipos de operaciones que se pueden hacer en la entidad por una función: creación o inserción de datos, lectura, actualización o eliminación.

- Glosario de términos:
  Se amplía el glosario de términos, con la definición, unidad de medida y sinónimos de las propiedades de los objetos resultantes en los modelos.

5.9.2.3. Fase de diseño:

- Modelo lógico de la base de datos:
  Este modelo se deriva del modelo de entidad relación o del diagrama de clases de UML, está compuesto por todos los objetos que contienen datos junto con todas las restricciones a las que están sujetos.

- Plan de capacidad:
  El plan de capacidad incluye aspectos de rendimiento y concurrencia para que el sistema responda adecuadamente ante los servicios que debe ofrecer.

- Diseño de la aplicación:
  Consiste de la colección de los objetos funcionales que deben ser creados para la correcta y eficiente operación del sistema informático.

- Esquema de autorización:
  Este esquema es usado para conceder privilegios a los usuarios, o grupos de usuarios, sobre los objetos de la base de datos.

- Modelo de pruebas del sistema:
  Este modelo es la espina dorsal para el desarrollo de las pruebas funcionales.

- Manual del usuario inicial:
  Consiste de la documentación primaria orientada a los usuarios finales, que indica como se usará la aplicación para responder a los eventos del negocio.

5.9.2.4. Fase de construcción:

- Diseño físico de los datos:
  Consiste de la descripción completa de los parámetros para el almacenamiento físico identificados en la fase anterior.

- Código de la aplicación:
  Se genera el código fuente, libre de error, de todas las unidades de programación necesarias para el correcto funcionamiento del sistema.

- Manual de usuario completo:
  En esta fase, se revisa el manual del usuario final y se crea la versión definitiva.

- Plan de instalación:
  Describe en detalle la secuencia de pasos a seguir para instalación exitosa tanto del hardware como del software que se haya adquirido para el nuevo sistema.

- Pruebas:
  Describe los resultados de las pruebas funcionales realizadas en el sitio de trabajo del proveedor, antes de su instalación.

5.9.2.5. Fase de transición:

- Sistema de ambientes simulados de producción:
  La nueva aplicación se pone en funcionamiento en el ambiente que simulan el ambiente real donde se encuentra la base de datos de producción para realizar las pruebas funcionales y las correcciones de entrar a operar en el ambiente definitivo antes de entrar a operar en el ambiente definitivo.

- Evaluación del entrenamiento:
  El entrenamiento o capacitación en el uso del nuevo sistema se inicia, en esta fase, con el personal informático responsable de los aplicativos.

- Resultados de pruebas:
  Después ejecutar las pruebas predefinidas en los ambientes simulados, se describe en detalle el resultado de las mismas.

5.9.2.6. Fase de producción:

- Evaluación del sistema de producción:
  En proyectos grandes o medianos, usualmente, se especifica un período de garantía donde los desarrolladores están comprometidos a analizar los problemas funcionales y a hacer todos los arreglos necesarios para solucionar dichos problemas.

- Evaluación del desempeño del sistema:
  El desempeño tiene que ver con el tiempo que tarda el sistema para dar respuesta a los requerimientos de los usuarios y con el número de transacciones que el sistema puede ejecutar en período de tiempo determinado.

- Mejoras futuras:
  Este componente describe las mejoras futuras funcionales al sistema.

6. PROCESO DEL DESARROLLO DEL SISTEMA DE INFORMACIÓN

Como se mencionó anteriormente esta metodología, consiste en ejecutar una serie de actividades que se especifican como componentes de un entregable. Un entregable, por lo tanto, es un documento donde se consignan los resultados del trabajo correspondiente a una fase del desarrollo de software, a continuación se presentan los entregables de cada fase:

6.1. ENTREGABLE DE LA FASE DE DEFINICIÓN

6.1.1. Introducción

En este entregable se hará una descripción general de cómo ha venido y viene funcionando actualmente Bienestar Universitario, en la administración de los implementos y espacios físicos deportivos de la Universidad del Magdalena. En este entregable se podrá comprender la visión que se tiene para mejorar la gestión del Bienestar Universitario, en el ámbito de la administración de los implementos y espacios físicos deportivos, con el apoyo de una solución informática.

Además, este documento está determinado como punto de referencia para el desarrollo del módulo informático, ya que aquí se hace una descripción inicial y se profundizan en aspectos indispensables de la conformación y funcionamiento de los procesos que se destacan en esta dependencia, con el propósito de ayudar a adoptar con mejor disposición la solución informática que se pretende implementar.

6.1.2. Descripción del dominio del problema

La Coordinación de Deportes en la Universidad del Magdalena es un área que pertenece a la dependencia de Bienestar universitario, y se encarga de fomentar y estimular la práctica deportiva dentro de esta comunidad.
Entre los servicios que presta esta área se encuentra la reserva y préstamo de implementos y escenarios deportivos, procesos que hasta ahora se han realizado manualmente, ya que no cuentan con una herramienta para estos procesos, quedando expuestos a la pérdida de

información, demora en los procesos, almacenamiento físico (información en hojas de papel) en la oficina, entre otros problemas.

Con el eventual crecimiento que ha presentado la Universidad del Magdalena, ha orientado sus esfuerzos hacia la modernización y reestructuración de las actividades que realizan cada una de las dependencias.

La Coordinación de Deportes, en sus estrategias para unirse al desarrollo de la Universidad ha demostrado la necesidad de implementar un módulo para manejar la información referente a las reservas y préstamos de implementos y escenarios deportivos.

6.1.3. Modelo de procesos

Los procesos involucrados en este proyecto son el préstamo de implementos y espacios físicos, la descripción y el diagrama de flujos de estos mismos, dentro del proyecto se encuentran a continuación:

Préstamos de implementos deportivos: En la figura 16, se muestra el diagrama de flujo, relacionado al proceso de préstamos de implementos deportivos.

Descripción: Se puede prestar implementos deportivos a los estudiantes activos mediante el formato BU-P04-F04 "Préstamos de implementos del área de cultura y deportes" y se diligencia el formato BU-P04-F05 "Confirmación de préstamos de implementos o espacios físicos".

Diagrama de flujos:

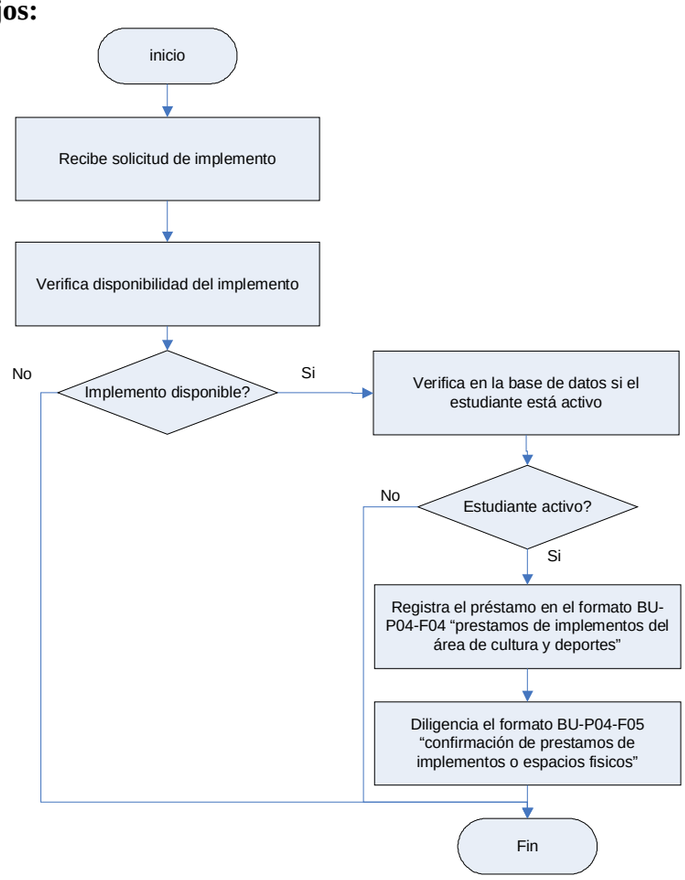

Figura 16: Diagrama de Préstamo de Implementos Deportivos

Préstamo de espacios físicos deportivos: En la figura 17, se muestra el diagrama de flujo, relacionado al proceso de préstamos de espacios físicos deportivos.

Descripción: Recibe las solicitudes de espacio físico y verifica su disponibilidad, lo anota en el formato BU-P04-F09 "Solicitud de reserva de espacios físicos del área de cultura y deportes" y diligencia el formato BU-P04-F05 "Confirmación de préstamos de implementos o espacios físicos"

Diagrama de flujos:

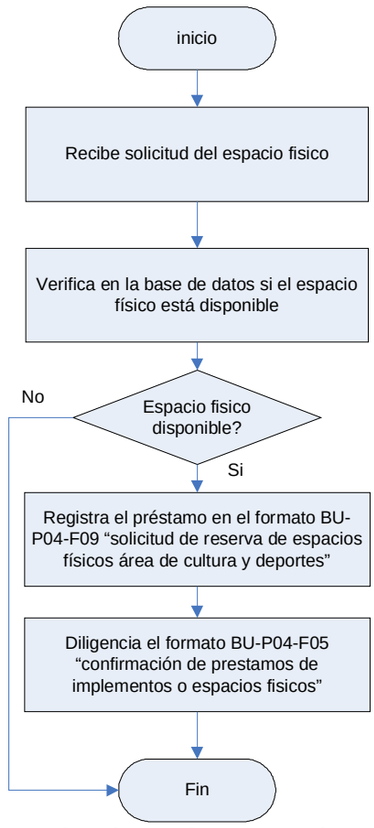

Figura 17: Diagrama de Préstamo de Espacios Físicos Deportivos

6.1.4. Servicio y usuarios finales

La Tabla 1, describe los servicios, usuarios finales directos e indirectos, que ofrecerá el nuevo módulo de préstamos y reservas.

Servicio: Reserva de implementos deportivos.	
Usuarios finales directos: Coordinador de deportes, asistente de coordinación de deportes.	**Usuarios finales indirectos:** Estudiantes de la Universidad del Magdalena.
Servicio: Préstamo de implementos deportivos.	
Usuarios finales directos: Coordinador de deportes, asistente de coordinación de deportes.	**Usuarios finales indirectos:** Estudiantes de la Universidad del Magdalena.
Servicio: Reserva de escenarios deportivos.	
Usuarios finales directos: Coordinador de deportes, asistente de coordinación de deportes.	**Usuarios finales indirectos:** Estudiantes de la Universidad del Magdalena.
Servicio: Préstamo de escenarios deportivos.	
Usuarios finales directos: Coordinador de deportes, asistente de coordinación de deportes.	**Usuarios finales indirectos:** Estudiantes de la Universidad del Magdalena.

Tabla 1: Servicio y usuarios finales.

6.1.5. Riesgos organizacionales

El área de deportes al no contar con una herramienta como la propuesta en este proyecto para realizar los procesos referentes a las reservas y préstamos, seguirá incurriendo en los mismos errores como son: la no distribución equitativa de los recursos deportivos, ya que es posible que no se respeten las solicitudes, no habrá forma de amonestar a quienes no hagan un buen uso de los servicios, por otro lado seguirá siendo necesario dejar el carné estudiantil como requisito para realizar el préstamo, impidiendo la utilización de éste para otros servicios necesarios para los estudiantes (utilización de los servicios bibliotecarios, servicio de Internet, identificación como estudiantes de esta universidad, entre otros), posibles pérdidas de datos por el indebido manejo de la información en archivos planos, que deben ser almacenados o acumulados en la oficina, con el riesgo que cualquier persona ajena pueda tener acceso a esta.

6.1.6. Impacto esperado

El Módulo que se plantea, permite evitar los errores anteriormente mencionados lo que contribuye a descongestionar la oficina, ya que podría realizar reservas a través de un módulo en línea, además los usuarios podrán hacer los préstamos de objetos o escenarios deportivos deseados al instante con el único requisito de ser estudiante activo de la Universidad del Magdalena. Para este propósito, se cuenta con una red interna propia de la Institución que se ofrece a toda la Comunidad Universitaria.

Además con el módulo que se pretende desarrollar, se garantiza una distribución más equitativa de los recursos, ya que se respetarán las solicitudes y se amonestarán a quienes no hagan un buen uso de los servicios, implementos y escenarios deportivos. La dificultad descrita en el planteamiento del problema quedaría superada a través de los controles que se ofrecerán con este módulo.

6.1.7. Factores críticos de éxito

Para que este módulo logre una buena acogida, debe tener acceso a la información referente a las reservas y préstamos de implementos y escenarios deportivos del área de deportes, por lo que es necesario tener acceso a los datos de algunas tablas que pertenecen a la base de datos del sistema de información de bienestar universitario (SIADUM), las tablas son: ESTUDIANTE, HORARIO_ENTRENAMIENTO, PROGRAMA y FACULTAD.

Además es necesario que el manejo de esta aplicación pueda ser comprendido rápidamente, por lo cual es necesario que su administrador deba estar capacitado para realizar consultas, ingresos, modificaciones y generar informes, los indispensables para el óptimo funcionamiento del módulo.

6.1.8. Diagrama jerárquico de funciones

La figura 18, es un ejemplo de cómo se llevan a cabo las funciones de reservas y préstamos en el área de deportes, donde solo se muestran los procesos que involucran a este módulo.

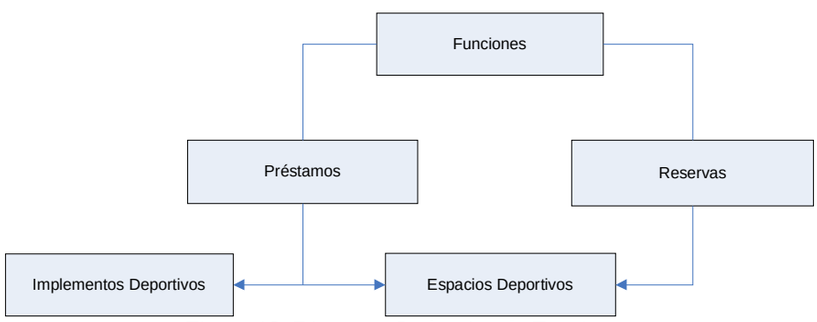

Figura 18: Diagrama Jerárquico de Funciones

6.1.9. Interfaces de usuario

A continuación se mostraran algunas vistas y cuadros de diálogo que se incluirán en el módulo que se construye.

En la Figura 19 se muestra la página de inicio del módulo, en ella se puede realizar la validación de usuario.

Figura 19: Página de inicio

Luego de realizar la validación de usuario, se ingresa al menú (ver Figura 20), en el cual cada usuario (administrador o estudiante en línea) tendrá las opciones respectivas.

Figura 20: Menú administrador

Al escoger alguna opción en el menú, se ingresa a la página respectiva, en el ejemplo de la Figura 21, se muestra una vista en caso en que se haya escogido en el menú “Reserva de Implementos Deportivos”, en la que se muestra una tabla con algunos datos de las reservas ya realizadas, y algunas otras opciones entre las cuales están, modificar, eliminar, crear nuevas reservas, entre otras.

Figura 21: Reserva de implementos.

Siguiendo con el ejemplo de “Reserva de Implementos Deportivos”, se escoge en la reserva de implementos deportivos “nueva reserva”, se abrirá un formulario como el que se presenta a continuación en la Figura 22, para registrar todos los datos necesarios para la reserva.

Figura 22: Registro reserva de implemento.

Esta aplicación muestra cuadros de diálogos, con el fin de informar al usuario si está seguro de realizar alguna acción, si olvidó ingresar algún dato, si cometió algún error entre otros, como se aprecia en la figura 23.

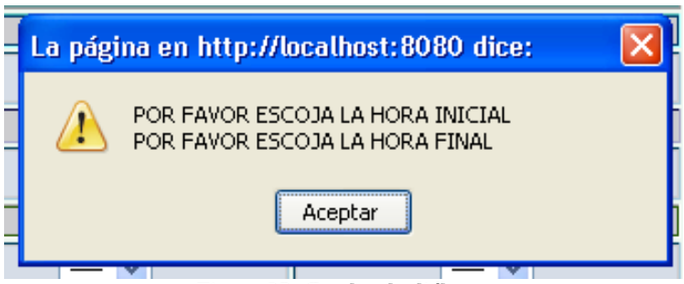

Figura 23: Cuadro de diálogo.

6.1.10. Arquitectura técnica:

Este proyecto se realizará bajo una arquitectura de aplicaciones Web cliente/servidor, usando Java Server Pages (JSP) con backend en AJAX y DOM como lenguaje de programación, Oracle como manejador de la Base de Datos y Apache Tomcat como contenedor Web.

Los sistemas típicos cliente/servidor pertenecen a la categoría de las aplicaciones de dos niveles, como lo muestra la figura 24, la aplicación reside en el cliente mientras que la base de datos se encuentra en el servidor.

Algunas tecnologías para el desarrollo de aplicaciones Web son:

- PHP
- ASP
- JSP (Tecnología Java)

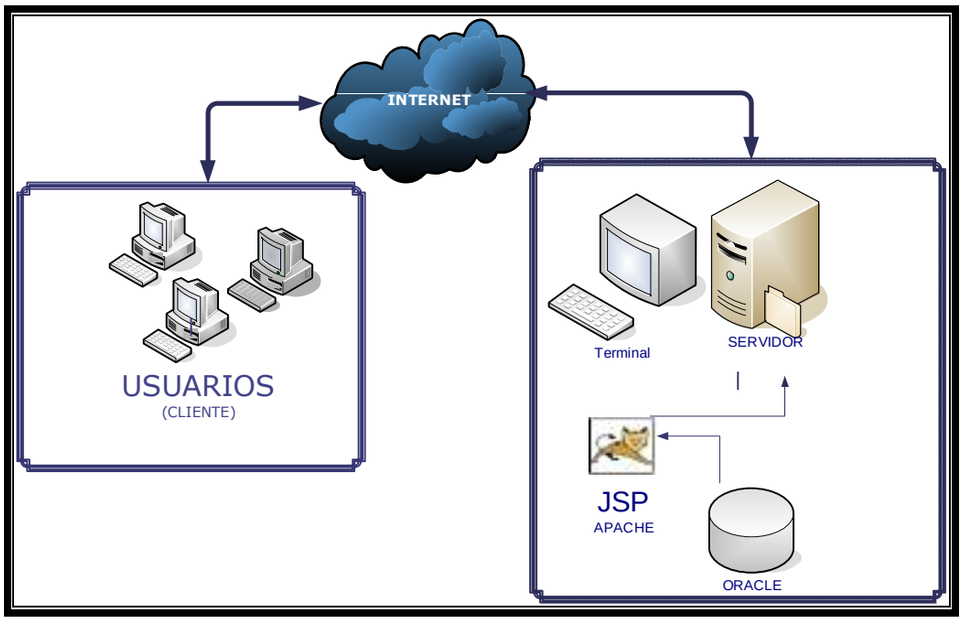

Figura 24: Arquitectura técnica.

