Universidad de las Ciencias Informáticas
Facultad 1

Título: Análisis y diseño del sistema para la gestión de la información de los almacenes de las unidades presupuestadas que se subordinan a la Oficina de Atención a Misiones de la Salud.

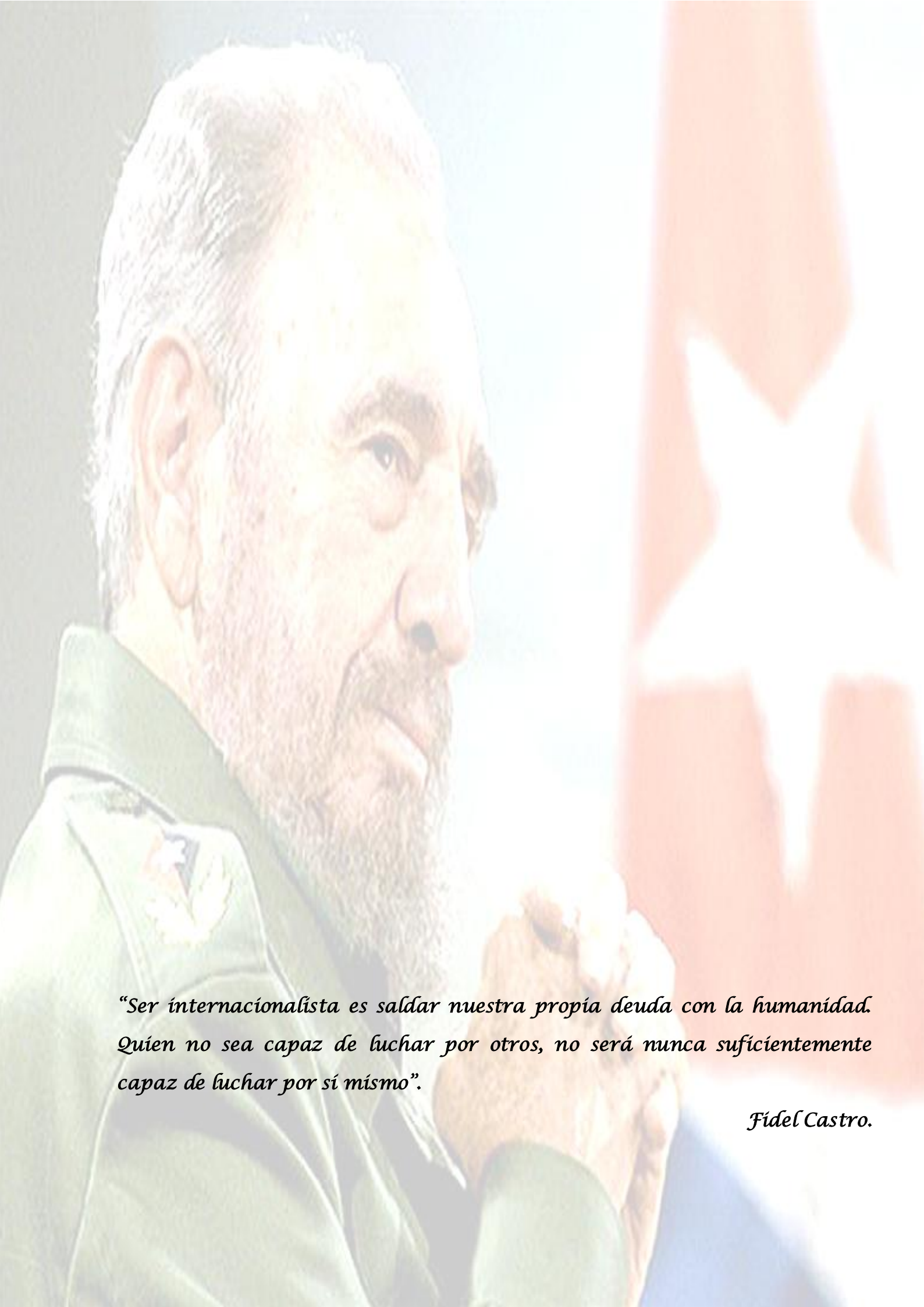
Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autoras:
Yanely Benitez Pérez.
Maidelys Nilva Céspedes Vargas.

Tutores:
Ing. Yariel Ramos Negrín.
Ing. Sissy Ramos López.

Ciudad de La Habana, junio del 2010.

"Año 52 de la Revolución".



"Ser internacionalista es saldar nuestra propia deuda con la humanidad. Quien no sea capaz de luchar por otros, no será nunca suficientemente capaz de luchar por sí mismo".

Fidel Castro.

Declaración de autoría

Declaramos ser autoras del trabajo de diploma Análisis y diseño del sistema para la gestión de la información de los almacenes de las unidades presupuestadas que se subordinan a la Oficina de Atención a Misiones de la Salud, y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del autor.

Yanely Benitez Pérez.

_____

Firma del autor.

Maidelys Nilva Céspedes Vargas.

_____

Firma del tutor.

Ing. Yariel Ramos Negrín.

_____

Firma del tutor.

Ing. Sissy Ramos López.

A nuestros tutores, por guiarnos y orientarnos durante el desarrollo del trabajo de diploma.

A Alexander, jefe del tribunal, por sus sabios consejos que nos sirvieron de gran ayuda.

A todos los compañeros que de una forma u otra nos ayudaron muchísimo, en especial a: Israel, Yuniel, Irán y Yoan Carlos.

De Yanely:

A mis papitos del alma, por estar siempre allí con ese amor tan grande que enternece cualquier corazón.

A mi mejor amigo entre todos los amigos, Timón, por su gran ayuda, su gran paciencia y sobre todas las cosas su gran amor. Negro te quiero muchísimo, y te prometo que nunca te voy a olvidar aun cuando la distancia se impone.

A Neicy y Manolo, dos amigos de la familia que me ayudaron muchísimo durante el transcurso de mi carrera.

A un grupo de compañeros que conocí en el transcurso de estos cinco años y que se han ganado mi respeto y admiración: Maidelys, Eliska, Yaité, Yeriuska, Suley, Gregseen, Yuniel, Irán.

A mi Cosi del alma por todo su apoyo, gracias mi amor.

De Maidelys:

A toda mi familia por el apoyo brindado, en especial a Mimi y a Pipo, mis padres queridos, por apoyarme incondicionalmente en todo momento, por estar presentes siempre que los necesité, por darme cariño, amor, los quiero mucho.

Al amor de mi vida, Luis Carlos, por llenarme de felicidad, por amarme, respetarme, apoyarme a pesar de la distancia, por su paciencia, te amo con la vida mi amor.

A mi abuelita linda Ángela, por estar siempre pendiente de mí, ayudándome en todo lo que podía.

Agradecimientos

A mi hermano Maikel y a mi cuñada Norelys, por apoyarme y ayudarme en todo momento.

A mis tios Addis, Elia Noris, Leo, Juan Carlos, Neyda, Irsel, Zory, Beto, Nolys, Edel, por su preocupación y ayuda.

A mis primos Gretel, Michel, Cely, Legna, Nelson, por ayudarme en todo lo que necesité.

A la familia de mi novio Luis Carlos, por haberme acogido como un miembro más y apoyarme siempre.

A mis amigos Yanelys, Eliska, Yaité, Suley, Gregseen, Yuniel, por apoyarme y motivarme en todo momento.