Algunas definiciones con respecto a la arquitectura del proyecto:

Cliente: Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Servidor: es una aplicación informática o programa que proporciona información, recursos y servicios a los clientes de la red. Cuando un cliente pide un recurso como, por ejemplo, un archivo, datos de una base de datos, acceso a aplicaciones remotas o impresión centralizada, el servidor proporciona estos recursos al cliente.

Algunos servidores Web importantes son:
- Apache Tomcat
- Internet Information Server

Apache Tomcat: es un servidor Web con soporte de servlets y JSP's. Incluye el compilador Jasper, que compila JSP's convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache.

Lenguaje De Programación Java: es una plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

JSP: es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es una tecnología orientada a crear páginas Web con programación en Java.

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

DOM: es un acrónimo de Document Object Model, que en español quiere decir "Modelo de Objetos de Documento". Es una forma de representar los elementos de un documento estructurado (tal como una página Web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. Es el estándar oficial de la World Wide Web Consortium (W3C) para representar documentos estructurados de una manera neutral a la plataforma y al lenguaje de programación.

6.1.11. Interacción con otros sistemas informáticos

Este módulo, se creó para complementar el sistema de información de bienestar universitario (SIADUM), en lo referente a los procesos de préstamos y reservas de implementos y escenarios deportivos.

SIADUM, nos facilitará la información de los estudiantes activos, los inventarios de implementos deportivos, los horarios de entrenamiento y los escenarios utilizados en dichos entrenamientos, para lo cual es necesario establecer una conexión con su base de datos.

Para completar la integración de estos sistemas, y facilitar su adaptación, este módulo se creará con la misma arquitectura, en cuanto a lenguajes de programación y motores de bases de datos, y para la navegación de los usuarios, se colocará un acceso directo a este módulo desde la página principal de SIADUM.

6.1.12. Glosario de términos:

Aplicación: Es un programa informático diseñado para facilitar al usuario la realización de un determinado tipo de trabajo.

Bienestar universitario: Es el conjunto de políticas y programas encaminado a lograr el desarrollo integral de cada uno de los miembros de la Universidad del Magdalena, el mejoramiento de su calidad de vida y, en consecuencia, el de la institución.

BU-P04-F04: Formato de préstamos de implementos del área de cultura y deportes.

BU-P04-F05: Formato de confirmación de préstamos de implementos o espacios físicos.

BU-P04-F09: Formato de solicitud de reserva de espacios físicos del área de cultura y deportes

Diagrama de flujo: Se basan en la utilización de diversos símbolos para representar operaciones específicas de un proceso.

Entregable: Es un documento donde se consignan los resultados del trabajo correspondiente a una fase del desarrollo de software.

Módulo: Es un componente de un sistema, el cual posee una interfaz bien definida hacia otros componentes.

Proceso: Es un conjunto de actividades o eventos que se realizan o suceden con un determinado fin.

6.2. ENTREGABLE DE LA FASE DE ANÁLISIS

6.2.1. Introducción

En este entregable se encuentra el diagrama de clases, de entidad relación, de estados y los de casos de uso con sus respectivos escenarios, estos representan los modelos conceptuales, los cuales se elaboraron en base a los modelos de procesos realizados en la fase de definición.

6.2.2. Modelo de datos

Para el modelo de datos se muestran a continuación en la figura 25 el diagrama de clases y en la figura 26 el diagrama de entidad relación.

6.2.2.1. Diagrama de Clases de UML

En la figura 25 se puede apreciar el diagrama de clases utilizado para el diseño de este módulo.

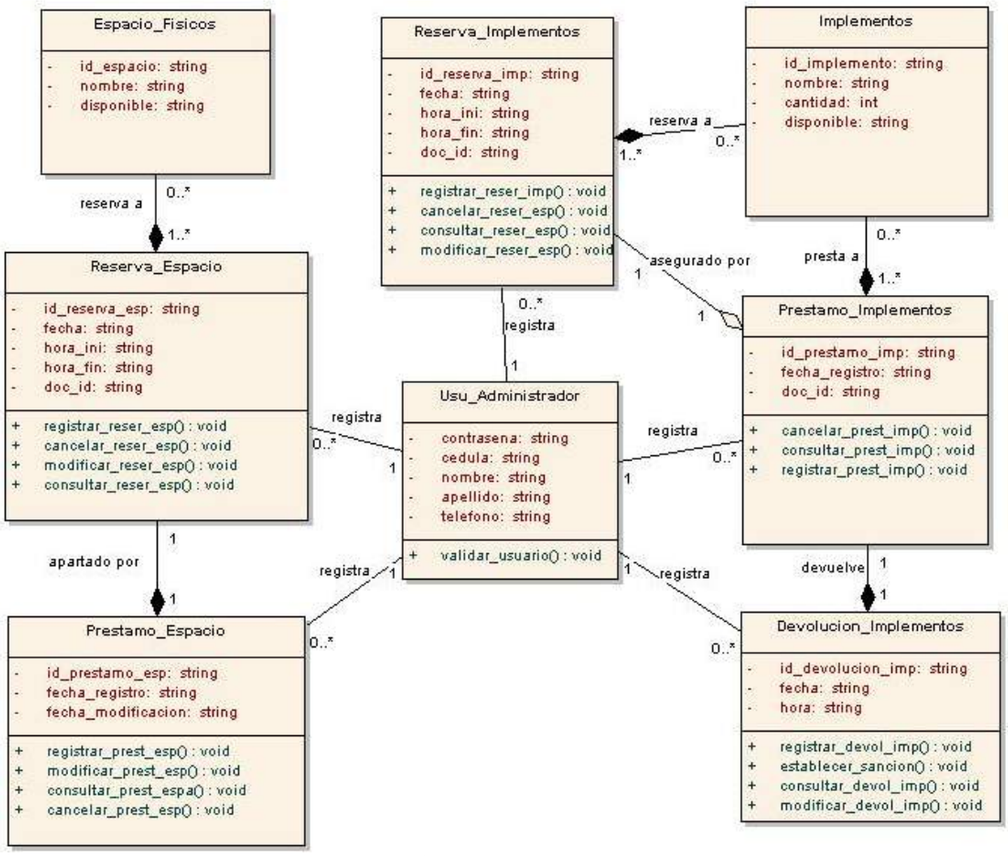

Figura 25: Diagrama de Clases de UML.

6.2.2.2. Diagrama de Entidad Relación

En la figura 26 se aprecia el diagrama de entidad relación utilizado para el diseño de este módulo.

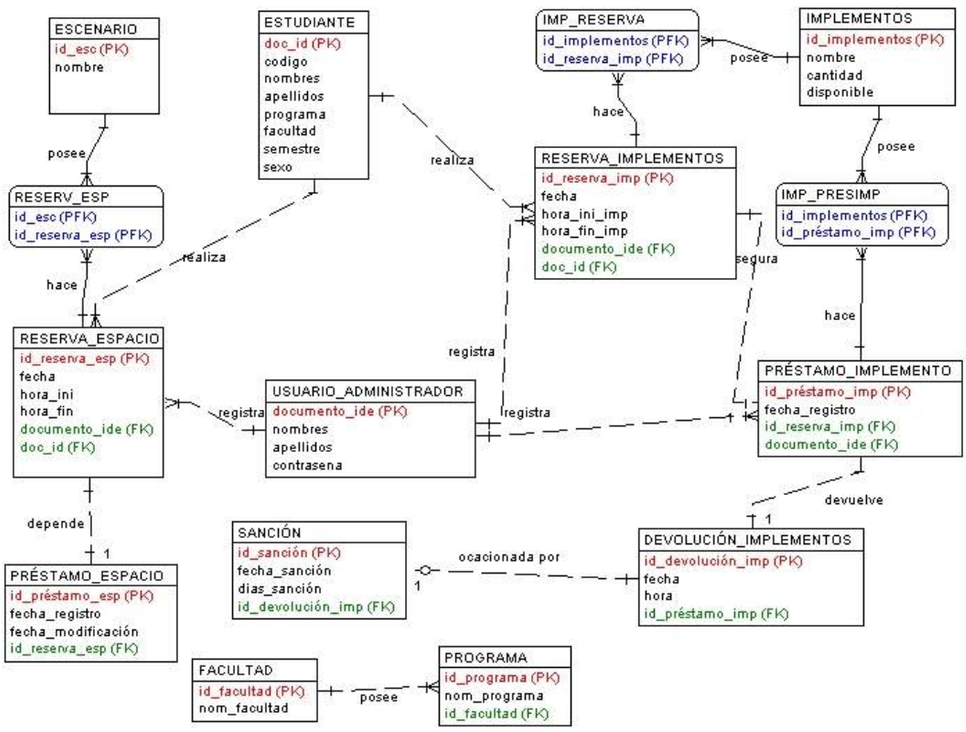

Figura 26: Diagrama de entidad relación.

6.2.3. Modelos funcionales

En los modelos funcionales se encuentran los casos de uso (escenarios y diagramas), los diagramas de transición de estados y la matriz crud.

6.2.3.1. Casos de Uso

A continuación se muestran cada caso de uso con sus respectivos escenarios y diagramas.

Escenarios: Es la descripción de los casos de uso, representados desde la Tabla 2 hasta la Tabla 7, y constan de:

- Nombre Caso de Uso
- Necesario/Deseable
- Actores
- Precondiciones
- Flujo Normal
- Caminos de Excepción

Diagramas: Es la representación gráfica de cada caso de uso y están representados desde la figura 27 hasta la figura 32.

Nombre: Administrar sistema de préstamos y reservas

Escenario:

• **Nombre Caso de Uso:**
Administrar sistema de préstamos y reservas
• **Necesario/Deseable:**
Necesario
• **Actores:**
Administrador
• **Precondiciones:**
1. Realizar validación y autenticación de usuario.
• **Resumen:**
Este es el escenario principal y el cual posee solo un actor administrador, el sistema muestra un menú de opciones con el cual, el usuario administrador es el encargado de registrar las reservas, préstamos, devoluciones y generar los reportes necesarios.
• **Flujo Normal:**
1. El usuario administrador ingresa su documento de identidad y contraseña, para que el sistema lo valide. 2. El sistema muestra un menú de opciones: • Reserva o Escenarios deportivos (para registrar las reservas de los espacios físicos deportivos). o Implementos deportivos (para registrar las reservas de los implementos deportivos). • Préstamo o Escenarios deportivos (para registrar los préstamos de los espacios físicos deportivos). o Implementos deportivos (para registrar los préstamos de los implementos deportivos). • Devolución (para registrar la devolución de los implementos deportivos) 3. El usuario administrador escoge la opción deseada. 4. El sistema muestra al usuario la vista inicial, que corresponde a la opción escogida.

• **Caminos de Excepción:**
2.1. Si el documento de identidad o contraseña del usuario es incorrecta, el sistema rechazará el ingreso del usuario y mostrará el mensaje, “Nombre o Contraseña que ingresó es Inválido”.

Tabla 2: Escenario administrar SIPRES.

Diagrama:

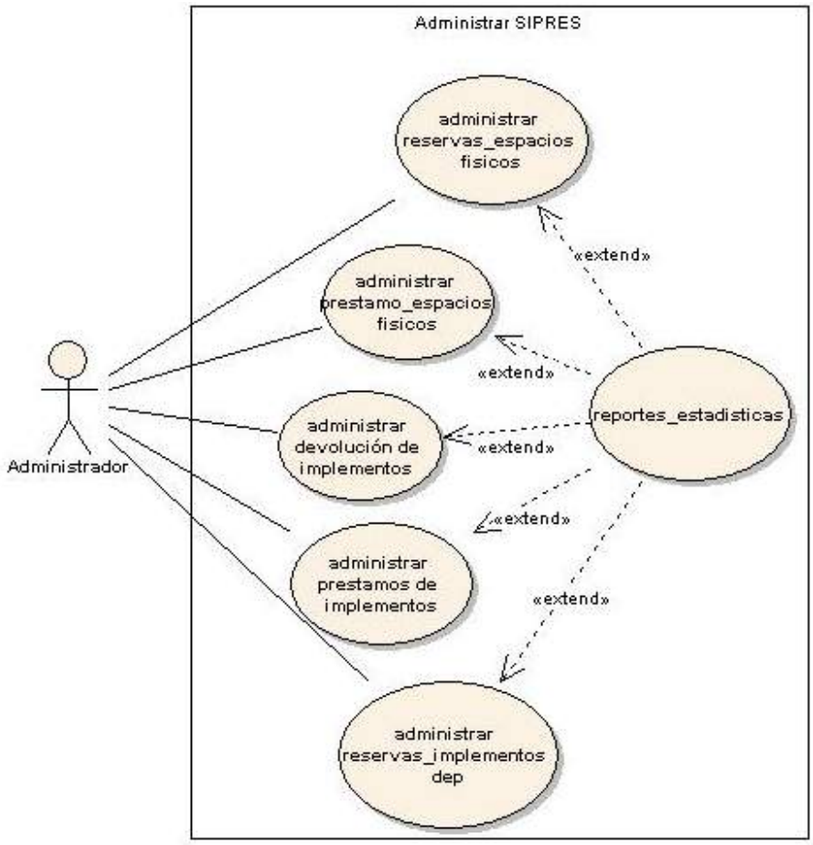

Figura 27: SIPRES (Sistema de Préstamos y Reservas).

Nombre: Administrar reserva de implementos deportivos

Escenario:

• **Nombre Caso de Uso:**
Administrar reserva de implementos deportivos
• **Necesario/Deseable:**
Necesario.
• **Actores:**
Administrador y Estudiante en línea.
• **Precondiciones:**
1. Realizar validación y autenticación de usuario.
• **Poscondiciones:**
El sistema SIPRES registrará en su base de datos, la información del estudiante, la fecha y el nombre del implemento deportivo que se reservó.

- **Resumen:**

Este escenario presenta dos actores los cuales son el administrador (administra el sistema) y el estudiante activo (solo puede hacer reservas en línea). Inmediatamente se ingrese el código del estudiante, ya sea el administrador o en línea, se comprobará si este tiene alguna sanción que le impida realizar la reserva, si el estudiante no tiene ningún problema se solicita la hora y el nombre del implemento a reservar, con lo cual se verificará que dicho implemento se encuentra disponible, en caso positivo el sistema registrará la reserva, si es el caso contrario el sistema rechazará la reserva.

- **Flujo Normal:**

1. El usuario escoge la opción "Reserva de Implementos Deportivos".
2. Ingresa los datos solicitados por el sistema:
   - Documento de identidad (del estudiante que solicita la reserva)
   - Implemento (Nombre del implemento a reservar)
   - Fecha (fecha a reservar)
   - Hora de inicio (hora inicial de la reserva)
   - Hora final (hora final de la reserva)
3. El sistema valida los datos ingresados y muestra un mensaje: "la reserva fue realizada con éxito".

- **Caminos de Excepción:**

2.1. El documento de identidad del usuario estudiante no se encuentra en la base de datos, el sistema rechaza la reserva y muestra el mensaje "Estudiante no se encuentra activo en la Universidad del Magdalena".

2.2. El estudiante se encuentra sancionado, el sistema rechaza la reserva y muestra el mensaje "El estudiante se encuentra sancionado".

2.3. Las existencias del implemento escogido ya están reservadas para la fecha y hora seleccionadas, el sistema muestra el mensaje "No hay existencias del implemento para la fecha y hora seleccionadas".

Tabla 3: Escenario reserva de implementos.

Diagrama:

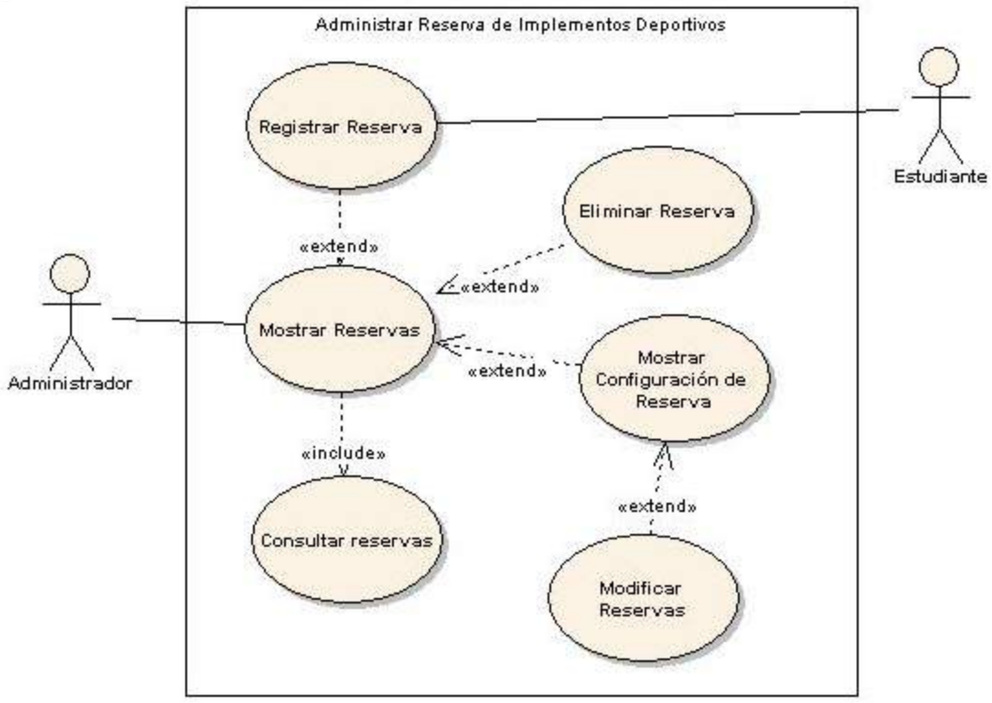

Figura 28: Reservar Implementos Deportivos.

Nombre: Administrar préstamos de implementos deportivos

Escenario:

• **Nombre Caso de Uso:**
Administrar préstamos de implementos deportivos
• **Necesario/Deseable:**
Necesario
• **Actores:**
Administrador
• **Precondiciones:**
1. Realizar validación y autenticación de usuario 2. Registrar reserva del implemento a prestar
• **Poscondiciones:**
El sistema SIPRES registrará en su base de datos, la información del estudiante, la fecha y el nombre del implemento deportivo que se prestó.
• **Resumen:**
El usuario administrador ingresa el código del estudiante y se comprobará si este está activo o si tiene alguna sanción que le impida realizar el préstamo, si el estudiante no tiene ningún problema se solicita la hora y el nombre del implemento a prestar, con lo cual se verificará que dicho implemento se encuentra disponible, en caso positivo el sistema registrará el préstamo, si es el caso contrario el sistema rechazará la operación.
• **Flujo Normal:**
1. El usuario escoge la opción “Préstamo de Implementos Deportivos”. 2. Ingresa los datos solicitados por el sistema: • Documento de identidad (del estudiante que solicita el préstamo). • Implemento (Nombre del implemento a prestar). • Hora de inicio (hora inicial del préstamo). • Hora final (hora final del préstamo). 3. El sistema valida los datos ingresados y muestra un mensaje: “El préstamo fue realizado con éxito”.
• **Caminos de Excepción:**
2.1.Si el escenario no ha sido reservado anteriormente por el estudiante, el sistema mostrará el mensaje, “El estudiante debe reservar el Escenario”

Tabla 4: Escenario préstamo de implemento.

Diagrama:

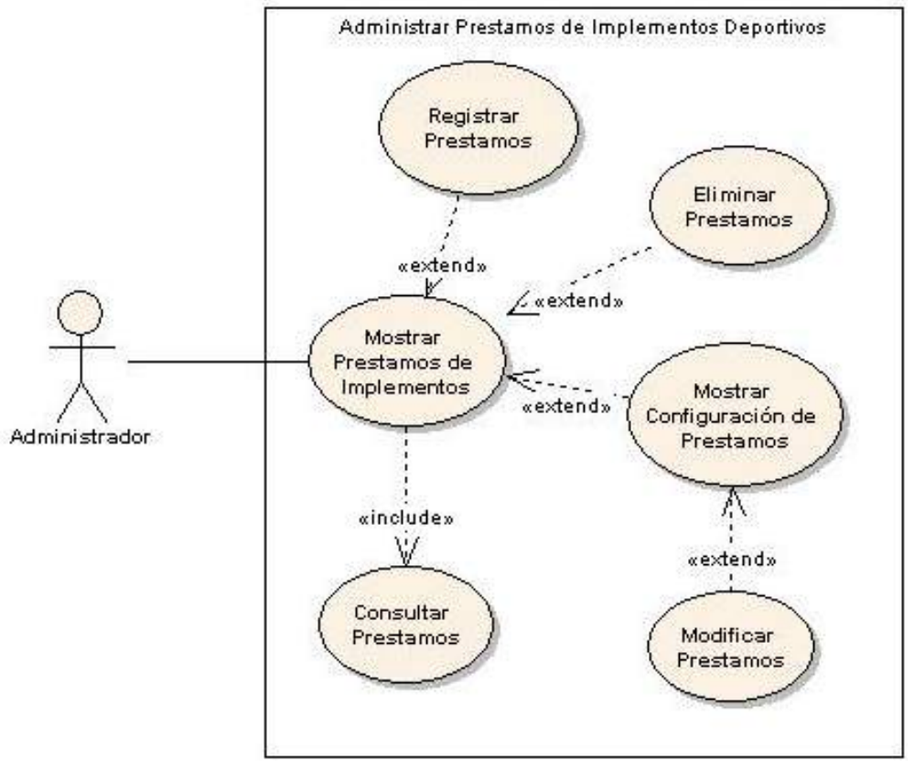

Figura 29: Prestar Implementos Deportivos.

Nombre: Administrar devolución de implementos deportivos

Escenario:

• **Nombre Caso de Uso:**
Administrar devolución de implementos deportivos
• **Necesario/Deseable:**
Necesario
• **Actores:**
Administrador
• **Precondiciones:**
1. Realizar validación y autenticación de usuario 2. Registrar en el sistema el préstamo del implemento a devolver.
• **Poscondiciones:**
El sistema SIPRES registrará en su base de datos, la información del estudiante, la fecha y el nombre del implemento deportivo que se devuelve y si es necesario aplicar alguna sanción, también se registrará en el sistema.
• **Resumen:**
Este escenario es muy sencillo ya que al devolver un implemento solo se requiere verificar el código del estudiante que realizó el préstamo del implemento a devolver, la fecha y hora de entrega, y si el día de entrega no es el mismo que el día del préstamo, el sistema establecerá sanciones dependiendo de los días de falta.
• **Flujo Normal:**
1. El usuario escoge la opción “Devolver Implementos Deportivos” 2. Ingresa los datos solicitados por el sistema: • Documento de identidad (del estudiante que solicitó el préstamo del

implemento a devolver) • Implemento (Nombre del implemento a devolver) Fecha (fecha de devolución del implemento) • Hora (hora de devolución del implemento) 3. El sistema valida los datos ingresados y muestra un mensaje: "La devolución fue realizado con éxito".
• **Caminos de Excepción:**
2.1. El estudiante excedió el límite de tiempo establecido para los préstamos, el sistema registra su código en la tabla de sanción y le establece un tiempo en el cual no se le podrá registrar reservas ni préstamos, inmediatamente el sistema muestra el mensaje "El estudiante excedió el límite de tiempo establecido para los préstamos y ha sido sancionado por "x" días".

Tabla 5: Escenario devolución de implemento.

Diagrama:

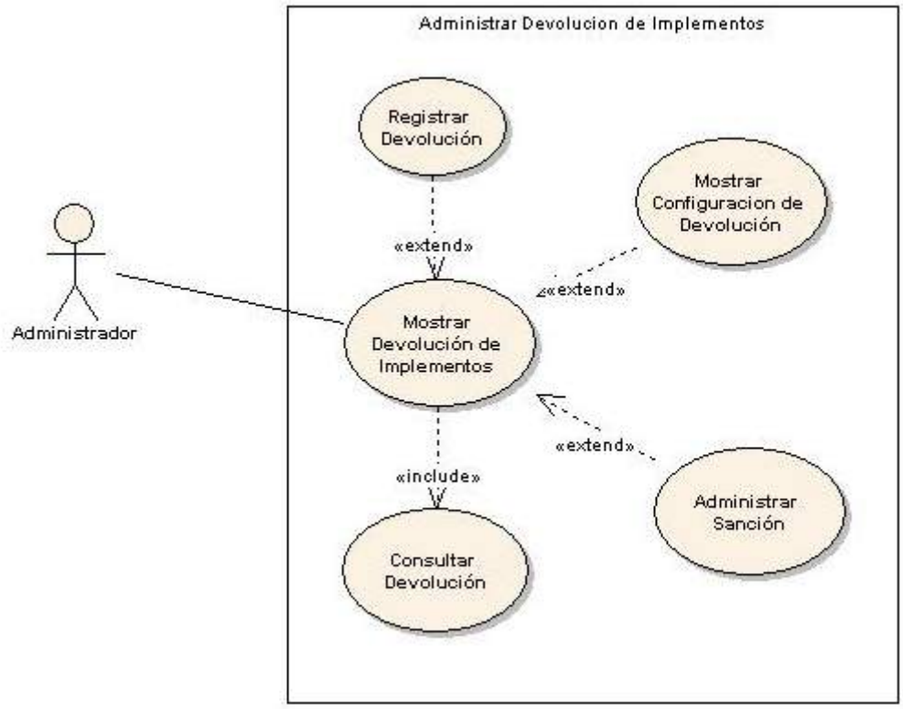

Figura 30: Devolver Implementos Deportivos.

Nombre: Administrar reserva de espacios físicos deportivos

Escenario:

• **Nombre Caso de Uso:**
Administrar reserva de espacios físicos deportivos
• **Necesario/Deseable:**
Necesario
• **Actores:**
Administrador y Estudiante en línea
• **Precondiciones:**
1. Realizar validación y autenticación de usuario
• **Poscondiciones:**
El sistema SIPRES registrará en su base de datos, la información del estudiante, la fecha y el nombre del espacio físico deportivo que se reservó.

- **Resumen:**

Este escenario presenta dos actores los cuales son el administrador (administrador del sistema) y el estudiante activo (solo puede hacer reservas en línea). Inmediatamente se ingrese el código del estudiante se comprobará si este tiene alguna sanción que le impida realizar la reserva, si el estudiante no tiene ningún problema se solicita la hora y el nombre del espacio físico a reservar, con lo cual se verificará que se encuentra disponible, en caso positivo el sistema registrará la reserva, si es el caso contrario el sistema rechazará la reserva.

- **Flujo Normal:**

1. El usuario escoge la opción "Reserva de Escenarios Deportivos"
2. Ingresa los datos solicitados por el sistema:
   - Documento de identidad (del estudiante que solicita la reserva)
   - Escenario (Nombre del espacio físico a reservar)
   - Fecha (fecha a reservar)
   - Hora de inicio (hora inicial de la reserva)
   - Hora final (hora final de la reserva)
3. El sistema valida los datos ingresados y muestra un mensaje: "La reserva fue realizada con éxito"

- **Caminos de Excepción:**

2.1. El documento de identidad del usuario estudiante no se encuentra en la base de datos, el sistema rechaza la reserva y muestra el mensaje "Estudiante no se encuentra activo en la Universidad del Magdalena".

2.2. El estudiante se encuentra sancionado, el sistema rechaza la reserva y muestra el mensaje "El estudiante se encuentra sancionado".

2.3. El espacio físico escogido ya se encuentra reservado para la fecha y hora seleccionada, el sistema muestra el mensaje "El escenario seleccionado ya se encuentra reservado".

Tabla 6: Escenario reserva de espacio.

Diagrama:

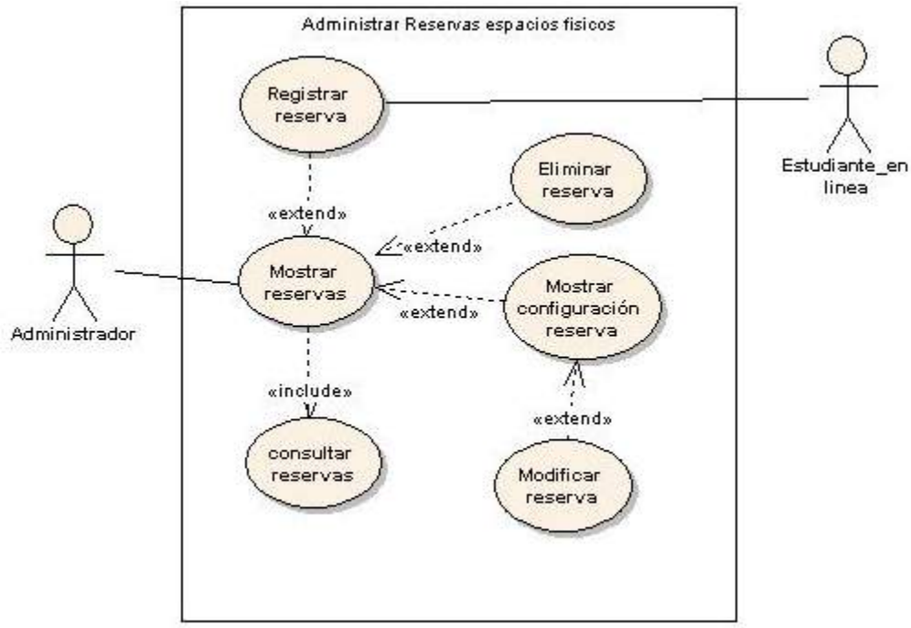

Figura 31: Reservar Espacios Físicos Deportivos.

Nombre: Administrar préstamo de espacios físicos deportivos

Escenario:

• **Nombre Caso de Uso:**
Administrar préstamo de espacios físicos deportivos
• **Necesario/Deseable:**
Necesario
• **Actores:**
Administrador
• **Precondiciones:**
1. Realizar validación y autenticación de usuario 2. Registrar la reserva del espacio a prestar
• **Poscondiciones:**
El sistema SIPRES registrará en su base de datos, la información del estudiante, la fecha y el nombre del espacio físico deportivo que se prestó.
• **Resumen:**
El usuario administrador ingresa el código del estudiante y el nombre del escenario a prestar, el sistema verifica que exista una reserva de este escenario para el estudiante que está solicitando el préstamo.
• **Flujo Normal:**
1. El usuario escoge la opción “Préstamo de escenarios Deportivos”. 2. Ingresa los datos solicitados por el sistema: • Documento de identidad (del estudiante que solicita el préstamo). • Escenario (Nombre del espacio físico a prestar). • Fecha (fecha del préstamo) • Hora de inicio (hora inicial del préstamo) • Hora final (hora final del préstamo) 3. El sistema valida los datos ingresados y muestra un mensaje: “El préstamo fue realizado con éxito”.
• **Caminos de Excepción:**
2.1 Si el escenario no ha sido reservado anteriormente por el estudiante, el sistema mostrará el mensaje, “El estudiante debe reservar el Escenario”

Tabla 7: Escenario préstamo de espacio.

Diagrama:

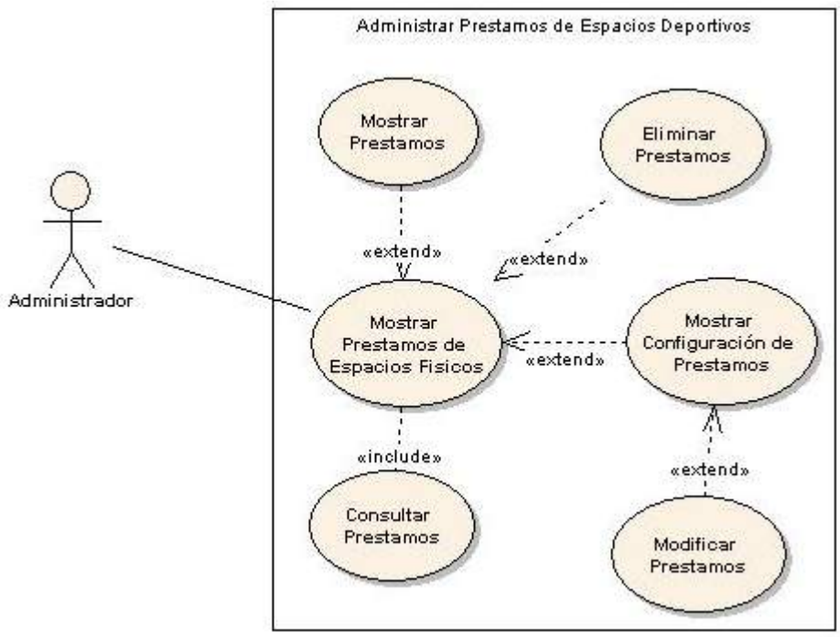

Figura 32: Prestar Espacios Físicos Deportivos.

6.2.3.2. Diagrama de transición de estados

En los diagramas de transición de estados se muestran los procesos de las reservas y préstamos de implementos y espacios físicos deportivos.

Reserva de implementos o espacios físicos deportivos

En la figura 33 se observa como es el proceso de las reservas, se comienza con la solicitud, es necesario ser estudiante activo, luego el sistema verifica si el estudiante está sancionado o no, si está sancionado rechaza la reserva, pero si no está sancionado sigue con el proceso, luego al escoger la opción implemento o espacio físico, se verifica si está disponible en la hora solicitada para la reserva, si lo está, realiza la reserva, sino lo está, rechaza la reserva.

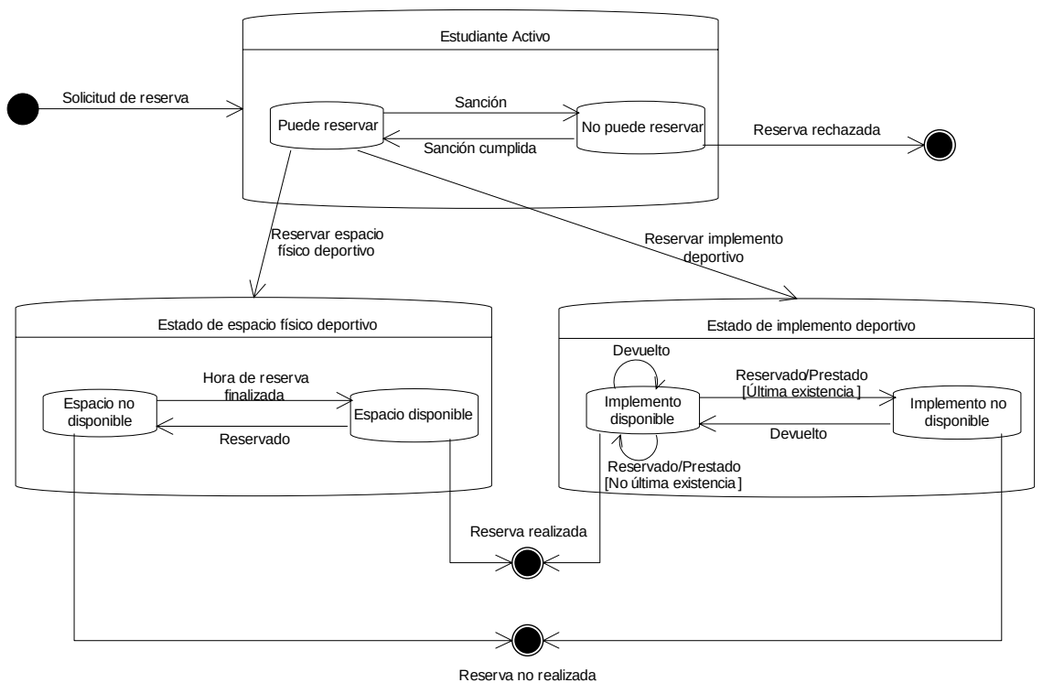

Figura 33: Diagrama de Transición de Estados para las Reservas.

Préstamos de implementos o espacios físicos deportivos
En la figura 34 se observa como es el proceso de los préstamos, se comienza con la solicitud, es necesario ser estudiante activo, luego el sistema verifica si el estudiante está sancionado o no, si está sancionado rechaza el préstamo, pero si no está sancionado sigue con el proceso, luego al escoger la opción implemento o espacio físico, se verifica si está disponible en la hora solicitada para el préstamo, si lo está, realiza el préstamo, sino lo está, rechaza el préstamo.