De Yanely:

En primer lugar, a las dos personas más importantes de mi vida:

A mi papá, por ese amor tan grande que pasa por encima de cualquier cosa, que hace que los ojos le brillen cuando habla de mí, que le hacen verme como una niña cuando hace ya algún tiempo comencé a transformarme en mujer, por el gran esfuerzo que ha realizado durante los cinco años de mi carrera para complacerme en todo, por la gran confianza que siempre me ha tenido, aun cuando las cosas se ponían feas, por creer siempre en mí y darme las fuerzas necesarias para soportar el dolor de solo verlo dos veces al año, no te imaginas cuanto te amo papito.

A mi mamá, por ese amor incondicional que solo una madre puede dar, por esos besitos tan ricos que me da antes de dormir y que tanto he extrañado durante estos cinco años, por su apoyo, su paciencia con mis malcriadeces, por su forma de mimarme cuando lo necesito, por descubrir con tan solo una mirada si mi corazón está triste, por esas noches de fiebre que pasaba en vela a mi lado acariciando mi pelo, por sacrificarse tanto para que no me falte nada y por esa mirada de amor que me calienta hasta el último de mis huesos, te amo mucho mamita.

A mi tía Titi, por su amor y su apoyo no solo durante la carrera, sino en toda mi vida.

A mis abuelitos Mima, Machucha y Ricardo por darme esos momentos de alegría característicos del amor de abuelo.

A mis tíos, por estar siempre pendientes de todos mis problemas y por su amor tan grande.

A mi novio Alexis, por todo el amor que me ha dado, por su comprensión y su paciencia, por su apoyo, por ese hombro que tantas veces me sirvió de paño de lágrimas, a ti también te amo Cosi.

De Maidelys:

A toda mi familia por el cariño y apoyo que me brindaron, en especial a mis adorables padres Mimi y Pipo, mi razón de ser, por su amor incondicional, por el sacrificio a que se sometieron todos estos años para complacerme en todo y para que no me faltara nada, por su grandioso ejemplo, por el apoyo moral, la confianza depositada en mí, para que mi sueño se hiciera realidad.

A mi querido amor Luis Carlos, por ser mi fuente de inspiración, amor, cariño y confianza.

A mi querida abuelita Ángela, por apoyarme y quererme en todo momento.

A mis preciosas sobrinitas Beatriz Lianet y Brianna Linet, por llenar mi vida de felicidad.

Resumen

La Oficina de Atención a Misiones de la Salud (OAMS), se crea con el objetivo de gestionar los recursos necesarios para las unidades presupuestadas (UP), donde cursan estudios de medicina jóvenes provenientes de América Latina y el Caribe, como parte del Programa de Formación de Médicos y el ALBA. A medida que el programa ha ido avanzando, se ha hecho necesario llevar el control de estos recursos dentro de los almacenes, para evitar la desviación y el mal empleo de los mismos. Actualmente, la OAMS no cuenta con un sistema que le permita llevar este control. Los procesos se realizan de forma manual, provocando el atraso en la entrega de la información, que imposibilita la toma de decisiones oportunas al no conocer el estado actual de la misma, además, se incrementan las posibilidades del error humano que han provocado pérdidas de datos importantes. Por ello, el presente trabajo de diploma tiene como objetivo general, desarrollar el análisis y diseño de una aplicación que permita gestionar la información derivada de los almacenes de las unidades presupuestadas subordinadas a la Oficina de Atención a Misiones de la Salud. Con el análisis y diseño del sistema para la gestión de la información de los almacenes de las unidades presupuestadas que se subordinan a la OAMS, se podrá desarrollar una aplicación que le permita al subdirector de aseguramiento de la misma, contar con la información necesaria, en el momento que lo necesite, podrá realizar cálculos precisos y confiables.

Índice

Introducción 1

Capítulo 1: Fundamentación teórica 4

1.1 Introducción 4

1.2 Fundamentación del tema 4

1.2.1 Diseño del sistema para la gestión de la información de los almacenes de las unidades presupuestadas subordinadas a la Oficina de Atención a Misiones de la Salud 4

1.2.2 Conceptos relacionados con el dominio del problema 4

1.3 Sistemas similares 5

1.3.1 Sistemas similares en el mundo 5

1.4 Estudio de las metodologías de desarrollo 7

1.4.1 Selección de la metodología de desarrollo de *software* 10

1.4.2 Lenguaje Unificado de Modelado (UML) 10

1.4.3 Herramienta de modelado 12

1.5 Plataforma de desarrollo 13

1.5.1 Lenguaje de desarrollo 14

1.5.2 Herramienta de desarrollo 15

1.6 Sistema gestor de base de datos 16

1.6.1 Lenguaje de base de datos 17

1.6.2 Herramienta para el modelado de la base de datos 19

1.7 Conclusiones 19

Capítulo 2 Características del sistema 20

2.1 Introducción 20

2.2 Descripción de los procesos del negocio 20

2.3 Modelación del negocio 21

2.3.1 Determinación y justificación de los actores del negocio 21

2.3.2 Determinación y justificación de los trabajadores del negocio 21
2.3.3 Diagrama de casos de uso del negocio 21
2.3.4 Diagrama de objetos 22
2.3.5 Descripción de los casos de uso del negocio 23
2.3.6 Diagramas de actividades de los casos de uso del negocio 26
2.4 Descripción del sistema 28
2.4.1 Especificación de los requerimientos 29
2.4.1.1 Requerimientos funcionales 29
2.4.1.2 Requerimientos no funcionales 31
2.4.2 Modelo de casos de uso del sistema 32
2.4.2.1 Descripción de los actores del sistema 32
2.4.2.2 Diagrama de casos de uso del sistema 33
2.4.2.3 Descripción de los casos de uso del sistema 34
2.5 Conclusiones 37
Capítulo 3 Análisis y diseño del sistema 38
3.1 Introducción 38
3.2 Modelo del análisis 38
3.2.1 Diagrama de clases del análisis 38
3.3 Diagrama de clases 39
3.3.1 Diagrama de interacción del análisis (colaboración) 41
3.4. Patrones de diseño a utilizar 43
3.5 Arquitectura del sistema 46
3.6 Modelo de diseño 47
3.6.1 Diagrama de clases del diseño 48
3.6.1.1 Descripción de las clases del diseño 49
3.7 Diagrama de despliegue 60

3.8 Diseño de la base de datos 61

3.9 Conclusiones.....61

Conclusiones generales.....62

Recomendaciones.....63

Referencias bibliográficas.....64

Bibliografía consultada.....65

Glosario de términos.....68

Introducción

Cuando en diciembre de 2004, en La Habana, los comandantes Hugo Chávez por Venezuela y Fidel Castro por Cuba, firmaban la Alternativa Bolivariana para las Américas, pocos visualizaban la perspectiva y potencialidad de dicho acuerdo; más bien lo ubicaban dentro del marco general de una simple declaración bilateral de reafirmación de hermandad cubano-venezolana.

El ALBA se fundamenta en la creación de mecanismos para crear ventajas cooperativas entre las naciones, que permitan compensar las asimetrías existentes entre los países del hemisferio. Se basa en la cooperación de fondos compensatorios para corregir las disparidades que colocan en desventaja a los países débiles frente a las primeras potencias.

Como parte de los convenios del ALBA, actualmente se forman en Cuba futuros médicos de disímiles países del mundo, con el objetivo de llevar a todos los rincones del planeta la asistencia médica y la atención primaria.

Para esto se crea el Programa de Formación de Médicos, que incluye la apertura de sedes con estos fines docentes en diferentes provincias del país, idea del Comandante en Jefe, basada en los principios internacionalistas, solidarios y revolucionarios promulgados por la Revolución.

Hoy, con diferentes centros dedicados a esta noble misión, se da oportunidad a que miles de jóvenes de otros países, realicen su preparación en el sistema docente–asistencial cubano, considerado uno de los mejores del mundo.

Para llevar a cabo esta tarea, se crea la Oficina de Atención a Misiones de la Salud, la cual es la responsable de asegurar los recursos humanos, financieros y materiales de este sistema, subordinándose a la misma los centros y unidades presupuestadas.

En la actualidad, existen procesos que a pesar de estar relacionados no se vinculan en uno solo: asignación de recursos materiales, sistemas de inventarios, coberturas, consumos, planes y control de los mismos. Resulta engorroso por los métodos actuales (hojas Excel), comprobar el estado y la veracidad de la información, lo que imposibilita la toma de decisiones oportunas y una correcta gestión de la información, por lo que se define como **problema científico:** ¿cómo mejorar la gestión de la información de la Oficina de Atención a Misiones de la Salud?

El **objeto de estudio** del presente trabajo se enfocó en la gestión de la información de almacenes.

A partir del objeto de estudio, se define como **campo de acción** la gestión de la información de los almacenes de las unidades presupuestadas que se subordinan a la OAMS.

Para responder el problema anteriormente planteado, se trazó como **objetivo general:** desarrollar el análisis y diseño de una aplicación que permita gestionar la información derivada de los almacenes de las unidades presupuestadas subordinadas a la Oficina de Atención a Misiones de la Salud.

Para cumplir el objetivo planteado, se propusieron las siguientes **tareas*:***

1. Realización de entrevistas a los clientes.
2. Estudio de sistemas similares en el mundo.
3. Estudio de metodología, herramientas y lenguajes a utilizar.
4. Estudio del funcionamiento del negocio.
5. Realización del análisis del sistema, confeccionando modelos que sirven de base para el diseño de la aplicación.
6. Realización del diseño del sistema, confeccionando modelos que sirven de base para el desarrollo de la aplicación.

El desarrollo del trabajo se sustentó en la siguiente **idea a defender:**

El análisis y diseño del sistema para la gestión de la información de los almacenes de las unidades presupuestadas subordinadas a la OAMS, facilitará la implementación del mismo, agilizando su proceso de desarrollo.

Los **posibles resultados** de este trabajo son**:**

El diseño de una aplicación que gestione y consolide todos los procesos afines del negocio actual, de las unidades presupuestadas subordinadas a la OAMS, aplicable además en otros entornos dentro del Ministerio de Salud Pública (MINSAP), así como los artefactos generados en los flujos de trabajo a desarrollar.

Métodos de investigación científica empleados:

Entre los **métodos teóricos** que se utilizaron se encuentran:

- Histórico-lógico:

Este método se utilizó para dar cumplimiento a la tarea de investigar sistemas similares en el mundo, permitiendo conocer la evolución que han tenido los mismos, y de esta forma, analizar si algunos de ellos aportaban una solución a la investigación.

- Analítico-sintético:

Estudiando y analizando todo lo relacionado al tema específico a abordar, se pudo hacer una profundización de todas sus características, dígase objetivos, logros, o deficiencias que se derivan, lo que permitió proponer mejoras para obtener el resultado esperado.

- Modelación:

Durante el desarrollo del trabajo, se realizaron diferentes modelos que ayudaron a comprender de manera simplificada la realidad, facilitando así, el estudio de nuevas relaciones y cualidades del objeto de estudio.

Método empírico utilizado:

- Entrevista:

Se realizaron entrevistas al cliente Osnyell Oviedo del Sol, con el objetivo de obtener la mayor información posible sobre las necesidades de la OAMS.

Para lograr la comprensión y claridad de los contenidos de la investigación, se estructuró el documento en tres capítulos:

Capítulo 1- Fundamentación teórica: en este capítulo se realizó un estudio de la situación actual del negocio, tendencias en Cuba y en el mundo, así como de la utilización de sistemas similares, además de una caracterización de las herramientas, metodología y tecnologías utilizadas.

Capítulo 2- Características del sistema: en este capítulo se describieron detalladamente todos los procesos del negocio, se definieron los requerimientos con que debe cumplir el *software* (funcionales y no funcionales), y se generaron los artefactos correspondientes a los flujos de trabajo modelamiento del negocio y requerimientos.

Capítulo 3- Análisis y diseño del sistema: se generaron los artefactos correspondientes a los flujos de trabajo análisis y diseño, se argumentaron algunas de las terminologías empleadas. Además, se abordaron los patrones utilizados y se brindó una explicación de la arquitectura propuesta.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se realiza un estudio de los sistemas similares en Cuba y en el mundo, haciendo un análisis de cada uno de ellos, lo que permite deducir cuánto aportan a la solución de la problemática planteada. Para lograr un mayor entendimiento del dominio del problema, se especifican una serie de conceptos relacionados con el mismo. Se realiza un estudio de herramientas, lenguajes y metodologías, además se realiza una fundamentación del tema propuesto.

1.2 Fundamentación del tema

1.2.1 Diseño del sistema para la gestión de la información de los almacenes de las unidades presupuestadas subordinadas a la Oficina de Atención a Misiones de la Salud

En la actualidad, la cooperación entre Cuba y los países de Latinoamérica y el mundo, se ha incrementado, lo que ha traído consigo el aumento de las misiones dentro y fuera del país. Entre ellas, el Programa de Formación de Médicos, al que cada vez se incorporan un mayor número de estudiantes en todas las ramas de la medicina. Este incremento, trae aparejado un aumento en la disponibilidad de recursos para cubrir las necesidades de los centros de estudio. La OAMS, es la encargada de realizar la gestión de estos recursos.

Con este crecimiento, se hace necesario llevar un control sobre los medios que se han puesto a disposición del programa, y de esta forma, poder tomar decisiones oportunas en el momento que se haga necesario, además, es de vital importancia evitar el desvío de recursos. Actualmente, este control se realiza de forma manual, lo que trae como consecuencia la pérdida de información importante o el retraso en la entrega de la misma.

Con este trabajo se pretende diseñar un sistema que permita automatizar los procesos que ocurren en los almacenes de las unidades presupuestadas subordinadas a la OAMS, que es donde se reciben los recursos que se destinan a los mismos.

1.2.2 Conceptos relacionados con el dominio del problema

- OAMS: Oficina de Atención a Misiones de la Salud, es la encargada de gestionar todos los recursos que se emplean en las misiones de la salud dentro y fuera del país.
- Gestión de la información: proceso mediante el cual, se obtiene la información procedente de los procesos que se realizan en los almacenes de la OAMS como: la asignación de recursos materiales, los diferentes sistemas de inventarios, la cobertura y consumo de alimentos, permitiendo realizar planes reales.

- Inventarios: relación detallada de los bienes que se encuentran en los almacenes subordinados a la OAMS.
- Unidad presupuestada: entidad física donde obtienen su preparación los estudiantes latinoamericanos de medicina, y donde los médicos internacionalistas realizan cursos de adiestramiento.
- Almacén: espacio físico donde se guardan los recursos de las unidades presupuestadas.
- Consumo: cantidad de productos que se utilizan por día.
- Cobertura: alcance en días de los productos según el consumo.
- Plan: cantidad de un producto que se asigna a una unidad presupuestada.
- Balance: cantidad de recursos financieros y materiales consumidos por una unidad presupuestada en un período de tiempo.