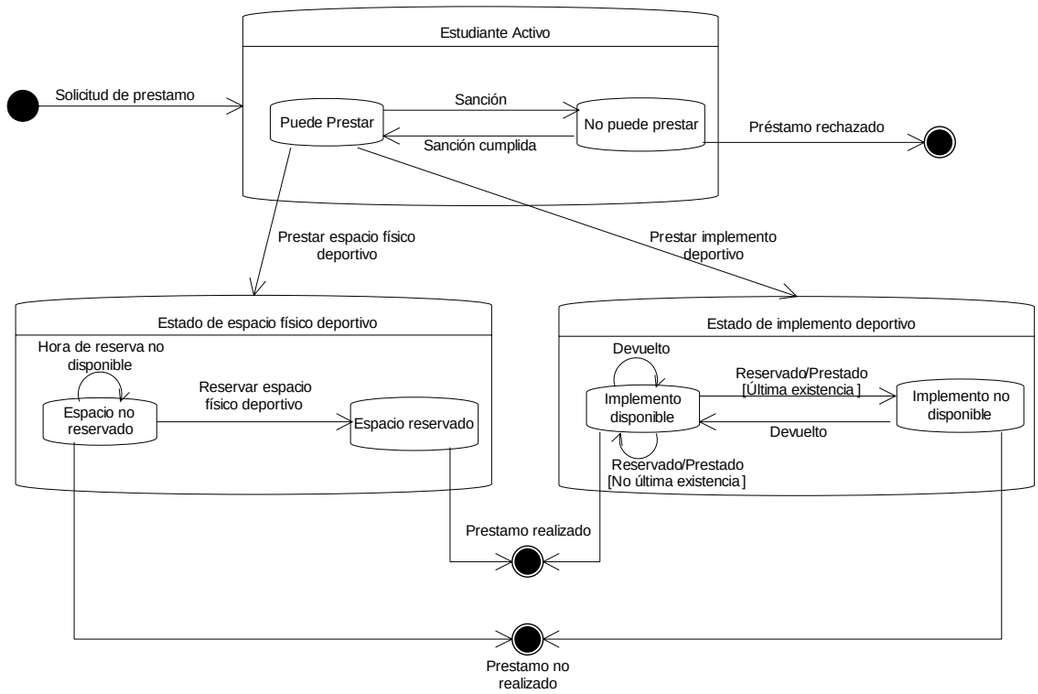

Figura 34: Diagrama de Transición de Estados para los Préstamos.

6.2.3.3. Matriz Crud
La matriz crud que se muestra en la tabla 8 muestra el uso de los datos por las funciones del sistema.

En las columnas de la matriz se especifican las funciones y en las filas se especifican las entidades, o más específicamente los atributos de éstas. En las celdas de la matriz van las letras iniciales de los tipos de operaciones que se pueden hacer en la entidad por una función: creación o inserción de datos, lectura, actualización o eliminación.

Entidad	Función / Atributos/Relación	Registrar reserva	Cancelar reserva	Registrar préstamo	Registra devolución	Generar informes
Reserva de espacios físicos deportivos		C	D			
	Nombre	R	D			R
	Fecha	R	D			R
	Hora	R	D			R
Reserva de implementos deportivos		C	D			
	Nombre	R	D			R
	Fecha	R	D			R
	Hora	R	D			R
Préstamo de espacios físicos deportivos				C		R
	Nombre			R		R
	Reserva			R		

Préstamo de implementos deportivos				C		
	Nombre			R	R	R
	Fecha			R	R	R
	Hora			R	R	R
Estudiante						
	Nombre					R
	Código	R	R	R	R	R
	Estado					R

Tabla 8: Matriz Crud.

Convenciones:
C: (Create) crear nuevo registro o insertar
R: (Retrieve) leer datos
U: (Update) actualizar
D: (Delete) eliminar registro leído

6.2.4. Estrategias de conversión o de comunicación de datos

Es importante recordar que este sistema es un módulo para el sistema de información para el área de deportes de la Universidad del Magdalena (SIADUM), del cual es necesario consultar algunas tablas en su base de datos, que tiene como nombre DEPORTES, y el nombre de las tablas son ESTUDIANTE, HORARIO_ENTRENAMIENTO, PROGRAMA y FACULTAD.

En las sentencias utilizadas para realizar estas consultas es necesario indicar el nombre de la base de datos y el nombre de la tabla a la que se quiere consultar, estas sentencias son las siguientes:
Select * from DEPORTE.ESTUDINTE
Select * from DEPORTE.HORARIO_ENTRENAMIENTO
Select * from DEPORTE.PROGRAMA
Select * from DEPORTE.FACULTAD

6.2.5. Glosario de términos

Caso de uso: Es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización software.

Diagrama de clases: Son utilizados durante el proceso de análisis y diseño de los sistemas informáticos, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Diagrama de entidad relación: Son una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

Diagrama de transición de estados: El diagrama de transición de estados, muestra el conjunto de posibles vidas de los objetos importantes del sistema originadas por los eventos que ocurren en el sistema real.

Un objeto permanece en un estado para hacer o permitir algunas acciones, o esperar hasta que ocurra otro evento que lo haga cambiar a otro estado posible.

Matriz crud: Muestra el uso de los datos por las funciones del sistema. En las columnas de la matriz se especifican las funciones y en las filas se especifican las entidades, o más específicamente los atributos de éstas.

Modelo de datos: Es aquel que describe de una forma abstracta cómo se representan los datos, sea en una empresa, en un sistema de información o en un sistema de gestión de base de datos.

6.3. ENTREGABLE DE LA FASE DE DISEÑO

6.3.1. Introducción

En este entregable se encuentra una información muy específica y detallada con respecto al sistema que se elaborará, entre estos, se establece el modelo lógico, plan de capacidad, diseño de la aplicación y el esquema de autorización, todo esto con el objetivo de poder construir el sistema, en base al trabajo realizado en esta fase.

6.3.2. Modelo lógico de la base de datos

El modelo lógico fue creado teniendo en cuenta el diagrama de clases presentado en la figura 25, que se encuentra en el modelo de datos del entregable de la fase de análisis.

Como fue mencionado anteriormente, seguido se puede apreciar desde la Tabla 9 hasta la Tabla 16, el modelo lógico establecido en el diagrama de clases.

Tabla usuario administrador

Columna	Descripción	Valor nulo	Tamaño	Tipo
contraseña	Código necesario para ingresar al sistema	NN	20	String
cédula	Número del documento de identidad del usuario administrador	NN	20	String
nombre	Nombre del usuario administrador	NN	50	String
apellido	Apellido del usuario administrador	NN	64	String
teléfono	Teléfono del usuario administrador		20	String

Tabla 9: Usuario_Administrador.

Tabla reserva de espacios

Columna	Descripción	Valor nulo	Tamaño	Tipo
id_reserva_esp	Número de la reserva a realizar	NN	20	String
fecha	Fecha en que se realiza la reserva	NN		Date
hora_ini	Hora a reservar para el inicio del préstamo	NN	2	String
hora_fin	Hora límite a reservar	NN	2	String
doc_id	Documento de identidad del estudiante que realiza la reserva	NN	20	String

Tabla 10: Reserva_Espacio.

Tabla reserva de implementos

Columna	Descripción	Valor nulo	Tamaño	Tipo
id_reserva_imp	Número de la reserva a realizar	NN	20	String
fecha	Fecha en que se realiza la reserva	NN		Date
hora_ini	Hora a reservar para el inicio del préstamo	NN	2	String
hora_fin	Hora límite a reservar	NN	2	String
doc_id	Documento de identidad del estudiante que realiza la reserva	NN	20	String

Tabla 11: Reserva_Implemento.

Tabla préstamo de espacios

Columna	Descripción	Valor nulo	Tamaño	Tipo
id_prestamo_esp	Número del préstamo a realizar	NN	20	String
fecha_registro	Fecha en que se registra el préstamo	NN		Date
fecha_modificacion	Fecha en que se realiza alguna modificación			Date

Tabla 12: Préstamo_Espacio.

Tabla préstamo de implementos

Columna	Descripción	Valor nulo	Tamaño	Tipo
id_prestamo_imp	Número del préstamo a realizar	NN	20	String
fecha_registro	Fecha en que se registra el préstamo	NN		Date
doc_id	Documento de identidad del estudiante que realiza la reserva	NN	20	String

Tabla 13: Préstamo_Implemento.

Tabla devolución de implementos

Columna	Descripción	Valor nulo	Tamaño	Tipo
id_devolucion_imp	Número de la devolución a realizar	NN	20	String
fecha	Fecha en que se registra la devolución	NN		Date
hora	Hora en que se registra la devolución	NN	20	String

Tabla 14: Devolución_Implemento.

Tabla espacio físico

Columna	Descripción	Valor nulo	Tamaño	Tipo
id_espacio	Número de identificación de cada escenario deportivo	NN	20	String
nombre	Nombre del escenario deportivo	NN	40	String
disponible	Estado del escenario deportivo	NN	3	String

Tabla 15: Espacios Físicos.

Tabla implemento

Columna	Descripción	Valor nulo	Tamaño	Tipo
id_implemento	Número de identificación del implemento deportivo	NN	20	String
Nombre	Nombre del implemento	NN	20	String
Cantidad	Cantidad existente del implemento	NN	3	String
Disponible	Cantidad disponible del implemento	NN	3	String

Tabla 16: Implementos.

6.3.3. Plan de capacidad

En este punto se indicarán los aspectos de rendimiento y concurrencia para que el sistema responda adecuadamente ante los servicios que debe ofrecer, a continuación, la tabla 17 hace referencia a la concurrencia esperada y la tabla 18 a los requerimientos de las transacciones.

6.3.3.1. Concurrencia esperada

Localización	**Número de usuarios concurrentes en horas picos**	**Número de usuarios concurrentes en horas normales**	**Total de usuarios**
Área de deportes	2000	1000	3000

Tabla 17: Concurrencia esperada.

6.3.3.2. Requerimiento de las transacciones

Transacción	**Tipo (en lote/en línea)**	**frecuencia**	**Prioridad (alta/media/baja)**	**Complejidad (alta/promedio/simple)**	**Duración máxima**	**Actividad en la base de datos**
Reserva de espacio	En línea	A solicitud	Alta	Simple	15 minutos	Inserción, consulta
Reserva de implemento	En línea	A solicitud	Alta	Simple	15 minutos	Inserción, consulta
Préstamo de espacio	En línea	A solicitud	Alta	Simple	15 minutos	Inserción, consulta
Préstamo de implemento	En línea	A solicitud	Alta	Simple	15 minutos	Inserción, consulta
Devolución de implemento	En línea	A solicitud	Alta	Simple	15 minutos	Inserción, consulta

Tabla 18: Requerimiento de las transacciones.

6.3.4. Diseño de la aplicación

Para el diseño de la aplicación se crearon los módulos de la aplicación en la Tabla 19 y los diagramas de secuencia desde la Figura 35 a la Figura 41.

6.3.4.1. Módulos de la aplicación

La tabla 19 muestra los módulos que hacen parte de esta aplicación.

Nombre	Tipo	Funcionalidad
Reserva de escenarios deportivos	Interfase	Inserción, modificación, consulta y eliminación de datos
Reserva de implementos deportivos	Interfase	Inserción, modificación, consulta y eliminación de datos
Préstamo de escenarios deportivos	Interfase	Inserción, modificación, consulta y eliminación de datos
Préstamo de implementos deportivos	interfase	Inserción, modificación, consulta y eliminación de datos
Devolución de implementos deportivos	Interfase	Inserción, modificación, consulta y eliminación de datos
Generación de informes	Informe	Mostrar una vista preliminar de los datos estadísticos referentes a las reservas, préstamos o devoluciones, ya que cada módulo consta de la opción de generar dicho informe

Tabla 19: Módulos de la aplicación.

6.3.4.2. Diagrama de secuencias

Desde la figura 35 hasta la 41 se encuentran los Diagramas de secuencias utilizados en el diseño de este módulo.

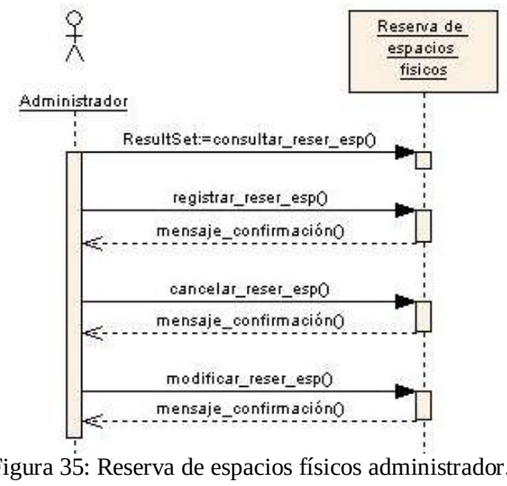

Figura 35: Reserva de espacios físicos administrador.

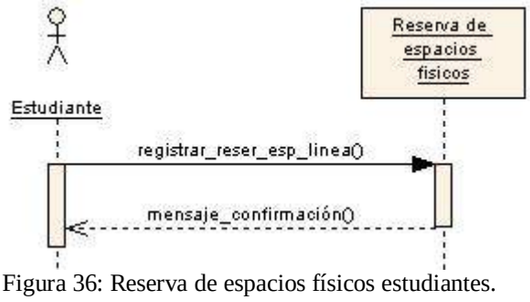

Figura 36: Reserva de espacios físicos estudiantes.

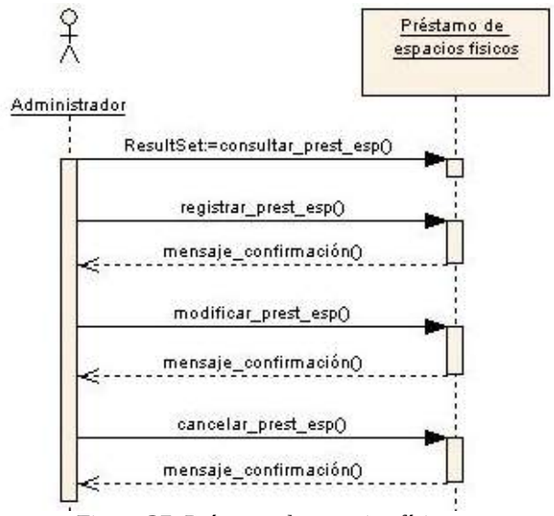

Figura 37: Préstamo de espacios físicos.

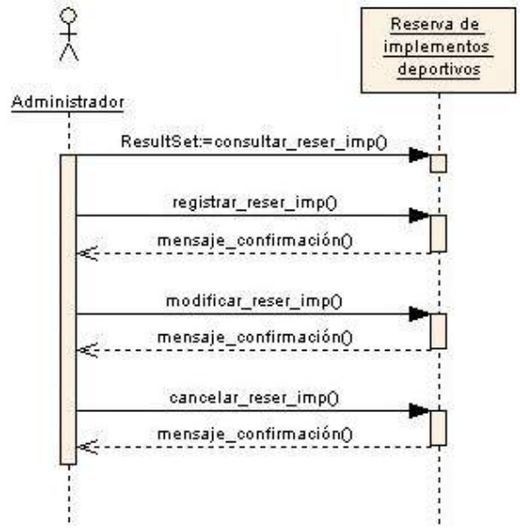

Figura 38: Reserva de implementos deportivos administrador.

Figura 39: Reserva de implementos deportivos estudiantes.

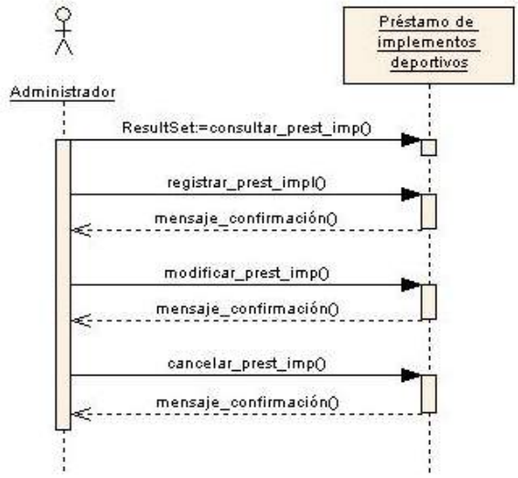

Figura 40: Préstamo de implementos deportivos.

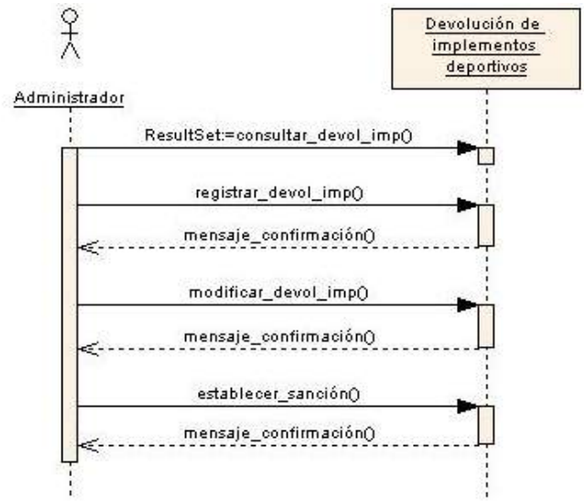

Figura 41: Devolución de implementos deportivos.

6.3.5. Esquema de autorización

Esta aplicación solo posee dos usuarios, los cuales son Administrador y Estudiantes.

6.3.5.1. Administrador: tiene todos los permisos sobre la base de datos, insertar, consultar, actualizar y eliminar.

6.3.5.2. Estudiante: solo tiene permisos para registrar los datos necesarios en las reserva en línea.

6.3.6. Modelo de pruebas del sistema

En la Tabla 20, Modelo de pruebas del sistema, se establecen en primera instancia las funciones, descripción, resultados esperados y datos requeridos, necesarios para realizar las primeras pruebas funcionales del sistema.