1.3 Sistemas similares

En la actualidad, existen disímiles sistemas que facilitan el control sobre los procesos que se llevan a cabo en los almacenes. Estos varían en dependencia del tipo de almacén o del tamaño de la empresa. Hoy en Cuba no existe un sistema encargado de gestionar estos procesos.

1.3.1 Sistemas similares en el mundo

***Easy WMS*:** es un *software* de gestión de almacenes dirigido a empresas de cualquier envergadura, dedicadas a los más diversos sectores de la actividad logística, desde una pequeña empresa con un almacén común, hasta el control logístico de un aeropuerto.

Tiene como principal objetivo controlar, coordinar y gestionar todos los procesos que se desarrollan dentro de un almacén. Este sistema permite adaptar sus funciones a las necesidades concretas de cada institución, teniendo en cuenta el grado de complejidad de las operaciones que se realizan dentro del almacén y la automatización del mismo, además no es un sistema aplicable solamente a empresas con experiencia, también puede adaptarse fácilmente al crecimiento de un negocio.

El *Easy WMS* brinda una gran seguridad, definiendo cuáles usuarios o grupos de usuarios hacen uso de la aplicación y restringiendo además la operatividad para cada usuario, de esta forma, garantiza que solo las personas autorizadas podrán acceder a las funcionalidades críticas de las que depende el funcionamiento del sistema.

Permite una visualización gráfica del almacén, de este modo, es posible conocer en tiempo real la capacidad del mismo y las áreas que quedan sin ocupar, disminuyendo así los errores en los envíos. Cuenta con un control de posición de los recursos, facilitando agrupar la mercancía por tipo y así, la búsqueda de los productos dentro del almacén. Brinda la posibilidad de realizar inventarios

permanentes con sus valores reales, por tanto, siempre se conoce la cantidad de productos dentro del almacén. Lleva un control de la fecha de vencimiento de cada producto, por lo que los costos de pérdida por caducidad son mínimos.

Sistema de Control Logístico (SCL): es un sistema de gestión tanto para centros de distribución como de almacenamiento, realizado sobre la base del más moderno sistema informático de gestión logística. Está basado en comunicación por radios terminales, que permite la interacción entre los proveedores y los clientes en el momento que está siendo trasladada la mercancía. Cuenta además con lectores de código de barras como medio de reconocimiento, haciendo más precisa la identificación de los productos, y con comunicación por internet, permitiendo el pago de facturas.

Mediante un sistema de localización, permite manejar información operativa como la ubicación de los transportes pertenecientes al proveedor, durante el tiempo en el que se traslada la mercancía. Brinda información en tiempo real, lo que permite llevar el control de los productos dentro del almacén evitando la pérdida de los mismos. Los procesos se realizan en el menor número posible de operaciones, lo que facilita el uso del sistema y aumenta su productividad en cuanto a tiempo empleado en operaciones.

Alvisoft: es una empresa dedicada al desarrollo de soluciones informáticas para el sector empresarial e industrial. El *software* para la gestión de los almacenes es precisamente uno de los productos que ofrece. El mismo fue diseñado para controlar todos los procesos involucrados en los diferentes tipos de almacenes, desde los documentos de ingreso y salida de productos, hasta la realización de inventarios y control de los mismos, brindando herramientas efectivas para el análisis logístico y de almacenaje.

Controla procesos como: movimientos de almacén (notas de ingreso y vales de salida), chequeo (consultas, búsquedas por agrupaciones, categorías, fechas, entre otros), costo promedio y recálculos de operaciones (balances y planes), tipos de movimiento de almacén (entrada o salida por factura, por sobrante o faltante), despacho de almacén, guías de remisión, reportes de reparto y consolidado, creación y control de inventarios críticos.

Este sistema tiene todas sus operaciones de almacén en línea y en tiempo real, permitiendo ahorrar tiempos de respuesta en la toma de decisión de negocio, pues no es preciso el contacto con la mercancía dentro del almacén, lo cual significa un ahorro considerable de tiempo y recursos. Controla cada proceso y aspecto de los almacenes tanto físicos como virtuales, por lo que permite un seguimiento riguroso tanto de los movimientos de almacén como de los inventarios, llevando un control absoluto sobre cada existencia en los almacenes. Es un sistema seguro, basado en permisos sobre actividades por cada usuario, lo cual le permitirá saber quién hizo “qué”, “cuándo” y “cómo” en el sistema, impidiendo así que se realicen operaciones no deseadas.

El estudio de todos estos sistemas ha sido de gran ayuda para dar solución al problema, porque todos de una forma u otra tratan el manejo de los procesos que se llevan a cabo actualmente en un almacén como por ejemplo: los movimientos de almacén, que cuentan con una nota de ingreso y los vales de salida, el control de los costos, la trazabilidad de los productos así como la realización de inventarios y planes. Sin embargo, no brindan una solución completa, debido a que son aplicaciones *web* en la que los procesos se realizan en línea y en tiempo real, y la OAMS no cuenta con los suficientes recursos para lograr la conectividad con las distintas unidades presupuestadas que se le subordinan.

1.4 Estudio de las metodologías de desarrollo

En un proyecto de desarrollo de *software*, la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo.

Proceso Unificado de Desarrollo (RUP): es el resultado de varios años de desarrollo y uso práctico, en el que se han unificado técnicas de desarrollo, a través del Lenguaje Unificado de Modelado (UML), y trabajo de muchas metodologías utilizadas por los clientes.

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

"RUP es un proceso de ingeniería de *software"*. (1)

Los aspectos que caracterizan el proceso unificado se resumen en tres frases: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental. Esto es lo que hace único al proceso unificado.

1. Dirigido por casos de uso: los casos de uso reflejan lo que los usuarios futuros necesitan y desean. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

2. Centrado en la arquitectura: describe los elementos del modelo que son más importantes para su construcción. Relaciona la toma de decisiones que indica cómo tiene que ser construido el sistema y en qué orden.

3. Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es

una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

"RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada uno de ellos, estos a la vez se dividen en fases y donde se debe tomar una decisión importante". (2)

- Fases:

• Conceptualización: se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

• Elaboración: se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

• Construcción: se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario.

• Transición: una vez concluidas las pruebas, el producto está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

- Flujos de ingeniería:

- Modelamiento del negocio: describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: define qué es lo que el sistema debe hacer.
- Análisis y diseño: describe cómo el sistema será realizado a partir de las funcionalidades previstas y las restricciones impuestas (requerimientos).
- Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba: busca los defectos a lo largo del ciclo de vida.
- Instalación: se produce la liberación del producto y se realizan actividades como empaque, instalación, asistencia a usuarios, entre otros, para entregar el *software* a los usuarios finales.

- Flujos de apoyo:

- Administración del proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Administración de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, entre otros.
- Ambiente: contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

- Principales elementos:

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

Programación Extrema (XP): la programación extrema, es una metodología de desarrollo ligera (o ágil), basada en una serie de valores y de prácticas de buenas maneras, que persigue el objetivo de aumentar la productividad y ganar tiempo a la hora de desarrollar programas.

Este modelo de programación se basa en una serie de metodologías de desarrollo de *software*, en la que se da prioridad a los trabajos que dan un resultado directo, y que reducen la burocracia que hay alrededor de la programación, o sea, que reduce la cantidad de artefactos a obtener.

"La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica". (3)

Tiene como principal objetivo la satisfacción del cliente, tratando de ofrecerle el *software* que él necesita y cuando lo necesita. El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo, están involucrados en el desarrollo del *software* y pueden tomar decisiones.

XP trata de evitar algunos de los riesgos que se corren cuando se desarrolla un *software* como son:

- Retrasos en la planificación: llegada la fecha de entregar el *software*, éste aún no está disponible.
- Sistemas deteriorados: el *software* se ha creado, pero transcurridos algunos años el costo de su mantenimiento es tan complicado que definitivamente se abandona su producción.
- Tasa de defectos: el *software* se pone en producción, pero los defectos son tantos que nadie lo usa.
- Requisitos mal comprendidos: el *software* no resuelve los requisitos planificados inicialmente.
- Cambios de negocio: el problema que resolvía el *software* ha cambiado y éste no se ha adaptado.
- Cambios de personal: después de unos años de trabajo, algunos programadores abandonan el proyecto.

Por otra parte, esta metodología define cuatro variables para proyectos de *software*: costo, tiempo, calidad y ámbito. Trata de minimizar la complejidad de un proyecto enfocándose directamente hacia el objetivo, haciendo uso de las relaciones interpersonales y la rapidez de reacción. Se caracteriza por llevar a cabo pruebas unitarias, basadas en pruebas hechas a los procesos de mayor importancia. Permite además refabricación, que no es más que la reutilización de código, siendo el cambio más flexible. Una de las características interesantes de esta metodología, es que permite la programación

en pares, la cual consiste en que dos programadores trabajen una misma estación de trabajo, haciendo más eficiente el trabajo, ya que un programador complementa la tarea del otro y viceversa.

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

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

1.4.1 Selección de la metodología de desarrollo de *software*

Por todas las características vistas anteriormente se decidió utilizar la metodología RUP, pues XP no cumple con los objetivos trazados en este trabajo, ya que está más orientada a la implementación y propone generar poca documentación. RUP realiza un levantamiento exhaustivo de requerimientos, previendo que no se quede fuera nada de lo que el cliente necesita, busca detectar defectos en las fases iniciales, para no tener que retroceder el desarrollo cuando se encuentre en una fase avanzada. Presenta un diseño genérico intentando anticiparse a futuras necesidades, de esta forma, el *software* que se diseñe podrá ser usado por un tiempo más prolongado.

Para el diseño del sistema se utilizarán los siguientes flujos de trabajo: modelado del negocio, requerimientos, análisis y diseño.

1.4.2 Lenguaje Unificado de Modelado (UML)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de *software*.

Es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones *Booch, Rumbaugh* y *Coad-Yourdon*. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representan la arquitectura del proyecto, base sobre la cual se implementará.

UML también intenta solucionar el problema de comunicación que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrolladores se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

Es un estándar, no existe otra especificación de diseño orientado a objetos. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que ha sido

diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama.

Permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir, mediante la restricción se está forzando el comportamiento que debe tener el objeto al que se le aplica.

El modelo gráfico de UML tiene un vocabulario en el que se identifican elementos que se clasifican en:

- Estructurales: partes que representan cosas.
  - Clase: conjunto de objetos que comparten atributos, operaciones, relaciones y semántica.
  - Colaboración: colección de operaciones que especifican un servicio de una clase o un componente.
  - Colaboración: define la interacción entre los elementos que proporcionan un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos.
  - Caso de uso: conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor.
  - Clase activa: clase cuyos objetos tienen uno o más procesos o hilos de ejecución.
  - Componente: es una parte física y reemplazable de un sistema que conforman un conjunto de interfaces y proporciona la implementación de dicho conjunto.
  - Nodo: elemento físico que dispone de memoria y con frecuencia capacidad de almacenamiento.
- Comportamiento: partes del modelo que representan el comportamiento en el tiempo y el espacio.
  - Interacción: conjunto de mensajes intercambiados entre un conjunto de objetos para alcanzar un propósito específico.
  - Máquina de estado: especifica las secuencias de estados por las que pasa un objeto o una interacción durante su vida.
- Agrupamiento: cajas en las cuales puede descomponerse un modelo.
  - Paquete: mecanismo de propósito general para organizar elementos en grupos.
  - Subsistemas.
  - Marco de trabajo.
  - Modelo.
- Anotación: comentarios que se pueden aplicar para describir, clarificar y hacer observaciones sobre cualquier elemento de un modelo.

“Los elementos de UML se muestran mediante diagramas que presentan múltiples vistas del sistema, ese conjunto de vistas es conocido como modelos”**.** (4)

Clasificación de diagramas:

- Diagramas de estructura estática: describen las propiedades estructurales del sistema.
  - Diagrama de clases: conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
  - Diagrama de objetos: conjunto de objetos y sus relaciones.
- Diagramas de comportamiento:
  - Diagramas de interacción (secuencia y colaboración): objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
  - Diagrama de estados: muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.
  - Diagrama de actividad: es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.
  - Diagrama de casos de uso: conjunto de casos de uso y actores y sus relaciones.
- Diagramas de implementación:
  - Diagrama de componentes: organización y las dependencias entre un conjunto de componentes.
  - Diagrama de despliegue: configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

1.4.3 Herramienta de modelado

***Rational Rose Enterprise Edition*:** es el producto más completo de la familia *Rational Rose*. Incluye soporte UML (que es el lenguaje seleccionado para modelar los artefactos del sistema).

Características adicionales incluidas:

- Característica de control por separado tanto de componentes como de modelos, que permite una administración más detallada y facilita el trabajo de analistas y diseñadores.
- Modelado UML para trabajar en diseños de base de datos (BD), con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.

Esta herramienta propone la utilización de cuatro tipos de modelo (caso de usos, lógico, despliegue, implementación) para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del mismo (lógico y físico). Permite crear y refinar estas vistas, creando de esta forma un modelo completo que representa el dominio del problema y el sistema de *software*. Los modelos definidos por la herramienta coinciden con las vistas definidas por la metodología seleccionada aumentando la compatibilidad.

Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. La corporación *Rational Rose* ofrece el Proceso Unificado de Desarrollo, que unifica las mejores prácticas

de muchas disciplinas en un consistente y completo proceso del ciclo de vida, que permite al equipo de desarrollo disminuir los tiempos de liberación.

Otras características:

- Desarrollo iterativo: utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones.
- Trabajo en grupo: permite que varias personas trabajen a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado, que contiene el modelo completo, y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.
- Ingeniería inversa: a partir del código de un programa, se puede obtener información sobre su diseño.

1.5 Plataforma de desarrollo

Plataforma *.NET*: es una capa de *software* que se coloca entre el sistema operativo (SO) y el programador, y que abstrae los detalles internos del SO. Las características fundamentales de esta plataforma son las siguientes:

- Portabilidad: debido a la abstracción del programador respecto al SO, una aplicación *.NET* puede ser ejecutada en cualquier SO de cualquier máquina que disponga de una versión de la plataforma. En estos momentos la plataforma *.NET* tan solo está disponible para la familia *Windows*, aunque se está desarrollando una versión para *Linux* de *Corel.*
- Multilenguaje: cualquier lenguaje de programación puede adaptarse a la plataforma *.NET* y ejecutarse en ella.
- Interoperabilidad: la interoperabilidad entre diferentes trozos de código escritos en diferentes lenguajes es total.