Función	Descripción	Resultado esperado	Datos requeridos
Reservar Escenario	Registrar reserva de escenario	Tener todos los datos solicitados para la nueva reserva, registrados en la base de datos	Tablas de espacios físicos, estudiantes, horarios de entrenamientos y reserva espacios creadas y actualizadas
Eliminar Reserva de Escenario	Eliminar reserva de escenario	Borrar los datos de la reserva registrada anteriormente	Tabla de reserva de espacio actualizada
Reservar Escenario	Registrar reserva de escenario	Tener todos los datos solicitados para la nueva reserva, registrados en la base de datos	Tablas de espacios físicos, estudiantes, horarios de entrenamientos y reserva espacios creadas y actualizadas
Prestar Escenario	Registrar préstamo del escenario reservado	Tener todos los datos solicitados para el nuevo préstamo, registrados en la base de datos	Tablas de espacios físicos, estudiantes, horarios de entrenamientos y reserva espacios creadas y actualizadas
Prestar Escenario	Registrar préstamo de un escenario no reservado	Mostrar mensaje de error, ya que es necesario que primero se realice la reserva de dicho escenario	Tablas de espacios físicos, estudiantes, horarios de entrenamientos y reserva espacios creadas y actualizadas
Reservar Implemento	Registrar reserva de implemento	Tener todos los datos solicitados para la nueva reserva, registrados en la base de datos	Tablas de implementos, estudiantes y reserva implementos creadas y actualizadas

Prestar Implemento	Registrar préstamo del implemento reservado	Tener todos los datos solicitados para el nuevo préstamo, registrados en la base de datos	Tablas de implementos, estudiantes, reserva implementos y préstamo implementos creadas y actualizadas
Devolver Implemento	Registrar devolución normal del implemento prestado	Tener todos los datos solicitados para la nueva devolución, registrados en la base de datos	Tablas de implementos, estudiantes, préstamo implementos y devolver implemento creadas y actualizadas
Reservar Implemento	Registrar reserva de implemento	Tener todos los datos solicitados para la nueva reserva, registrados en la base de datos	Tablas de implementos, estudiantes y reserva implementos creadas y actualizadas
Prestar Implemento	Registrar préstamo del implemento reservado	Tener todos los datos solicitados para el nuevo préstamo, registrados en la base de datos	Tablas de implementos, estudiantes, reserva implementos y préstamo implementos creadas y actualizadas
Devolver Implemento	Registrar devolución atrasada del implemento prestado	Tener todos los datos solicitados para la nueva devolución y la sanción correspondiente al atraso, registrados en la base de datos	Tablas de implementos, estudiantes, préstamo implementos y devolver implemento creadas y actualizadas
Prestar Implemento	Registrar préstamo de un implemento no reservado	Mostrar mensaje de error, ya que es necesario que primero se realice la reserva de dicho implemento	Tablas de implementos, estudiantes, reserva implementos y préstamo implementos creadas y actualizadas
Generar Informe	Generar informe de las reserva de escenarios	Mostrar una vista de las reservas de escenarios	Tablas de espacio físico, estudiante y reserva espacio, actualizadas
Generar Informe	Generar informe de las reserva de implementos	Mostrar una vista de las reservas de implementos	Tablas de implemento, estudiante y reserva implemento, actualizadas
Generar Informe	Generar informe de los préstamo de escenarios	Mostrar una vista de los préstamos de escenarios	Tablas de espacio físico, estudiante y préstamo espacio, actualizadas
Generar Informe	Generar informe de los préstamos de implementos	Mostrar una vista de los préstamos de implementos	Tablas de implemento, estudiante y préstamo implemento, actualizadas
Generar Informe	Generar informe de las devoluciones de implementos	Mostrar una vista de las devoluciones de implementos	Tablas de implemento, estudiante y devolver implemento, actualizadas

Tabla 20: Modelo de pruebas del sistema.

6.3.7. Estrategias para la transición

En este punto se encuentra el plan de transición y el plan de entrenamiento, con el fin de mostrar como se introducirá el nuevo sistema, de tal forma que se minimice el impacto que producirá el cambio,

6.3.7.1. Plan de Transición

El área de deportes de la Universidad del Magdalena realiza los procesos de reserva y préstamo manualmente, utilizando tablas de Excel para guardar su información, con la utilización del módulo SIPRES estos procesos tendrán un manejo sistematizado con herramientas propias para el manejo de entorno Web, facilitando el ingreso de los diferentes usuarios (personal de esta área y estudiantes) y el ingreso, almacenamiento y recuperación de los datos fundamentales para el desarrollo de los procesos dichos anteriormente.

6.3.7.2. Plan de Entrenamiento

Este módulo dispondrá de manuales de usuario y manuales técnicos completos en donde se especifican las diferentes herramientas y métodos utilizados para la consecución del mismo, guiando paso a paso a los usuarios de tal forma que se familiaricen con la aplicación y aprendan a manejarla.

6.3.8. Manual del usuario inicial

Esta es la primera documentación sobre el manual de usuario, y es la base para el entrenamiento de los nuevos usuarios.

6.4. ENTREGABLE DE LA FASE DE CONSTRUCCIÓN

6.4.1. Introducción

En este entregable se presenta el modelo físico de la base de datos, el código fuente de la aplicación y las pruebas necesarias para el funcionamiento esperado de este módulo.

6.4.2. Modelo físico de los datos

La base de datos de esta aplicación fue creada utilizando el Modelo de Entidad Relación mostrado en la figura 26, para realizar este modelo se utilizó como herramienta un programa llamado CASE Studio 2, con el cual se generaron los scripts para la Base de Datos.

Los scripts que se utilizaron para generar la base de datos de esta aplicación son los siguientes:

```
CREATE TABLE "ESCENARIO" (
	"ID_ESC" NVARCHAR2(20) NOT NULL ,
	"NOMBRE" NVARCHAR2(20) NOT NULL )
/

CREATE TABLE "IMPLEMENTOS" (
	"ID_IMPLEMENTOS" NVARCHAR2(20) NOT NULL ,
	"NOMBRE" NVARCHAR2(20) NOT NULL ,
	"CANTIDAD" NUMBER NOT NULL ,
	"DISPONIBLE" NUMBER NOT NULL )
/

CREATE TABLE "RESERVA_ESPACIO" (
	"ID_RESERVA_ESP" NVARCHAR2(20) NOT NULL ,
	"FECHA" DATE NOT NULL ,
	"HORA_INI" NUMBER NOT NULL ,
	"HORA_FIN" NUMBER NOT NULL ,
	"DOCUMENTO_IDE" NVARCHAR2(20),
	"DOC_ID" NVARCHAR2(20)  NOT NULL)
/

CREATE TABLE "RESERVA_IMPLEMENTOS" (
	"ID_RESERVA_IMP" NVARCHAR2(20) NOT NULL ,
	"FECHA" DATE NOT NULL ,
	"HORA_INI_IMP" INTEGER NOT NULL ,
	"HORA_FIN_IMP" INTEGER NOT NULL ,
	"DOCUMENTO_IDE" NVARCHAR2(20),
	"DOC_ID" NVARCHAR2(20) NOT NULL )
/

CREATE TABLE "PR_STAMO_ESPACIO" (
	"ID_PR_STAMO_ESP" NVARCHAR2(20) NOT NULL ,
	"FECHA_REGISTRO" DATE,
	"FECHA_MODIFICACI_N" DATE NOT NULL ,
	"ID_RESERVA_ESP" NVARCHAR2(20) NOT NULL )
/
```

```
CREATE TABLE "PR_STAMO_IMPLEMENTO" (
        "ID_PR_STAMO_IMP" NVARCHAR2(20) NOT NULL ,
        "FECHA_REGISTRO" DATE,
        "ID_RESERVA_IMP" NVARCHAR2(20) NOT NULL ,
        "DOCUMENTO_IDE" NVARCHAR2(20) NOT NULL )
/

CREATE TABLE "DEVOLUCI_N_IMPLEMENTOS" (
        "ID_DEVOLUCI_N_IMP" NVARCHAR2(20) NOT NULL ,
        "FECHA" DATE NOT NULL ,
        "HORA" NUMBER NOT NULL ,
        "ID_PR_STAMO_IMP" NVARCHAR2(20) NOT NULL )
/

CREATE TABLE "ESTUDIANTE" (
        "DOC_ID" NVARCHAR2(20) NOT NULL ,
        "CODIGO" NVARCHAR2(20) NOT NULL ,
        "NOMBRES" NVARCHAR2(20) NOT NULL ,
        "APELLIDOS" NVARCHAR2(50) NOT NULL ,
        "PROGRAMA" NVARCHAR2(25) NOT NULL ,
        "FACULTAD" NVARCHAR2(20) NOT NULL ,
        "SEMESTRE" NVARCHAR2(20) NOT NULL ,
        "SEXO" VARCHAR2(1) NOT NULL )
/

CREATE TABLE "USUARIO_ADMINISTRADOR" (
        "DOCUMENTO_IDE" NVARCHAR2(20) NOT NULL ,
        "NOMBRES" NVARCHAR2(50) NOT NULL ,
        "APELLIDOS" NVARCHAR2(64) NOT NULL )
/

CREATE TABLE "IMP_RESERVA" (
        "ID_IMPLEMENTOS" NVARCHAR2(20) NOT NULL ,
        "ID_RESERVA_IMP" NVARCHAR2(20) NOT NULL )
/

CREATE TABLE "IMP_PRESIMP" (
        "ID_IMPLEMENTOS" NVARCHAR2(20) NOT NULL ,
        "ID_PR_STAMO_IMP" NVARCHAR2(20) NOT NULL )
/


CREATE TABLE "RESERV_ESP" (
        "ID_ESC" NVARCHAR2(20) NOT NULL ,
        "ID_RESERVA_ESP" NVARCHAR2(20) NOT NULL )
/

CREATE TABLE "SANCI_N" (
        "ID_SANCI_N" NVARCHAR2(20) NOT NULL ,
        "FECHA_SANCI_N" DATE NOT NULL ,
        "DIAS_SANCI_N" NUMBER NOT NULL ,
        "ID_DEVOLUCI_N_IMP" NVARCHAR2(20) NOT NULL )
/

CREATE TABLE "FACULTAD" (
        "ID_FACULTAD" NVARCHAR2(20) NOT NULL ,
        "NOM_FACULTAD" VARCHAR2(50))
/

CREATE TABLE "PROGRAMA" (
        "ID_PROGRAMA" NVARCHAR2(20) NOT NULL ,
        "NOM_PROGRAMA" VARCHAR2(20),
        "ID_FACULTAD" NVARCHAR2(20) NOT NULL )
/



ALTER TABLE "ESCENARIO" ADD CONSTRAINT "PK_ESCENARIO" PRIMARY KEY ("ID_ESC")
/
ALTER TABLE "IMPLEMENTOS" ADD CONSTRAINT "PK_IMPLEMENTOS" PRIMARY KEY ("ID_IMPLEMENTOS")
/
ALTER TABLE "RESERVA_ESPACIO" ADD CONSTRAINT "PK_RESERVA_ESPACIO" PRIMARY KEY ("ID_RESERVA_ESP")
/
ALTER TABLE "RESERVA_IMPLEMENTOS" ADD CONSTRAINT "PK_RESERVA_IMPLEMENTOS" PRIMARY KEY
("ID_RESERVA_IMP")
/
ALTER TABLE "PR_STAMO_ESPACIO" ADD CONSTRAINT "PK_PR_STAMO_ESPACIO" PRIMARY KEY
("ID_PR_STAMO_ESP")
/
ALTER TABLE "PR_STAMO_IMPLEMENTO" ADD CONSTRAINT "PK_PR_STAMO_IMPLEMENTO" PRIMARY KEY
("ID_PR_STAMO_IMP")
/
ALTER TABLE "DEVOLUCI_N_IMPLEMENTOS" ADD CONSTRAINT "PK_DEVOLUCI_N_IMPLEMENTOS" PRIMARY KEY
("ID_DEVOLUCI_N_IMP")
/
ALTER TABLE "ESTUDIANTE" ADD CONSTRAINT "PK_ESTUDIANTE" PRIMARY KEY ("DOC_ID")
/
ALTER TABLE "USUARIO_ADMINISTRADOR" ADD CONSTRAINT "PK_USUARIO_ADMINISTRADOR" PRIMARY KEY
("DOCUMENTO_IDE")
/
ALTER TABLE "IMP_RESERVA" ADD CONSTRAINT "PK_IMP_RESERVA" PRIMARY KEY
("ID_IMPLEMENTOS","ID_RESERVA_IMP")
/
```

```
ALTER TABLE "IMP_PRESIMP" ADD CONSTRAINT "PK_IMP_PRESIMP" PRIMARY KEY ("ID_IMPLEMENTOS","ID_PR_STAMO_IMP")
/
ALTER TABLE "RESERV_ESP" ADD CONSTRAINT "PK_RESERV_ESP" PRIMARY KEY ("ID_ESC","ID_RESERVA_ESP")
/
ALTER TABLE "SANCI_N" ADD CONSTRAINT "PK_SANCI_N" PRIMARY KEY ("ID_SANCI_N")
/
ALTER TABLE "FACULTAD" ADD CONSTRAINT "PK_FACULTAD" PRIMARY KEY ("ID_FACULTAD")
/
ALTER TABLE "PROGRAMA" ADD CONSTRAINT "PK_PROGRAMA" PRIMARY KEY ("ID_PROGRAMA")
/


ALTER TABLE "RESERV_ESP" ADD CONSTRAINT "RESERVA_EL" FOREIGN KEY ("ID_ESC") REFERENCES "ESCENARIO" ("ID_ESC")
/


ALTER TABLE "IMP_RESERVA" ADD CONSTRAINT "RESERVA" FOREIGN KEY ("ID_IMPLEMENTOS") REFERENCES "IMPLEMENTOS" ("ID_IMPLEMENTOS")
/

ALTER TABLE "IMP_PRESIMP" ADD CONSTRAINT "PRESTA_A" FOREIGN KEY ("ID_IMPLEMENTOS") REFERENCES "IMPLEMENTOS" ("ID_IMPLEMENTOS")
/

ALTER TABLE "RESERV_ESP" ADD CONSTRAINT "SE_UTILIZA_PARA" FOREIGN KEY ("ID_RESERVA_ESP") REFERENCES "RESERVA_ESPACIO" ("ID_RESERVA_ESP")
/

ALTER TABLE "PR_STAMO_ESPACIO" ADD CONSTRAINT "DEPENDE" FOREIGN KEY ("ID_RESERVA_ESP") REFERENCES "RESERVA_ESPACIO" ("ID_RESERVA_ESP")
/

ALTER TABLE "IMP_RESERVA" ADD CONSTRAINT "HACE" FOREIGN KEY ("ID_RESERVA_IMP") REFERENCES "RESERVA_IMPLEMENTOS" ("ID_RESERVA_IMP")
/

ALTER TABLE "PR_STAMO_IMPLEMENTO" ADD CONSTRAINT "ASEGURA" FOREIGN KEY ("ID_RESERVA_IMP") REFERENCES "RESERVA_IMPLEMENTOS" ("ID_RESERVA_IMP")
/

ALTER TABLE "IMP_PRESIMP" ADD CONSTRAINT "SIRVE_PARA" FOREIGN KEY ("ID_PR_STAMO_IMP") REFERENCES "PR_STAMO_IMPLEMENTO" ("ID_PR_STAMO_IMP")
/

ALTER TABLE "DEVOLUCI_N_IMPLEMENTOS" ADD CONSTRAINT "DEVUELVE" FOREIGN KEY ("ID_PR_STAMO_IMP") REFERENCES "PR_STAMO_IMPLEMENTO" ("ID_PR_STAMO_IMP")
/

ALTER TABLE "SANCI_N" ADD CONSTRAINT "OCACIONADA_POR" FOREIGN KEY ("ID_DEVOLUCI_N_IMP") REFERENCES "DEVOLUCI_N_IMPLEMENTOS" ("ID_DEVOLUCI_N_IMP")
/

ALTER TABLE "RESERVA_IMPLEMENTOS" ADD CONSTRAINT "REALIZA" FOREIGN KEY ("DOC_ID") REFERENCES "ESTUDIANTE" ("DOC_ID")
/

ALTER TABLE "RESERVA_ESPACIO" ADD CONSTRAINT "REALIZA_EN_LINEA" FOREIGN KEY ("DOC_ID") REFERENCES "ESTUDIANTE" ("DOC_ID")
/

ALTER TABLE "RESERVA_IMPLEMENTOS" ADD CONSTRAINT "REGISTRA_LA" FOREIGN KEY ("DOCUMENTO_IDE") REFERENCES "USUARIO_ADMINISTRADOR" ("DOCUMENTO_IDE")
/

ALTER TABLE "RESERVA_ESPACIO" ADD CONSTRAINT "REGISTRA" FOREIGN KEY ("DOCUMENTO_IDE") REFERENCES "USUARIO_ADMINISTRADOR" ("DOCUMENTO_IDE")
/

ALTER TABLE "PR_STAMO_IMPLEMENTO" ADD CONSTRAINT "REGISTRA_EL" FOREIGN KEY ("DOCUMENTO_IDE") REFERENCES "USUARIO_ADMINISTRADOR" ("DOCUMENTO_IDE")
/

ALTER TABLE "PROGRAMA" ADD CONSTRAINT "POSEE" FOREIGN KEY ("ID_FACULTAD") REFERENCES "FACULTAD" ("ID_FACULTAD")
/
```

6.4.3. Código de la aplicación

En este documento solo se muestran las clases utilizadas para la programación de esta aplicación, ya que el código completo se encuentra en medio magnético adjunto a este documento.