Se pueden desarrollar aplicaciones en múltiples lenguajes dentro de la plataforma *.NET*, además, es una aplicación que puede tener diferentes partes desarrolladas en diferentes lenguajes, y todas se pueden comunicarse entre sí, transparentemente, sin tener que utilizar ningún tipo de capa intermedia que posibilite esta comunicación. Esto permite a su vez, una gran reutilización de código, ya que las clases desarrolladas para un proyecto en un lenguaje concreto, podrán ser reutilizadas en un nuevo proyecto, independientemente del lenguaje en el que se desarrolle éste.

Al igual que en *Java*, el código *.NET* no se compila a código máquina, sino a un código en un formato intermedio, independiente de la plataforma. Esto permite llevar los binarios producidos de una plataforma a otra. Para que esto sea posible, en la plataforma *.NET* existe el lenguaje común de

tiempo de ejecución (CLR), que se encarga de ejecutar el código intermedio, de esta forma, se obtiene independencia de la plataforma.

Algunas ventajas de la plataforma:

- Código administrado: el CLR realiza un control automático del código para que éste sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente, por lo que asegura a clientes y desarrolladores un resultado final más eficiente.
- Recolector de basura: el CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (*garbage collector*). El CLR detecta el momento en que el programa deja de utilizar la memoria y la libera automáticamente. De esta forma, el programador no tiene por qué liberar la memoria de forma explícita, aunque también sea posible hacerlo manualmente, facilitando así el trabajo del equipo de desarrollo.
- Seguridad de acceso al código: se puede especificar que una pieza de código tenga permisos de lectura de archivos, pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar tranquilamente, sin que el mismo se dañe, teniendo en cuenta la importancia del cliente y el tipo de datos que maneja, esta es una ventaja que puede ser muy útil.
- Despliegue: por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El marco de trabajo realiza esta tarea de forma automática, mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones, lo que le brinda al cliente la posibilidad de prescindir del personal requerido para dar mantenimiento a la aplicación.

1.5.1 Lenguaje de desarrollo

***C Sharp* (*C#*):** lenguaje orientado a objetos, es simple, elegante y con seguridad en el tratamiento de tipos, que permite a los programadores de aplicaciones empresariales crear una gran variedad de aplicaciones confiables.

Proporciona la capacidad de generar componentes de sistemas duraderos en virtud de las siguientes características:

- Gran robustez, gracias a la recolección de elementos no utilizados (liberación de memoria) y a la seguridad en el tratamiento de tipos.
- Seguridad implementada por medio de mecanismos de confianza intrínsecos del código.

Además, es posible interaccionar con otros lenguajes, entre plataformas distintas, y con datos heredados, lo que proporciona la posibilidad de integrar la aplicación a otras ya existentes si se hace necesario, debido a:

- Plena interoperabilidad por medio de los servicios del marco de trabajo de *.NET* con un acceso limitado basado en bibliotecas.
- Compatibilidad con el Lenguaje de Marcación Extendido (XML*),* característica muy útil en el caso del sistema que se pretende desarrollar, debido a la necesidad de importar y exportar información de las bases de datos de las diferentes UPs y la OAMS.
- Su sintaxis básica se deriva de C/C++ y utiliza el modelo de objetos de la plataforma *.NET*, el cual es similar al de *Java* aunque incluye mejoras derivadas de otros lenguajes (entre ellos *Delphi*).

Algunas de sus ventajas son:

- Elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en *.NET.*
- Incorpora elementos muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como *Java* o C++ hay que simular, facilitando el trabajo de los desarrolladores y la calidad del producto final.

Es adecuado para escribir aplicaciones desde las más grandes y sofisticadas como sistemas operativos, hasta las más pequeñas funciones. Es un lenguaje nuevo y a la vez maduro. Para crear *C#* se ha cogido lo mejor de *Visual Basic*, C++ y *Java,* y mejorado aquello en lo que éstos fallaban. Tiene una serie de herramientas completas para tratar cadena de caracteres (lo que facilitará la comparación entre los nombres de los nomencladores y determinar si ya existen ante la acción de insertar), una gestión de memoria más rápida (disminuyendo el tiempo de respuesta) y listas genéricas. Este lenguaje presenta decenas de mejoras en extensibilidad de tipos, de componentes y operadores. Permite el uso de instrucciones seguras, que facilitan el manejo de excepciones dando mayor seguridad al sistema.

1.5.2 Herramienta de desarrollo

***Microsoft Visual Studio*:** es un entorno de desarrollo integrado (IDE) para sistemas *Windows*. Es una herramienta de desarrollo profesional para programadores individuales o para aquellos que trabajen en pequeños equipos. Soporta varios lenguajes de programación tales como *Visual C++*, *Visual C#*, *Visual J#*, *ASP.NET* y *Visual Basic .NET*, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. *Visual Studio* permite a los desarrolladores crear sitios, aplicaciones *web* y de escritorio, así como servicios *web* en cualquier entorno que soporte la plataforma *.NET* (a partir de la versión *.NET* 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas *web* y dispositivos móviles.

***Visual Studio* 2005** proporciona una amplia gama de herramientas que ofrecen multitud de ventajas para desarrolladores individuales y equipos de desarrollo de *software.*

Ventajas:

- Tiene mayor productividad y obtención más rápida de resultados disminuyendo el tiempo de respuesta.
- Crea soluciones dinámicas basadas en *Windows*, la *Web*, dispositivos móviles y *Office*.
- Tiene comunicación y colaboración más eficaz en sus equipos de *software*.
- Garantiza calidad rápida y continua en todo el proceso de desarrollo.
- Ayuda con refactorización.
- Soporte de pruebas para todo tipo de aplicaciones.

1.6 Sistema gestor de base de datos

Como gestor de base de datos se seleccionó ***PostgreSQL.***

Algunas de las características de este gestor son:

- Atomicidad (indivisible): es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema, no puede quedar a medias, asegurando de esta forma que no se pierda información importante que puede reportar grandes sumas de dinero.
- Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto, se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema, característica de gran importancia debido a que la mayoría de las operaciones está vinculadas con grandes cantidades de dinero y sería catastrófico la pérdida de los datos de una operación.
- Corre en casi todos los principales sistemas operativos: *Linux*, *Unix*, *BSDs*, *Mac OS*, *Beos*, *Windows*, entre otros.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios, facilitando de esta forma el trabajo del usuario con la base de datos.
- Comunidades muy activas, varias comunidades en castellano, lo cual facilita la investigación del gestor.
- Altamente adaptable a las necesidades del cliente, facilitando los posibles cambios que puedan surgir en un futuro, ya que la OAMS está buscando vías de optimización y por tanto es susceptible a los cambios.

- Utilidades para limpieza de la base de datos, permitiendo de esta forma evacuar los datos en otro lugar y volver a tener la base de datos limpia para los datos más actuales.
- Utilidades para análisis y optimización de consultas, haciendo así más eficiente el trabajo de los desarrolladores al facilitar los procesos de obtención de datos.
- Almacenaje especial para tipos de datos grandes, esta característica es muy importante para el sistema debido a que en determinado punto se manejarán grandes cifras de dinero y de otros conceptos como la demanda de un alimento para un individuo.
- Varios tipos de índices.

Se utilizará específicamente la versión ***PostgreSQL* 8.4**.

Esta versión se caracteriza por:

- Mejoras de rendimiento: la versión 8.4 mejora el rendimiento alrededor de un 20% en total. Los cambios incluyen ordenamientos más rápidos en disco y en memoria, planes de ejecución en consultas sobre datos particionados y cargas masivas más rápidas.
- Construcción en línea de índices: la construcción de índices puede ocurrir mientras otras aplicaciones escriben en tablas de la base de datos, permitiendo optimización de rendimiento sin afectar la disponibilidad.
- Índices generalizados invertidos: soporta una manera más escalable y programable de indexación semiestructuradas y de datos de texto completo.

Características adicionales:

- Bloqueos consultivos: permiten el control de objetos de bases de datos a nivel de aplicación, usando el motor rápido de bloqueos de *PostgreSQL.*
- Sentencias preparadas: tiene nuevas interfaces administrativas y mejoras de rendimiento en sentencias preparadas.

1.6.1 Lenguaje de base de datos

Structure Query Language* (*SQL*)*: es el lenguaje estándar *ANSI/ISO* de definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo: sólo hay que indicar qué se quiere hacer. En cambio, en los lenguajes procedimentales es necesario especificar cómo hay que hacer cualquier acción sobre la base de datos, por tanto, facilita el trabajo de los desarrolladores. Es muy parecido al lenguaje natural y muy expresivo, facilitando así su entendimiento.

“Una de sus características es el manejo del álgebra y el cálculo relacional, permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre la misma”. (5)

"El *Structured Query Language* no es más que un lenguaje estándar de comunicación con bases de datos". (6)

Se divide en dos secciones:

1. Lenguaje de definición de datos (DDL), incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre éstas. Normalmente, las instrucciones incluidas para aplicar sobre diferentes objetos son *CREATE, ALTER, DROP.*

2. Lenguaje de manipulación de datos (DML), permite recuperar los datos almacenados en la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados. La manipulación se hace a nivel de filas.

El modelo relacional tiene como estructura de almacenamiento de los datos, las relaciones. El esquema de una relación consiste en el nombre que se ha dado a la relación y un conjunto de atributos. La extensión de una relación es un conjunto de filas. Al trabajar con *SQL*, esta nomenclatura cambia, como se puede apreciar en la siguiente figura:

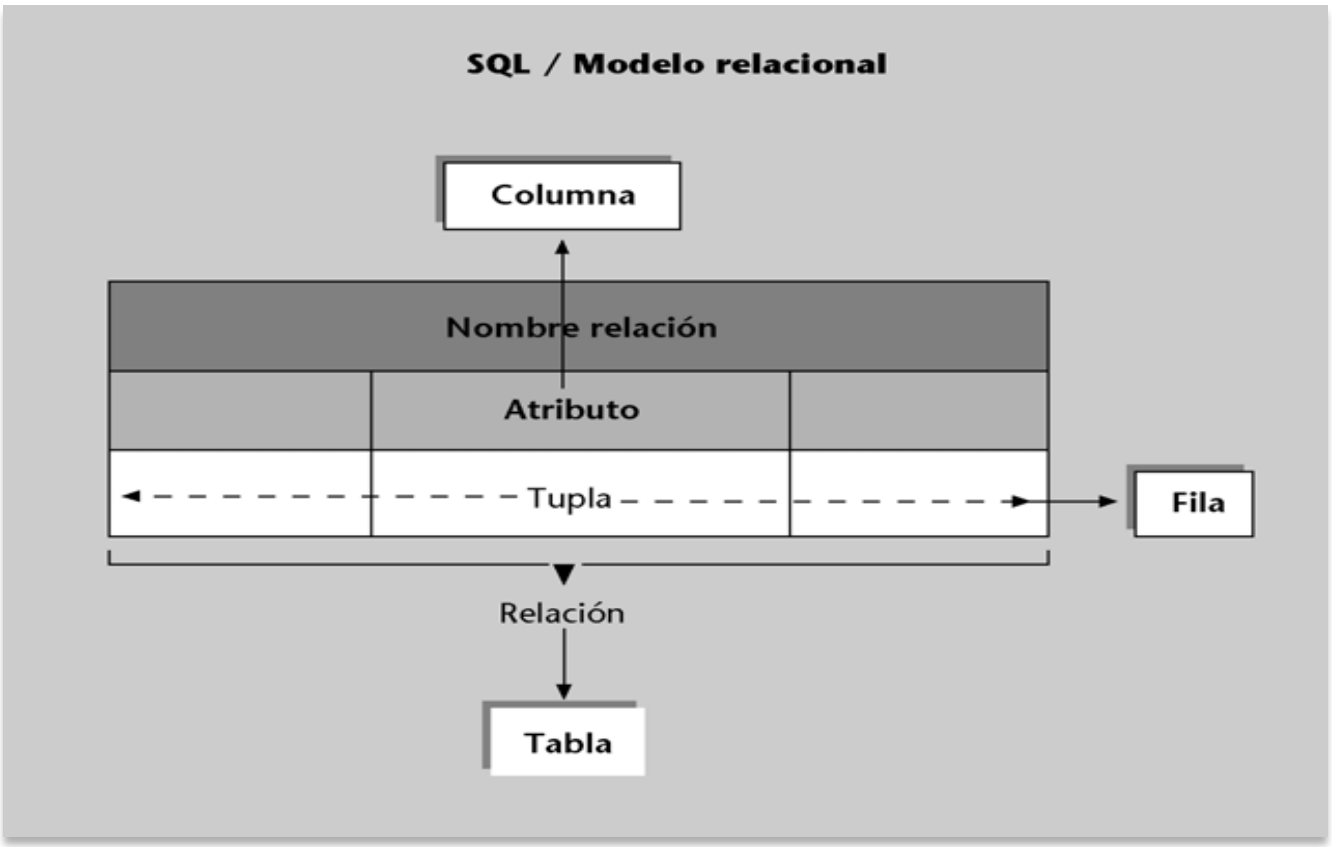


Figura 1: Imagen representativa de SQL

• Se habla de columnas en lugar de atributos.

• Se habla de filas en lugar de tuplas.

Sin embargo, a pesar de que la nomenclatura utilizada sea diferente, los conceptos son los mismos.

Otras características del *SQL*:

- Es una forma estándar de consulta de datos específicos.
- Es una forma de extraer y manipular datos de una base de datos.
- Usado para todas las funciones de bases de datos, incluyendo administración.
- Creación de esquemas y datos recuperables.

1.6.2 Herramienta para el modelado de la base de datos

***Erwin Studio* (*ER/Studio*):** Provee una interfaz visual de fácil utilización para documentar, entender y publicar información acerca de las bases de datos existentes, de tal forma que puedan ser mejor controladas para soportar los objetivos del negocio. Es una herramienta multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejas.

El *ER/Studio* facilita la definición y reutilización de los elementos comunes de los datos y elementos de modelado, lo que permite entender y utilizar mejor los datos, reducir la redundancia y tener mayor consistencia.

1.7 Conclusiones

Este capítulo permitió el estudio de cada uno de los elementos necesarios para desarrollar una aplicación que satisfaga al cliente. Se propone utilizar como metodología RUP, ya que expone un conjunto de actividades que están orientadas a visualizar, especificar, construir y documentar los artefactos necesarios para el desarrollo de *software* con calidad. RUP hace uso del UML, cuya utilización de diagramas y gráficos brindan una mejor perspectiva de lo que se quiere. Como herramienta de modelado se hará uso del *Rational Rose Enterprise Edition.* Para el desarrollo del sistema se propone:

- La herramienta *Visual Studio* 2005 *Professional.*
- El lenguaje de programación *C#*.
- Gestor de base de datos *PostgreSQL.*
- Lenguaje de base de datos *SQL.*
- Herramienta de modelado de la base de datos *ER/Studio.*

Capítulo 2 Características del sistema

2.1 Introducción

En este capítulo se realiza la descripción de los procesos del negocio enmarcados dentro del campo de acción, se enumeran los requisitos funcionales y no funcionales con los que debe cumplir el sistema. Se realiza además, la identificación de los actores, trabajadores, casos de uso, entidades y las relaciones existentes entre ellos a través de los modelos de casos de uso del negocio y del sistema.