Las clases de esta aplicación son:

```
/**
 *Esta Clase es para la conexion a la base de datos del sistema de
 *Prestamo y Reservas de Implementos y escenarios deportivos
 *
 */


package BaseDatos;

import java.sql.*;
import java.io.*;

public class conexion{

  String dbURL = "jdbc:oracle:thin:@localhost:1521:prestam";
  String user = "prestamo";
  String contrasena = "prestamo";
  private Connection dbCon;

 /******************conecta*********************/
  public Connection conectar() throws ClassNotFoundException,SQLException,Exception
        {
                Class.forName("oracle.jdbc.driver.OracleDriver");
         DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
         dbCon = DriverManager.getConnection(dbURL,user,contrasena);

         return dbCon;
        }
  /***********************************************/
  /**************desconecta**********************/
  public void desconectar() throws ClassNotFoundException,SQLException,Exception
       {
              dbCon.close();
       }
  /***********************************************/

}




/**
* clase para validar usuarios
*/

package usuario;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;

public class Usu_Administrador implements Serializable
{

            private String contrasena;
            private String cedula;
            private String nombre;
            private String apellido;
            private String telefono;
            private String reserva_implementos;
            private String reserva_espacio;
            private String prestamo_espacio;
```

```
	private String prestamo_implementos;
	private String devolucion_implementos;

public void setContrasena(String contrasena)
	{
		this.contrasena = contrasena;
	}

	public String getContrasena()
	{
		return contrasena;
	}


	public void setCedula(String cedula)
	{
		this.cedula = cedula;
	}

	public String getCedula()
	{
		return cedula;
	}

	public void setNombre(String nombre)
	{
		this.nombre = nombre;
	}

	public String getNombre()
	{
		return nombre;
	}


	public void setApellido(String apellido)
	{
		this.apellido = apellido;
	}

	public String getApellido()
	{
		return apellido;
	}


	public void setTelefono(String telefono)
	{
		this.telefono = telefono;
	}

	public String getTelefono()
	{
		return telefono;
	}


	public void setReserva_implementos(String reserva_implementos)
	{
		this.reserva_implementos = reserva_implementos;
	}

	public String getReserva_implementos()
	{
		return reserva_implementos;
	}


	public void setReserva_espacio(String reserva_espacio)
	{
		this.reserva_espacio = reserva_espacio;
	}

	public String getReserva_espacio()
	{
```

```
		return reserva_espacio;
	}


	public void setPrestamo_espacio(String prestamo_espacio)
	{
		this.prestamo_espacio = prestamo_espacio;
	}

	public String getPrestamo_espacio()
	{
		return prestamo_espacio;
	}


	public void setPrestamo_implementos(String prestamo_implementos)
	{
		this.prestamo_implementos = prestamo_implementos;
	}

	public String getPrestamo_implementos()
	{
		return prestamo_implementos;
	}

	public void setDevolucion_implementos(String devolucion_implementos)
	{
		this.devolucion_implementos = devolucion_implementos;
	}

	public String getDevolucion_implementos()
	{
		return devolucion_implementos;
	}

/**********************************metodo para validar usuario**************************/
	public boolean validar_usuario(String tipo) throws
ClassNotFoundException,SQLException,Exception
	{

	 conexion con = new conexion();
	 PreparedStatement ps;
	 Connection conec = con.conectar();
	 ResultSet rs = null;
	 String sql = new String();
	 boolean ret;
	  ret = true;
	 /**********************************validando si es administrador o estudiante***********/
    if(tipo.equals("1"))
	 sql = "select * from usuario_administrador where documento_ide=? "+
	     "and contrasena=? ";
	  else

	 sql = "select * from estudiante where doc_id=? and codigo=?";

/*****************************************************************************************
*/	 ps = conec.prepareStatement(sql) ;
	 ps.setString(1,this.cedula);
	 ps.setString(2,this.contrasena);
	 rs = ps.executeQuery();

     if(!rs.next())
	 ret = false;
	rs.close();
	ps.close();
	conec.close();
	con.desconectar();

	 return ret;
	 }

/*****************************************************************************************
*/

  public void cambiarContrasena()throws ClassNotFoundException,SQLException,Exception
```

```
        {

    conexion con = new conexion();
            PreparedStatement ps;
            Connection conec = con.conectar();
            String sql = new String();

            sql = "update usuario_administrador set contrasena=? where documento_ide=?";
            /*********************************************************************/
            ps = conec.prepareStatement(sql);

    ps.setString(1,this.contrasena);
    ps.setString(2,this.cedula);

    ps.executeUpdate();
    /*********************************************************************/
    ps.close();
    conec.close();
    con.desconectar();
            }


/*****************************************************************************************/

public static void main (String[] args)throws ClassNotFoundException,SQLException,Exception {

            Usu_Administrador us = new Usu_Administrador();

            us.setCedula("1");
            us.setContrasena("100");
            ///System.out.print(us.validar_usuario("1"));
            us.cambiarContrasena();

}

}
```

```
/**
* clase para administrar la devolucion de implementos deportivos
*/

package devolucion;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;
import prestamo.Prestamo_Implementos;


public class Devolucion_Implementos implements Serializable
{

	private String id_devolucion_imp;
	private String fecha;
	private String hora;
	private String prestamo_implementos;
	private conexion con;
	/***********************************/

   public void setId_devolucion_imp(String id_devolucion_imp)
	{
		this.id_devolucion_imp = id_devolucion_imp;
	}

	public String getId_devolucion_imp()
	{
		return id_devolucion_imp;
	}


	public void setFecha(String fecha)
	{
		this.fecha = fecha;
	}

	public String getFecha()
	{
		return fecha;
	}

	public void setHora(String hora)
	{
		this.hora = hora;
	}

	public String getHora()
	{
		return hora;
	}


	public void setPrestamo_implementos(String prestamo_implementos)
	{
		this.prestamo_implementos = prestamo_implementos;
	}

	public String getPrestamo_implementos()
	{
		return prestamo_implementos;
	}

/***********************************/

/************metodo para registrar devolucion**********/
	public void registrar_devol_imp() throws ClassNotFoundException,SQLException,Exception
   {
   /*****************************************************/
   conexion con = new conexion();
```

```
            PreparedStatement ps;
            Connection conec = con.conectar();
            String sql = new String();
    ResultSet rs = null;
    /*******************************************************/
    String ultimo = "select nvl((to_number(max(id_devoluci_n_imp))+1),0) as ultimo from
DEVOLUCI_N_IMPLEMENTOS";
    PreparedStatement psPrim = conec.prepareStatement(ultimo) ;
    rs = psPrim.executeQuery();
    rs.next();
    ultimo =  rs.getString("ultimo");
    this.id_devolucion_imp = ultimo;

    /*******************************************************/
    sql="insert into DEVOLUCI_N_IMPLEMENTOS values(?,sysdate,"+
                    "to_number(to_char(localtimestamp,'hh24')),?)";

    ps = conec.prepareStatement(sql);
    ps.setString(1,this.id_devolucion_imp);
    ps.setString(2,this.prestamo_implementos);

    ps.executeUpdate();

    /*******************************************************/
     ps.close();
     psPrim.close();
     conec.close();
     con.desconectar();

            }

/************metodo para establecer sancion**********/
            public void establecer_sancion()
            {

            }

/************metodo para consultar devolucion**********/
            /*public Vector consultar_devol_imp() throws ClassNotFoundException,SQLException,Exception
            {
              Vector v = new Vector();
     ResultSet rs = con.consultar("select * from DEVOLUCI_N_IMPLEMENTOS where
id_devoluci_n_imp='"+this.id_devolucion_imp+"'");

     while(rs.next())
     {
            Devolucion_Implementos di = new Devolucion_Implementos();
            di.setId_devolucion_imp("id_devoluci_n_imp");
            di.setFecha(rs.getString("fecha"));
            di.setHora(rs.getString("hora"));
            di.prestamo_implementos.setId_prestamo_imp(rs.getString("id_pr_stamo_imp"));
            v.add(di);
     }

     return v;
            }

/************metodo para modificar devolucion**********/
            /*public void modificar_devol_imp()throws ClassNotFoundException,SQLException,Exception
            {
     con.modificar("update DEVOLUCI_N_IMPLEMENTOS set fecha=sysdate,hora='"+this.hora+"' "+
                    "where id_devoluci_n_imp='"+this.id_devolucion_imp+"'");
            }
/*******************************************************/
/*****************metodo para consultar devoluviones***********/
public String consultar_devoluviones()
{
            return "select * from DEVOLUCI_N_IMPLEMENTOS";
}
/*************************************************************/

}


/**
```

```
* clase para administrar los espacios fisicos deportivos
*/

package escenario;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;

public class Espacio_Fisicos implements Serializable
{

	private String id_espacio;
	private String nombre;
	private String disponible;
	private conexion con;

	public void setId_espacio(String id_espacio)
	{
		this.id_espacio = id_espacio;
	}

	public String getId_espacio()
	{
		return id_espacio;
	}


	public void setNombre(String nombre)
	{
		this.nombre = nombre;
	}

	public String getNombre()
	{
		return nombre;
	}


   public void setDisponible(String disponible)
	{
		this.disponible = disponible;
	}

	public String getDisponible()
	{
		return disponible;
	}

	/*******metodo para consultar escenarios deportivos******/
	public Vector consultar_escenarios() throws ClassNotFoundException,SQLException,Exception
	{
		conexion con = new conexion();
	   PreparedStatement ps;
	   Connection conec = con.conectar();
	   ResultSet rs = null;
	   String sql = new String();

		Vector escenarios = new Vector();
		sql="select * from escenario";

		ps = conec.prepareStatement(sql);
	rs = ps.executeQuery();

		/*****************************************/
		while(rs.next())
		{
		Espacio_Fisicos Ep = new Espacio_Fisicos();
		Ep.setId_espacio(rs.getString("id_esc"));
		Ep.setNombre(rs.getString("nombre"));
	   escenarios.add(Ep);
		}
		/*****************************************/
		rs.close();
```

```
        ps.close();
        conec.close();
        con.desconectar();

                return  escenarios;
        }
        /*******************************************************/

        public static void main(String [] arg) throws ClassNotFoundException,SQLException,Exception
        {
                Espacio_Fisicos ep = new Espacio_Fisicos();

                Vector  v = ep.consultar_escenarios();

                  for(int i=0; i<v.size(); i++)
                        {
                         System.out.print(((Espacio_Fisicos)v.elementAt(i)).getNombre());
                        }

        }
        /*******************************************************/
}


/**
* clase para administrar los implementos deportivos
*/

package implemento;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;


public class Implementos implements Serializable
{

        private String id_implemento;
        private String nombre;
        private String cantidad;
        private String disponible;

        /*********************************/
    public void setId_implemento(String id_implemento)
        {
                this.id_implemento = id_implemento;
        }

        public String getId_implemento()
        {
                return id_implemento;
        }


    public void setNombre(String nombre)
        {
                this.nombre = nombre;
        }

        public String getNombre()
        {
                return nombre;
        }

    public void setCantidad(String cantidad)
        {
                this.cantidad = cantidad;
        }

        public String getCantidad()
        {
                return cantidad;
        }
```

```
    public void setDisponible(String disponible)
        {
            this.disponible = disponible;
        }

        public String getDisponible()
        {
            return disponible;
        }
/******************metodo para consultar listado de implementos deportivos**********/
        public Vector consultar_listado_implementos() throws
ClassNotFoundException,SQLException,Exception
        {
          conexion con = new conexion();
          PreparedStatement ps;
          Connection conec = con.conectar();
          ResultSet rs = null;
          String sql = new String();
          Vector v = new Vector();

            sql = "select * from implementos";

     ps = conec.prepareStatement(sql);
        rs = ps.executeQuery();

            while(rs.next())
            {
             Implementos i = new Implementos();
             i.setId_implemento(rs.getString("id_implementos"));
             i.setNombre(rs.getString("nombre"));
             i.setCantidad(rs.getString("cantidad"));
             i.setDisponible(rs.getString("disponible"));
             v.add(i);
            }

    rs.close();
    ps.close();
    conec.close();
    con.desconectar();

    return v;

        }
/*****************************************************************************/
public void agregarImplemento()throws ClassNotFoundException,SQLException,Exception
{
          conexion con = new conexion();
          PreparedStatement ps;
          Connection conec = con.conectar();
          String sql = new String();
          ResultSet rs = null;

          String ultimo = "select nvl((to_number(max(id_implementos))+1),0) as ultimo from
IMPLEMENTOS";
          ps = conec.prepareStatement(ultimo);
          rs = ps.executeQuery();
    rs.next();
    ultimo =  rs.getString("ultimo");



          sql="insert into implementos values('"+ultimo+"','"+
                this.nombre+"','"+this.cantidad+"','"+this.disponible+"')" ;


          ps = conec.prepareStatement(sql);
    ps.executeUpdate();

   rs.close();
   ps.close();
   conec.close();
   con.desconectar();

}
```

```
public static void main(String arg[]) throws ClassNotFoundException,SQLException,Exception
{
/*	Implementos i = new Implementos();
	ResultSet rs = i.consultar_listado_implementos();

	while(rs.next())
		System.out.print(rs.getString("nombre"));*/
}

}

/**
* clase para administrar el prestamo de espacios deportivos
*/

package prestamo;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;
import reserva.Reserva_Espacio;


public class Prestamo_Espacio implements Serializable {

	private String id_prestamo_esp;
	private String fecha_registro;
	private String reserva_espacio;

	/**********************************************/
	public void setId_prestamo_esp(String id_prestamo_esp)
	{
		this.id_prestamo_esp = id_prestamo_esp;
	}

	public String getId_prestamo_esp()
	{
		return id_prestamo_esp;
	}



	public void setFecha_registro(String fecha_registro)
	{
		this.fecha_registro = fecha_registro;
	}

	public String getFecha_registro()
	{
		return fecha_registro;
	}


	public void setReserva_espacio(String reserva_espacio)
	{
		this.reserva_espacio = reserva_espacio;
	}

	public String getReserva_espacio()
	{
		return reserva_espacio;
	}
	/**********************************************/

	/**********************metodo para consultar prestamo**********/
public String consultar_prest_esc(String orden) throws ClassNotFoundException,SQLException,Exception
	{
    return                                                                                                  "select
resc.id_esc,re.id_reserva_esp,pe.id_PR_STAMO_ESP,e.DOC_ID,e.APELLIDOS,e.NOMBRES,"+
"re.HORA_INI,re.HORA_FIN,pe.fecha_registro, "+
"(select  es.nombre from reserv_esp r "+
"inner join escenario es "+
"on r.ID_ESC=es.id_esc "+
"where r.ID_RESERVA_ESP=pe.id_reserva_esp "+
```

```
") as nombre, "+
"(select r_e.fecha from reserva_espacio r_e "+
"inner join pr_stamo_espacio p_e on r_e.ID_RESERVA_esp=p_e.ID_RESERVA_esp "+
"where p_e.id_pr_stamo_esp=pe.id_pr_stamo_esp) as fecha_RESERVA "+
"from reserva_espacio re "+
"inner join  PR_STAMO_ESPACIO pe on re.ID_RESERVA_ESP=pe.ID_RESERVA_ESP "+
"inner join reserv_esp resc on re.ID_RESERVA_ESP=resc.ID_RESERVA_ESP "+
"inner join estudiante e on re.doc_id=e.doc_id " +orden;
            }
            /*****************************************************************/

            /**********metodo para registrar prestamo de espacio**********/
            public void registrar_prest_esp() throws ClassNotFoundException,SQLException,Exception
            {

              conexion con = new conexion();
              PreparedStatement ps;
              Connection conec = con.conectar();
              ResultSet rs = null;
              String sql = new String();

              String ultimo = "select nvl(max(to_number(id_pr_stamo_esp))+1,0) as ultimo from
PR_STAMO_ESPACIO";
     PreparedStatement psPrim = conec.prepareStatement(ultimo) ;
     rs = psPrim.executeQuery();
     rs.next();
     ultimo =  rs.getString("ultimo");
     this.id_prestamo_esp=ultimo;

              sql = "insert into PR_STAMO_ESPACIO values(?,sysdate,"+
                "?)";

     ps = conec.prepareStatement(sql);
     ps.setString(1,this.id_prestamo_esp);
     ps.setString(2,this.reserva_espacio);
              ps.executeUpdate();
              /*********************************************************************/
     rs.close();
     ps.close();
     psPrim.close();
     conec.close();
     con.desconectar();

            }
            /**********metodo para modificar prestamo de espacio**********/
/*          public void modificar_prest_esp() throws ClassNotFoundException,SQLException,Exception
            {
       con.modificar("update PR_STAMO_ESPACIO set fecha_registro=sysdate,fecha_modificaci_n=sysdate
"+
                    "where id_pr_stamo_esp='"+this.id_prestamo_esp+"'");
            }
            /**********metodo para consultar prestamo de espacio**********/
            /*public Vector consultar_prest_espa() throws ClassNotFoundException,SQLException,Exception
            {
                    Vector v = new Vector();

                    ResultSet rs = con.consultar("select * from "+
                "PR_STAMO_ESPACIO "+
                "where id_pr_stamo_esp='"+this.id_prestamo_esp+"'");

       while(rs.next())
       {
          Prestamo_Espacio pe = new Prestamo_Espacio();
         pe.setId_prestamo_esp(rs.getString("id_pr_stamo_esp"));
         pe.setFecha_registro(rs.getString("fecha_registro"));
         pe.setFecha_modificacion(rs.getString("fecha_mofificaci_n"));
         pe.reserva_espacio.setId_reserva_esp(rs.getString("id_reserva_esp"));

         v.add(pe);

       }
        return v;
            }
            /**********metodo para cancelar prestamo de espacio**********/
            public void cancelar_prest_esp() throws ClassNotFoundException,SQLException,Exception
```

```
            {
       conexion con = new conexion();
              PreparedStatement ps;
              Connection conec = con.conectar();
              String sql = new String();

              sql = "delete from PR_STAMO_ESPACIO where id_pr_stamo_esp='"+this.id_prestamo_esp+"'";
              ps = conec.prepareStatement(sql);
              ps.executeUpdate();

     /*****************************************************************/
     ps.close();
     conec.close();
     con.desconectar();
            }
   /*************************************************************/
     public static void main (String[] args) throws ClassNotFoundException,SQLException,Exception {
    Prestamo_Espacio p = new Prestamo_Espacio();
  p.setId_prestamo_esp("1");
  p.cancelar_prest_esp();
    }

}

/**
* clase para administrar el prestamo de implementos deportivos
*/

package prestamo;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;
import reserva.Reserva_Implementos;


public class Prestamo_Implementos implements Serializable
   {

    /*************atributos************/
            private String id_prestamo_imp;
            private String fecha_registro;
            private String documento_ide;
            private String reserva_implemento;
            private String implemento;
    /***********************************/
            public void setId_prestamo_imp(String id_prestamo_imp)
            {
                    this.id_prestamo_imp = id_prestamo_imp;
            }

            public String getId_prestamo_imp()
            {
                    return id_prestamo_imp;
            }

            public void setFecha_registro(String fecha_registro)
            {
                    this.fecha_registro = fecha_registro;
            }

            public String getFecha_registro()
            {
                    return fecha_registro;
            }

    public void setDocumento_ide(String documento_ide)
            {
                    this.documento_ide = documento_ide;
            }

            public String getDocumento_ide()
            {
                    return documento_ide;
```

```
        }


        public void setReserva_implemento(String reserva_implemento)
        {
                this.reserva_implemento = reserva_implemento;
        }

        public String getReserva_implemento()
        {
                return reserva_implemento;
        }

        public void setImplemento(String implemento)
        {
                this.implemento = implemento;
        }

        public String getImplemento()
        {
                return implemento;
        }
        /**********************metodo para modificar prestamo**************/
        /*public void modificar_prest_imp() throws ClassNotFoundException,SQLException,Exception
        {
                con.modificar("update                IMP_PRESIMP                set
id_implementos='"+this.implemento.getId_implemento()+
                        "' where id_pr_stamo_imp='"+this.id_prestamo_imp+"'");

           con.modificar("update pr_stamo_implemento set fecha_registro=sysdate "+
                   "where id_pr_stamo_imp='"+this.id_prestamo_imp+"'");
        }
        /*****************************************************************/
        /**********************metodo para consultar prestamo**************/
        /*public Vector consultar_pr_imp() throws ClassNotFoundException,SQLException,Exception
        {

                Vector v = new Vector();

           ResultSet rs = con.consultar("select * from "+
               "imp_presimp i_p "+
               "inner join implementos i "+
               "on i_p.id_implementos=i.id_implementos "+
               "inner join pr_stamo_implemento p_i "+
               "on i_p.id_pr_stamo_imp=p_i.id_pr_stamo_imp "+
               "where i_p.id_pr_stamo_imp='"+this.id_prestamo_imp+"'");

    while(rs.next())
    {
        Prestamo_Implementos pi = new Prestamo_Implementos();

        pi.setId_prestamo_imp(rs.getString("id_pr_stamo_imp"));
        pi.setFecha_registro(rs.getString("fecha_registro"));
        pi.reserva_implemento.setId_reserva_imp(rs.getString("id_reserva_imp"));
        pi.setDocumento_ide(rs.getString("documento_ide"));
        v.add(pi);
    }

    return v;
        }
        /*****************************************************************/
        /**********************metodo para cancelar prestamo*************/
        public void cancelar_prest_imp() throws ClassNotFoundException,SQLException,Exception
        {
          conexion con = new conexion();
          PreparedStatement ps;
          Connection conec = con.conectar();
          String sql = new String();

          sql = "delete from imp_presimp where id_pr_stamo_imp='"+this.id_prestamo_imp+"'";
          ps = conec.prepareStatement(sql);
          ps.executeUpdate();

          sql = "delete from pr_stamo_implemento where id_pr_stamo_imp='"+this.id_prestamo_imp+"'";
          ps = conec.prepareStatement(sql);
```

```
            ps.executeUpdate();

            /*********************************************************************/
      ps.close();
      conec.close();
      con.desconectar();
            }
    /*************************************************************/
    /*********************metodo para consultar prestamo**********/
            public String consultar_prest_imp(String orden) throws
ClassNotFoundException,SQLException,Exception
            {
      return "select p_i.id_pr_stamo_imp,i.id_implementos,i.nombre,p_i.id_reserva_imp,p_i.fecha_registro,"+
"(select e.nombres "+
"from estudiante e inner join reserva_implementos ri on e.DOC_ID=ri.doc_id "+
"where ri.id_reserva_imp=p_i.id_reserva_imp) as nombres,"+
"(select e.apellidos "+
"from estudiante e inner join reserva_implementos ri on e.DOC_ID=ri.doc_id "+
"where ri.id_reserva_imp=p_i.id_reserva_imp) as apellidos "+
",(select r_i.doc_id from reserva_implementos r_i "+
"where r_i.id_reserva_imp=p_i.id_reserva_imp"+
") as doc_id, "+
"(select ri.hora_ini_imp from reserva_implementos ri "+
"inner join pr_stamo_implemento pi on ri.ID_RESERVA_IMP=pi.ID_RESERVA_IMP "+
"where pi.id_pr_stamo_imp=p_i.id_pr_stamo_imp) as hora_ini_imp,"+
"(select ri.hora_fin_imp from reserva_implementos ri "+
"inner join pr_stamo_implemento pi on ri.ID_RESERVA_IMP=pi.ID_RESERVA_IMP "+
"where pi.id_pr_stamo_imp=p_i.id_pr_stamo_imp) as hora_fin_imp,"+
"(select ri.fecha from reserva_implementos ri "+
"inner join pr_stamo_implemento pi on ri.ID_RESERVA_IMP=pi.ID_RESERVA_IMP "+
"where pi.id_pr_stamo_imp=p_i.id_pr_stamo_imp) as fecha "+
","+
"case "+
"when (select d_i.ID_PR_STAMO_IMP from DEVOLUCI_N_IMPLEMENTOS d_i where
d_i.ID_PR_STAMO_IMP=p_i.ID_PR_STAMO_IMP)>=0 "+
"then "+
"1 "+
"else "+
"0 "+
"end as devuelto "+
"from "+
"imp_presimp i_p "+
"inner join implementos i "+
"on i_p.id_implementos=i.id_implementos "+
"inner join pr_stamo_implemento p_i "+
"on i_p.id_pr_stamo_imp=p_i.id_pr_stamo_imp "+orden;
            }
            /*************************************************************/
   /************************metodo para registrar prestamo*****************/
            public void registrar_prest_imp() throws ClassNotFoundException,SQLException,Exception
            {

              conexion con = new conexion();
              PreparedStatement ps;
              Connection conec = con.conectar();
              String sql = new String();
              ResultSet rs = null;

              String ultimo = "select nvl(max(to_number(id_pr_stamo_imp))+1,0) as ultimo from
pr_stamo_implemento";
      PreparedStatement psPrim = conec.prepareStatement(ultimo) ;
      rs = psPrim.executeQuery();
      rs.next();
      ultimo =  rs.getString("ultimo");

      this.id_prestamo_imp = ultimo;

              sql = "insert into pr_stamo_implemento values(?,sysdate,?,?)";

      ps = conec.prepareStatement(sql);
      ps.setString(1,this.id_prestamo_imp);
      ps.setString(2,this.reserva_implemento);
      ps.setString(3,this.documento_ide);
              ps.executeUpdate();
```

```
            sql = "insert into imp_presimp values(?,?)";

            ps = conec.prepareStatement(sql);
     ps.setString(1,this.implemento);
     ps.setString(2,this.id_prestamo_imp);
            ps.executeUpdate();
    /*************************************************************************/
     rs.close();
     ps.close();
     psPrim.close();
     conec.close();
     con.desconectar();
          }
    /*************************************************************************/

    public static void main (String[] args) throws ClassNotFoundException,SQLException,Exception
    {
          Prestamo_Implementos p = new Prestamo_Implementos();
          p.setDocumento_ide("1");
          p.setReserva_implemento("0");
          //p.registrar_prest_imp();
          p.setId_prestamo_imp("3");
          p.cancelar_prest_imp();

    }

}

/**
* clase para administrar la reserva de espacios deportivos
*/

package reserva;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;

public class Reserva_Espacio implements Serializable
{

          private String id_reserva_esp;
          private String fecha;
          private String hora_ini;
          private String hora_fin;
          private String doc_id;
          private String espacio_fisicos;
    private String documento_ide;




    public void setId_reserva_esp(String id_reserva_esp)
          {
                this.id_reserva_esp = id_reserva_esp;
          }

          public String getId_reserva_esp()
          {
                return id_reserva_esp;
          }


      public void setFecha(String fecha)
          {
                this.fecha = fecha;
          }

          public String getFecha()
          {
                return fecha;
          }
```

```
    public void setHora_ini(String hora_ini)
        {
            this.hora_ini = hora_ini;
        }

        public String getHora_ini()
        {
            return hora_ini;
        }



    public void setHora_fin(String hora_fin)
        {
            this.hora_fin = hora_fin;
        }

        public String getHora_fin()
        {
            return hora_fin;
        }



   public void setDoc_id(String doc_id)
        {
            this.doc_id = doc_id;
        }

        public String getDoc_id()
        {
            return doc_id;
        }

   public void setDocumento_ide(String documento_ide)
        {
            this.documento_ide = documento_ide;
        }

        public String getDocumento_ide()
        {
            return documento_ide;
        }


   public void setEspacio_fisicos(String espacio_fisicos)
        {
            this.espacio_fisicos = espacio_fisicos;
        }

        public String getEspacio_fisicos()
        {
            return espacio_fisicos;
        }

/****************metodo para consultar datos de la reserva /***paginacion*******/

   public String consultar_res_esc(String orden)throws ClassNotFoundException,SQLException,Exception
   {


        return "select r_esp.ID_ESC,r_esp.ID_RESERVA_ESP,r_esc.fecha,"+
"r_esc.hora_ini,r_esc.hora_fin,r_esc.documento_ide,r_esc.doc_id,esc.nombre,"+
"case when nvl((select e.nombres from  reserva_espacio re "+
"inner join estudiante e on re.DOC_ID=e.DOC_ID "+
"where re.id_reserva_esp=r_esp.ID_RESERVA_ESP),'0')!='0' then (select e.nombres from  reserva_espacio
re "+
"inner join estudiante e on re.DOC_ID=e.DOC_ID "+
"where re.id_reserva_esp=r_esp.ID_RESERVA_ESP) "+
//"else 'No es Estudiante!' "+
"end as nombres,"+
"case when nvl((select e.apellidos from  reserva_espacio re "+
"inner join estudiante e on re.DOC_ID=e.DOC_ID "+
"where re.id_reserva_esp=r_esp.ID_RESERVA_ESP),'0')!='0' then (select e.apellidos from  reserva_espacio
re "+
```

```
"inner join estudiante e on re.DOC_ID=e.DOC_ID "+
"where re.id_reserva_esp=r_esp.ID_RESERVA_ESP) "+
//"else ' ' "+
"end as apellidos "+
"from RESERV_ESP r_esp "+
"inner join RESERVA_ESPACIO r_esc on r_esp.id_reserva_esp=r_esc.id_reserva_esp "+
"inner join ESCENARIO esc on r_esp.id_esc=esc.id_esc "+
"where nvl((select p.id_PR_STAMO_ESP from PR_STAMO_ESPACIO p where
p.id_PR_STAMO_ESP=r_esp.ID_RESERVA_ESP),'no')='no' "+
orden;


   }
/**********metodo para registrar reserva de espacio o escenario**********/
          public void registrar_reser_esp(String tipo)throws
ClassNotFoundException,SQLException,Exception
          {
            conexion con = new conexion();
            PreparedStatement ps;
            Connection conec = con.conectar();
            ResultSet rs = null;
            String sql = new String();
            /*****************buscando el id de la reserva*************************/
    String ultimo = "select nvl((max(to_number(id_reserva_esp))+1),0) as total from reserva_espacio";
    PreparedStatement psPrim = conec.prepareStatement(ultimo) ;
    rs = psPrim.executeQuery();
    rs.next();
    ultimo =  rs.getString("total");
    this.id_reserva_esp = ultimo;

    /*************************************************************************/
    String cad = (tipo.equals("1"))?("'"+this.documento_ide+"'"):("null");
    sql = "insert into RESERVA_ESPACIO values(?,to_date(?,'YYYY-MM-DD'),"+
                "?,?,"+cad+",?"+")";

    ps = conec.prepareStatement(sql);
    ps.setString(1,this.id_reserva_esp);
    ps.setString(2,this.fecha);
    ps.setString(3,this.hora_ini);
    ps.setString(4,this.hora_fin);
    ps.setString(5,this.doc_id);
    ps.executeUpdate();
   /*********************************************************************/

    sql = "insert into RESERV_ESP values (?,?)";

    ps = conec.prepareStatement(sql);
    ps.setString(1,this.espacio_fisicos);
    ps.setString(2,this.id_reserva_esp);
    ps.executeUpdate();
    /*********************************************************************/
    rs.close();
    ps.close();
    psPrim.close();
    conec.close();
    con.desconectar();

          }

 /**********metodo para cancelar reserva de espacio o escenario*****************************/
          public void cancelar_reser_esp() throws ClassNotFoundException,SQLException,Exception
          {
            conexion con = new conexion();
            PreparedStatement ps;
            Connection conec = con.conectar();
            String sql = new String();

            sql = "delete from RESERV_ESP where id_reserva_esp=?";

            ps = conec.prepareStatement(sql);
    ps.setString(1,this.id_reserva_esp);
    ps.executeUpdate();

    sql = "delete from RESERVA_ESPACIO where id_reserva_esp=? ";
```

```
            ps = conec.prepareStatement(sql);
        ps.setString(1,this.id_reserva_esp);
        ps.executeUpdate();
        /*********************************************************************/
        ps.close();
        conec.close();
        con.desconectar();

            }
 /************metodo para modificar reserva*************************/
            public void modificar_reser_esp() throws ClassNotFoundException,SQLException,Exception
            {

              conexion con = new conexion();
              PreparedStatement ps;
              Connection conec = con.conectar();
              String sql = new String();
              /*********************************************************************/
              sql = "update RESERV_ESP set id_esc=?"+
                        " where id_reserva_esp=?";
              /*********************************************************************/
              ps = conec.prepareStatement(sql);
        ps.setString(1,this.espacio_fisicos);
        ps.setString(2,this.id_reserva_esp);
        ps.executeUpdate();
              /*********************************************************************/
              sql          =          "update          RESERVA_ESPACIO          set          fecha=to_date(?,'YYYY-MM-
DD'),hora_ini=?,hora_fin=?,"+
                                  "doc_id=?"+
                                  " where id_reserva_esp=?";
             /*********************************************************************/
              ps = conec.prepareStatement(sql);
        ps.setString(1,this.fecha);
        ps.setString(2,this.hora_ini);
        ps.setString(3,this.hora_fin);
        ps.setString(4,this.doc_id);
        ps.setString(5,this.id_reserva_esp);
        ps.executeUpdate();
              /*********************************************************************/
              ps.close();
        conec.close();
        con.desconectar();
              }

 /************metodo para consultar una reserva**********/
            public Vector consultar_reser_esp() throws ClassNotFoundException,SQLException,Exception
            {
               ResultSet rs = null;
                     Vector reser = new Vector();
               conexion con = new conexion();
               PreparedStatement ps;
               Connection conec = con.conectar();
               String sql = new String();

                     sql = "select r_esc.id_reserva_esp,r_esc.doc_id,r_esp.id_esc,to_char(r_esc.FECHA,'YYYY-
MM-DD') as fecha,"+
                                  "r_esc.documento_ide,r_esc.hora_ini,r_esc.hora_fin"+
                                  " from RESERV_ESP r_esp "+
            "inner join RESERVA_ESPACIO r_esc on r_esp.id_reserva_esp=r_esc.id_reserva_esp "+
            "inner join ESCENARIO esc on r_esp.id_esc=esc.id_esc "+
            "where r_esc.id_reserva_esp=? ";

              ps = conec.prepareStatement(sql);
          ps.setString(1,this.id_reserva_esp);
          rs = ps.executeQuery();

 /*****************************************************/
                     while(rs.next())
                     {

                     Reserva_Espacio re = new Reserva_Espacio();
                     re.setId_reserva_esp(rs.getString("id_reserva_esp"));
                     re.setFecha(rs.getString("fecha"));
                     re.setHora_ini(rs.getString("hora_ini"));
                     re.setHora_fin(rs.getString("hora_fin"));
```

```
                re.setEspacio_fisicos(rs.getString("id_esc"));
                re.setDocumento_ide(rs.getString("documento_ide"));
                re.setDoc_id(rs.getString("doc_id"));
            reser.add(re);
                }
/*********************************************************/
     rs.close();
     ps.close();
     conec.close();
     con.desconectar();

    return reser;
        }
/*********************main*****************************/
public static void main(String arg[]) throws ClassNotFoundException,SQLException,Exception
{
        Reserva_Espacio re = new Reserva_Espacio();
        /*ResultSet rs = re.consultar_res_esc_1(" ");

        while(rs.next())
        {
        System.out.print(rs.getString("fecha"));
        }*/

        re.setFecha("2007/11/12");
        re.setHora_ini("10");
        re.setHora_fin("12");
        re.setEspacio_fisicos("5");
        re.setDoc_id("1");
        re.setId_reserva_esp("0");
        re.setDocumento_ide("4");
        re.registrar_reser_esp("1");


        //re.modificar_reser_esp();
 /*Vector v = re.consultar_reser_esp();

        for(int i=0; i<v.size();i++)
                {
                  System.out.print(((Reserva_Espacio)v.elementAt(i)).getFecha());
                }

        //re.cancelar_reser_esp();

        //re.modificar_reser_esp();*/

}
/*************************************************/

}

/**
* clase para administrar el reserva de implementos deportivos
*/

package reserva;

import BaseDatos.conexion;
import java.sql.*;
import java.io.*;
import java.util.*;

public class Reserva_Implementos implements Serializable
  {

        private String id_reserva_imp;
        private String fecha;
        private String hora_ini;
        private String hora_fin;
        private String doc_id;
        private String implemento;
        private String documento_ide;


   public void setId_reserva_imp(String id_reserva_imp)
```

```
        {
                this.id_reserva_imp = id_reserva_imp;
        }

        public String getId_reserva_imp()
        {
                return id_reserva_imp;
        }

        public void setFecha(String fecha)
        {
                this.fecha = fecha;
        }

        public String getFecha()
        {
                return fecha;
        }

        public void setHora_ini(String hora_ini)
        {
                this.hora_ini = hora_ini;
        }

        public String getHora_ini()
        {
                return hora_ini;
        }

    public void setHora_fin(String hora_fin)
        {
                this.hora_fin = hora_fin;
        }

        public String getHora_fin()
        {
                return hora_fin;
        }

    public void setDoc_id(String doc_id)
        {
                this.doc_id = doc_id;
        }

        public String getDoc_id()
        {
                return doc_id;
        }

        public void setDocumento_ide(String documento_ide)
        {
                this.documento_ide = documento_ide;
        }

        public String getDocumento_ide()
        {
                return documento_ide;
        }

        public void setImplemento(String implemento)
        {
                this.implemento = implemento;
        }

        public String getImplemento()
        {
                return implemento;
        }

    /**********metodo para registrar nueva reserva**********/
        public void registrar_reser_imp(String tipo) throws
ClassNotFoundException,SQLException,Exception
        {
          conexion con = new conexion();
          PreparedStatement ps;
```

```
            Connection conec = con.conectar();
            ResultSet rs = null;
            String sql = new String();
    /*****************buscando el id de la reserva*************************/
    String ultimo = "select nvl((max(to_number(id_reserva_imp))+1),0) as ultimo from
reserva_implementos";
    PreparedStatement psPrim = conec.prepareStatement(ultimo) ;
    rs = psPrim.executeQuery();
    rs.next();
    ultimo =  rs.getString("ultimo");
    this.id_reserva_imp = ultimo;

    /*************************************************************************/
    /*************************************************************************/
    String cad = (tipo.equals("1"))?("'"+this.documento_ide+"'"):("null");
    sql = "insert into reserva_implementos values(?,TO_DATE(?,'YYYY-MM-DD'),?,?,"+cad+",?)";

    ps = conec.prepareStatement(sql);
    ps.setString(1,this.id_reserva_imp);
    ps.setString(2,this.fecha);
    ps.setString(3,this.hora_ini);
    ps.setString(4,this.hora_fin);
    ps.setString(5,this.doc_id);
    ps.executeUpdate();
    /*************************************************************************/
    sql = "insert into imp_reserva values(?,?)";
    ps = conec.prepareStatement(sql);
    ps.setString(1,this.implemento);
    ps.setString(2,this.id_reserva_imp);
    ps.executeUpdate();
    /*************************************************************************/
    rs.close();
    ps.close();
    psPrim.close();
    conec.close();
    con.desconectar();

          }
  /*************************************************/
  /***********metodo para cancelar reserva*****************/
         public void cancelar_reser_imp() throws ClassNotFoundException,SQLException,Exception
         {
           conexion con = new conexion();
           PreparedStatement ps;
           Connection conec = con.conectar();
           ResultSet rs = null;
           String sql = new String();

           sql = "delete from imp_reserva where id_reserva_imp=? ";
           ps = conec.prepareStatement(sql);
           ps.setString(1,this.id_reserva_imp);
           ps.executeUpdate();

           sql = "delete from reserva_implementos where id_reserva_imp=?";
           ps = conec.prepareStatement(sql);
           ps.setString(1,this.id_reserva_imp);
           ps.executeUpdate();

    /*************************************************************************/
    ps.close();
    conec.close();
    con.desconectar();

         }
```

```
   /************************************************/
   /******************metodo para consultar reservas************/
   public String consultar_reser_imp(String orden) throws
ClassNotFoundException,SQLException,Exception
           {
      return "select
imp.id_implementos,ri.id_reserva_imp,ri.doc_id,imp.nombre,ri.fecha,ri.hora_ini_imp,ri.hora_fin_imp,"+
"(select e.nombres "+
"from estudiante e inner join reserva_implementos r_i on e.DOC_ID=r_i.doc_id "+
"where r_i.id_reserva_imp=ri.id_reserva_imp) as nombres,"+
"(select e.apellidos "+
"from estudiante e inner join reserva_implementos r_i on e.DOC_ID=r_i.doc_id "+
"where r_i.id_reserva_imp=ri.id_reserva_imp) as apellidos "+
"from imp_reserva ir "+
"inner join implementos imp on ir.id_implementos=imp.id_implementos "+
"inner join reserva_implementos ri on ir.id_reserva_imp=ri.id_reserva_imp "+
"where nvl((select p.id_pr_stamo_imp from pr_stamo_implemento p where
p.id_reserva_imp=ri.id_reserva_imp),'no')='no' "+
orden;
           }
   /*************************************************************/
   /****************metodo para consultar datos de la reserva*******/

   public Vector consultar_res()throws ClassNotFoundException,SQLException,Exception
   {
    ResultSet rs = null;
           Vector reser = new Vector();

           conexion con = new conexion();
           PreparedStatement ps;
           Connection conec = con.conectar();
           String sql = new String();
            sql = "select r_i.id_reserva_imp,to_char(r_i.fecha,'YYYY-MM-DD') as fecha, "+
                    "r_i.hora_ini_imp,r_i.hora_fin_imp,imp.id_implementos,r_i.documento_ide,r_i.doc_id "+
                    "from imp_reserva  im_r "+
        "inner join reserva_implementos r_i on im_r.id_reserva_imp=r_i.id_reserva_imp "+
        "inner join implementos imp on imp.id_implementos=im_r.id_implementos "+
        "where r_i.id_reserva_imp=? ";

    ps = conec.prepareStatement(sql);
           ps.setString(1,this.id_reserva_imp);
    rs = ps.executeQuery();
          /*****************************************************/
                   while(rs.next())
                   {
                   Reserva_Implementos ri = new Reserva_Implementos();
                   ri.setId_reserva_imp(rs.getString("id_reserva_imp"));
                   ri.setFecha(rs.getString("fecha"));
                   ri.setHora_ini(rs.getString("hora_ini_imp"));
                   ri.setHora_fin(rs.getString("hora_fin_imp"));
                   ri.setImplemento(rs.getString("id_implementos"));
                   ri.setDocumento_ide(rs.getString("documento_ide"));
                   ri.setDoc_id(rs.getString("doc_id"));
             reser.add(ri);
                   }
  /*********************************************************/
    rs.close();
    ps.close();
    conec.close();
    con.desconectar();

           return reser;
   }

   /*************************************************************/
   /************metodo para modificar reservas*********************/
           public void modificar_reser_imp(String tipo_usuario) throws
ClassNotFoundException,SQLException,Exception
           {
                   String c = new String();
                   conexion con = new conexion();
             PreparedStatement ps;
             Connection conec = con.conectar();
             String sql = new String();
```

```
                sql =      "update reserva_implementos set fecha=to_date(?,'YYYY-MM-DD'),"+
            "hora_ini_imp=?,hora_fin_imp=?"+
            ",doc_id=? "+
            "where id_reserva_imp=?";


      ps = conec.prepareStatement(sql);
      ps.setString(1,this.fecha);
      ps.setString(2,this.hora_ini);
      ps.setString(3,this.hora_fin);
      ps.setString(4,this.doc_id);
      ps.setString(5,this.id_reserva_imp);

      ps.executeUpdate();

      sql = "update imp_reserva set id_implementos=? "+
            "where id_reserva_imp=?";

      ps = conec.prepareStatement(sql);
      ps.setString(1,this.implemento);
      ps.setString(2,this.id_reserva_imp);

      ps.executeUpdate();

     /*******************************************************/
     ps.close();
     conec.close();
     con.desconectar();

          }
          /*********************************************************************/
 public static void main (String[] args) throws ClassNotFoundException,SQLException,Exception
 {
          Reserva_Implementos r = new Reserva_Implementos();
          r.setImplemento("5");
          r.setDoc_id("12");
          r.setFecha("2007/12/12");
          r.setHora_fin("15");
          r.setHora_ini("16");
          r.setId_reserva_imp("0");

          r.modificar_reser_imp("1");
          //r.registrar_reser_imp("1");
          /*
          Vector v = r.consultar_res();
          for(int i=0; i<v.size();i++)
                {
                  System.out.print(((Reserva_Implementos)v.elementAt(i)).getImplemento());
                }
          //r.modificar_reser_imp("1")*/

}
 }
```

6.4.4. Manual de usuario

El manual de usuario al igual que el manual técnico se encuentra en un documento adjunto a este.

6.4.5. Plan de instalación

No es necesario definir un plan de instalación, ya que el hardware y software para esta aplicación, como lo son equipos, servidor, Oracle, entre otros, ya están disponibles en la Universidad del Magdalena.

6.4.6. Resultados de pruebas

A continuación En la tabla 21, se muestran los resultados obtenidos en las primeras pruebas realizadas al módulo.

Función	Descripción	Resultado esperado	Resultado obtenido	Solución
Reservar	Registrar Reserva de Implemento a estudiante activo	Datos de la nueva reserva y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	La Reserva se hizo satisfactoriamente, justo como fue diseñado y creado el Sistema.	No requerida.
Reservar	Registrar reserva de Escenario a estudiante no activo	Mensaje de error al intentar registrar la operación.	El Sistema respondió de acuerdo a lo que se esperaba.	No requerida.
Prestar	Registrar el Préstamo de Implemento a estudiante activo	Datos del nuevo préstamo y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	El Préstamo se hizo satisfactoriamente, justo como fue diseñado y creado el Sistema.	No requerida.
Devolver	Registrar la Devolución del Implemento prestado anteriormente	Datos de la nueva devolución y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	La Devolución se hizo satisfactoriamente, justo como fue diseñado y creado el Sistema	No requerida.
Reservar	Registrar Reserva de Implemento por algún estudiante que anteriormente ya haya utilizado el servicio de Préstamo y Devolución.	Datos de la nueva reserva y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	El Sistema rechazó la nueva Reserva y mostró un mensaje errado indicando que el usuario ya poseía un Reserva, ya que en las condiciones para esta consulta no se tuvo en cuenta esta situación.	Modificar condiciones en las consultas que establecen los límites de Reserva, habilitando aquel Usuario que haya realizado un proceso de Préstamo y Devolución correctamente.
Devolver	Registrar la Devolución de un Implemento en el cual el estudiante se haya pasado del límite de tiempo.	El sistema muestra un mensaje informando la falta y la sanción que se le impondrá al estudiante, luego	Datos de la nueva Devolución y del estudiante en la base de datos, con su respectivo mensaje que	Establecer las consultas y sus respectivas condiciones a través de las fechas de

		registra en la base de datos el nombre, código y la sanción que se le estableció.	confirma la operación, pero no establece sanción, ya que no se han creado las condiciones para esta operación.	Préstamos y de las Devoluciones, para controlar la entrega de Implementos.

Tabla 21: Resultado de pruebas.

6.5. ENTREGABLE DE LA FASE DE TRANSICIÓN

6.5.1. Sistema de ambientes simulados de producción

La instalación de prueba para este módulo, dio como resultado la información que se encuentra en la tabla 22.

Fecha inicial de instalación	Fecha final de instalación.	Dificultades	Solución
10-02-2008	15-02-2008	Ninguna	No requerida

Tabla 22: Instalación de prueba

6.5.2. Evaluación del entrenamiento

El entrenamiento se dio por parte de los desarrolladores de este proyecto, y fue recibido por el asistente y el coordinador del área de deportes de la Universidad del Magdalena.

La siguiente tabla muestra las preguntas hechas al asistente y coordinador del área de deportes de la Universidad del Magdalena y sus respectivas respuestas como calificación al adiestramiento que recibieron acerca de este módulo.

La calificación se realizó de la siguiente manera:

- En las casillas que aparezca 1, es porque solo uno de los dos encuestados escogió esa opción.

- En las casillas que aparezca 2, es porque los dos encuestados escogieron esa opción.

Preguntas	Deficiente	Regular	Bien	Excelente	No responde
El nivel de dificultad del entrenamiento fue:		1	1		
La organización y cumplimiento del instructor fue:				2	
La capacidad de responder a las preguntas por parte del instructor fue:				2	
La claridad de las presentaciones del instructor fue:				2	

Tabla 23: Evaluación de entrenamiento.

6.5.3. Resultados de pruebas

En la tabla 24 se muestra el resultado de las pruebas hechas al módulo.

Función	Descripción	Resultado esperado	Resultado obtenido	Solución
Reservar	Registrar Reserva de Implemento a estudiante activo	Datos de la nueva reserva y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	La Reserva se hizo satisfactoriamente, justo como fue diseñado y creado el Sistema.	No requerida.
Reservar	Registrar reserva de Escenario a estudiante no activo	Mensaje de error al intentar registrar la operación.	El Sistema respondió de acuerdo a lo que se esperaba y la Reserva no se pudo realizar.	No requerida.
Prestar	Registrar el Préstamo de Implemento a estudiante activo	Datos del nuevo préstamo y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	El Préstamo se hizo satisfactoriamente, justo como fue diseñado y creado el sistema.	No requerida.
Devolver	Registrar la Devolución del Implemento prestado anteriormente	Datos de la nueva devolución y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	La Devolución se hizo satisfactoriamente, justo como fue diseñado y creado el Sistema	No requerida.
Reservar	Registrar Reserva de Implemento por algún estudiante que anteriormente ya haya utilizado el servicio de Préstamo y Devolución.	Datos de la nueva reserva y del estudiante en la base de datos, con su respectivo mensaje que confirma la operación.	La Reserva se hizo satisfactoriamente, justo como fue diseñado y creado el Sistema.	No requerida.
Devolver	Registrar la Devolución de un Implemento en el cual el estudiante se haya pasado del límite de tiempo.	El sistema muestra un mensaje informando la falta y la sanción que se le impondrá al estudiante, luego registra en la base de datos el nombre, código y la sanción que se le estableció.	Datos de la nueva Devolución, Estudiante y la Sanción en la base de datos, con su respectivo mensaje confirmando el suceso.	No requerida.

Tabla 24: Resultado de pruebas.

6.6. ENTREGABLE DE LA FASE DE PRODUCCIÓN

6.6.1. Evaluación del sistema en operación

A continuación en la tabla 25 se muestran los resultados obtenidos en el módulo después de ponerlo en operación.

Operación	Resultado esperado	Resultado obtenido	Solución
Registrar nueva Reserva de Implemento.	Datos de la Reserva de Implemento registrado en la base de datos sin ningún error.	Reserva de Implemento registrada con éxito.	No requerida.
Registrar Préstamo de Implemento.	Datos del Préstamo del Implemento registrado en la base de datos sin ningún error.	Préstamo de Implemento registrado con éxito.	No requerida.
Registrar Reserva de Escenarios.	Datos de la Reserva de Escenario registrado en la base de datos sin ningún error.	Reserva de Escenario registrada con éxito.	No requerida.
Registrar Préstamo de Escenarios.	Datos del Préstamo de Escenario registrado en la base de datos sin ningún error.	Préstamo de Escenario registrado con éxito.	No requerida.
Devolver Implemento.	Datos de la Devolución de Implemento registrado en la base de datos sin ningún error.	Devolución de Implemento registrada con éxito.	No requerida.

Tabla 25: Evaluación del sistema.

6.6.2. Evaluación del desempeño del sistema

En la tabla 26 se muestran los problemas de desempeño que presenta la aplicación.

Operación	Tiempo de respuesta esperado	Tiempo de respuesta observado	Solución
Registrar nueva Reserva de Implemento.	Máx. 5 segundos	Menos de 5 segundos.	No requerida.
Registrar Préstamo de Implemento.	Máx. 5 segundos	Menos de 5 segundos.	No requerida.
Registrar Reserva de Escenarios.	Máx. 5 segundos	Menos de 5 segundos.	No requerida.
Registrar Préstamo de Escenarios.	Máx. 5 segundos	Menos de 5 segundos.	No requerida.
Devolver Implemento.	Máx. 5 segundos	Menos de 5 segundos.	No requerida.

Tabla 26: Evaluación de desempeño.

6.6.3. Mejoras futuras

Una de las características importantes de los sistemas de información es que poseen una flexibilidad y tendencia al cambio que facilita su adaptación a nuevas necesidades que requiera el área donde se ha aplicado, este módulo es creado para prestar un servicio a los estudiantes de la Universidad del Magdalena, pero se puede modificar para que ese servicio sea prestado a toda la comunidad universitaria, como lo son, profesores, trabajadores, egresados entre otros.

7. LIMITACIONES

En el desarrollo de este módulo se presentaron algunos inconvenientes o limitaciones.

Inicialmente se presentó la falta de equipos para trabajar, como computadores aptos para desarrollar software y además tener acceso al servicio de Internet, ya que aunque el Centro de Investigación y Desarrollo del Software había facilitado dos computadores para este proyecto, por motivos laborales se hacia imposible trabajar en el día, y conseguir un permiso para trabajar todos los días en la noche en el CIDS era complicado.

Falta de conocimiento en el manejo de bases de datos, ya que por algunos inconvenientes sólo se había tomado un curso teórico sobre el tema, lo cual originó falta de práctica en el manejo de ORACLE.

8. CRONOGRAMA DE ACTIVIDADES

Actividades	mes1				mes2				mes3				mes4			
Fase de definición																
Descripción del dominio del problema.																
Modelo de procesos.																
Servicios y usuarios.																
Riesgos, impactos y factores críticos del éxito.																
Diagrama jerárquico de funciones.																
Interfaces.																
Arquitectura técnica inicial.																
Interacción con sistemas existentes.																
Glosario de términos.																
Fase de análisis																
Modelo de datos																
Modelo de funciones																
Diagrama de transición de estados																
Matriz CRUD																
Glosario de términos																
Fase de diseño																
Modelo lógico de la base de datos																
Plan de capacidad																
Diseño de la aplicación																
Esquema de autorización																
Modelo de pruebas del sistema																
Manual del usuario inicial																
Fase de Construcción																
Diseño físico de los datos																
Código de la aplicación																
Manual de usuario completo																
Plan de instalación																
Pruebas																
Fase de Transición																
Sistema de ambientes simulados de producción																
Evaluación del entrenamiento																
Resultados de pruebas																
Fase de Producción																
Evaluación del sistema en operación																
Evaluación del desempeño del sistema																
Mejoras futuras																

Tabla 27: Cronograma de actividades.

9. PRESUPUESTO

CONCEPTO	COSTO UNITARIO ($)	CANTIDAD REQUERIDA	COSTO TOTAL ($)
Recursos Humanos			
Director	85.000 ($/hora)	24 horas	2.040.000
Estudiantes encargados del proyecto (Horas)	12.500 ($/hora)	240 horas	3.000.000
Subtotal			**5.040.000**
Suministros			
Papelería (resma de papel)	11.450	2	22.900
Cd-Roms (unidad)	1.200	10	12.000
Memoria USB(512Mb)	50.000	1	50.000
Cartuchos de tinta	70.000	2	140.000
Subtotal			**224.900**
Equipos			
Internet (2 horas/día)	2.000 ($/hora)	100 horas	200.000
Computador (5 horas/día)	2.000 ($/hora)	300 horas	600.000
Subtotal			**800.000**
Servicios			
Útiles varios (lápices, otros)	2.000	15	30.000
Fotocopias	50	300	15.000
subtotal			**45.000**
Total:			**6.109.900**

Tabla 28: Presupuesto.

10. CONCLUSIONES

Es importante y necesario para el desarrollo de una empresa o en este caso el de un área de trabajo que ofrece un servicio a una comunidad que se disponga de herramientas para el manejo de sus procesos.

Esta aplicación es un módulo del sistema de información de bienestar universitario (SIADUM) de la Universidad del Magdalena y se creo como una herramienta para el manejo y soporte sistematizado de las reservas y préstamos de los implementos y escenarios deportivos.

Teniendo en cuenta las necesidades planteadas de esta área y el requerimiento de un software de alta calidad se utilizo para el diseño y desarrollo de este módulo el método CDM (Custom Development Method) de Oracle Corporation, al que se le han incorporado algunos modelos más recientes con la notación estándar de UML (Unified Modeling Language), ya que esta herramienta servirá para brindar un mejor servicio y al mismo tiempo contribuya en lograr los objetivos del área de deportes de la Universidad del Magdalena.

BIBLIOGRAFÍA

[STE_1] STEVENS, Perdita y POOLEY, Rob. Utilización de UML en ingeniería del software con objetos y componentes. Addison Wesley, 2003.

[JAM_2] JAMES, Martin. Organización de las Bases de Datos. Englewood Cliffs (New Jersey). Prentice Hall Internacional, 1977.

[PRE_3] PRESSMAN, Roger. Ingeniería del Software, Un Enfoque Práctico. Cuarta Edición. Madrid (España). Mc Graw-Hill, 1998.

[SEN_4] SENN, James A. Análisis y Diseño de Sistemas de Información. Segunda Edición. México. Mc Graw-Hill, 1992.

[FIG_5] FIGUEROA, Pablo. Elementos Notacionales de UML [en línea]: Versión 1.1. Disponible en Internet: http://www.cs.balberta.ca/~pfiguero/soo/uml.

[MON_6] Monografías_Definición arquitectura cliente servidor www.monografias.com

[MON_7] Monografías_Sistemas de Información www.monografias.com

[SAL_8] SALINAS, Patricio. Modelamientos de Clases en UML [en línea]: Disponible en Internet: http://www.dcc.uchile.cl/~psalinas/uml/introdution.html

[UNI_9] Universidad Javeriana. Biblioteca General PUJ [En línea]: Disponible en Internet: http://www.javeriana.edu.co/biblos/colec_audiovisuales.htm.

[UNI_10] Universidad Javeriana. Reserva Material Audiovisual [En línea]: Disponible en Internet: http://www.javeriana.edu.co/biblos/servicios/formato_reserva.htm

[ORA_11] Benbrook, Finta y Otros. Oracle Designer 2000: System Modeling and Tools. Volumenes I y II: 1995.