2.2 Descripción de los procesos del negocio

Durante el estudio del negocio, se identificaron varios procesos que constituyen el pilar fundamental del desarrollo del mismo. La OAMS es la responsable de asignar los recursos necesarios a las unidades presupuestadas. Para ello se realiza un plan de productos alimenticios y otro de materiales de aseo e insumos, estos planes pueden ser mensuales, trimestrales, semestrales o anuales.

Los planes se diferencian en cuanto a los datos que se definen, para ambos se tiene en cuenta el nombre de la UP, la descripción del producto o material, el precio, el costo (en moneda nacional y en divisa), el proveedor, la unidad de medida, la cantidad del producto o del material y la fecha de confección. Para el plan de productos alimenticios se tienen en cuenta además los siguientes aspectos: categoría, norma, frecuencia, tipo de frecuencia, demanda (para estudiantes y trabajadores), demanda total, plan real del alimento. Por otra parte, para el plan de materiales de aseo e insumos se tienen en cuenta el modelo y el plan del año. Los planes finales están en correspondencia a la cantidad que existe en el almacén de la UP de los productos alimenticios y de los materiales de aseo e insumos, y a la cantidad que se debería entregar. Estos planes se entregan a las distintas empresas proveedoras que tienen contrato con la OAMS.

Una vez entregado el plan a los proveedores, las unidades presupuestadas pueden solicitarle recursos. La solicitud de los productos alimenticios y de los materiales de aseo e insumos para los almacenes de las UPs, debe estar en correspondencia con los planes realizados por la OAMS.

Cuando el producto llega al almacén de la UP, se anotan los datos correspondientes al mismo en una factura; con dichos datos se elabora al final de mes, un balance de todos los productos y materiales que han entrado o salido del almacén en dicha UP. Este balance también se realiza en las sedes, agrupando las UPs que se le subordinan, y en la OAMS a su vez, para tener el control de todas las sedes.

2.3 Modelación del negocio

2.3.1 Determinación y justificación de los actores del negocio

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

Actor	Descripción
Subdirector de aseguramiento de la OAMS.	Es el encargado de planificar y controlar los recursos destinados a las sedes y unidades presupuestadas.

Tabla 1: Actor del negocio.

2.3.2 Determinación y justificación de los trabajadores del negocio

Un trabajador del negocio representa a un ser humano, *software* o *hardware* que desempeña un rol dentro de las realizaciones de un caso de uso del negocio.

Trabajador	Descripción
Almacenero de la unidad presupuestada.	Es el encargado de llevar el control de los recursos que se le asignan a su unidad presupuestada.
Encargado de aseguramiento en la sede.	Es el encargado de llevar el control de los recursos que se le asignan a las unidades presupuestadas subordinadas a dicha sede.
Proveedor.	Es el encargado de proveer los recursos que necesitan las sedes según la planificación.
Económico de la unidad presupuestada.	Es el encargado de llevar la contabilidad en la unidad presupuestada.

Tabla 2: Trabajadores del negocio.

2.3.3 Diagrama de casos de uso del negocio

Un diagrama de casos de uso del negocio, representa la interacción entre los actores y los procesos que se desarrollan en el mismo.

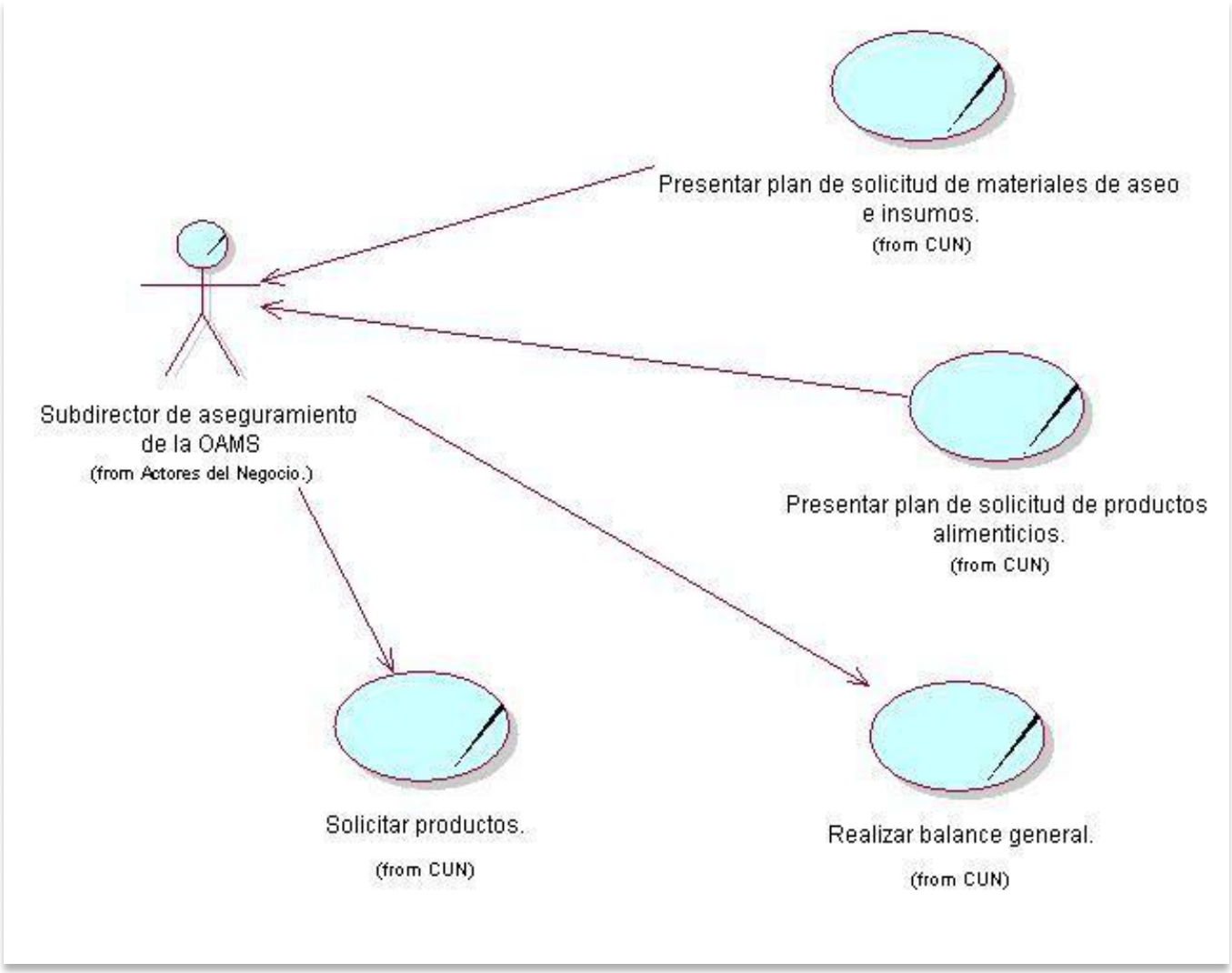


Figura 2: Diagrama de casos de uso del negocio.

2.3.4 Diagrama de objetos

El diagrama de objetos representa la interacción entre los trabajadores del negocio y las entidades que se manejan en el mismo.

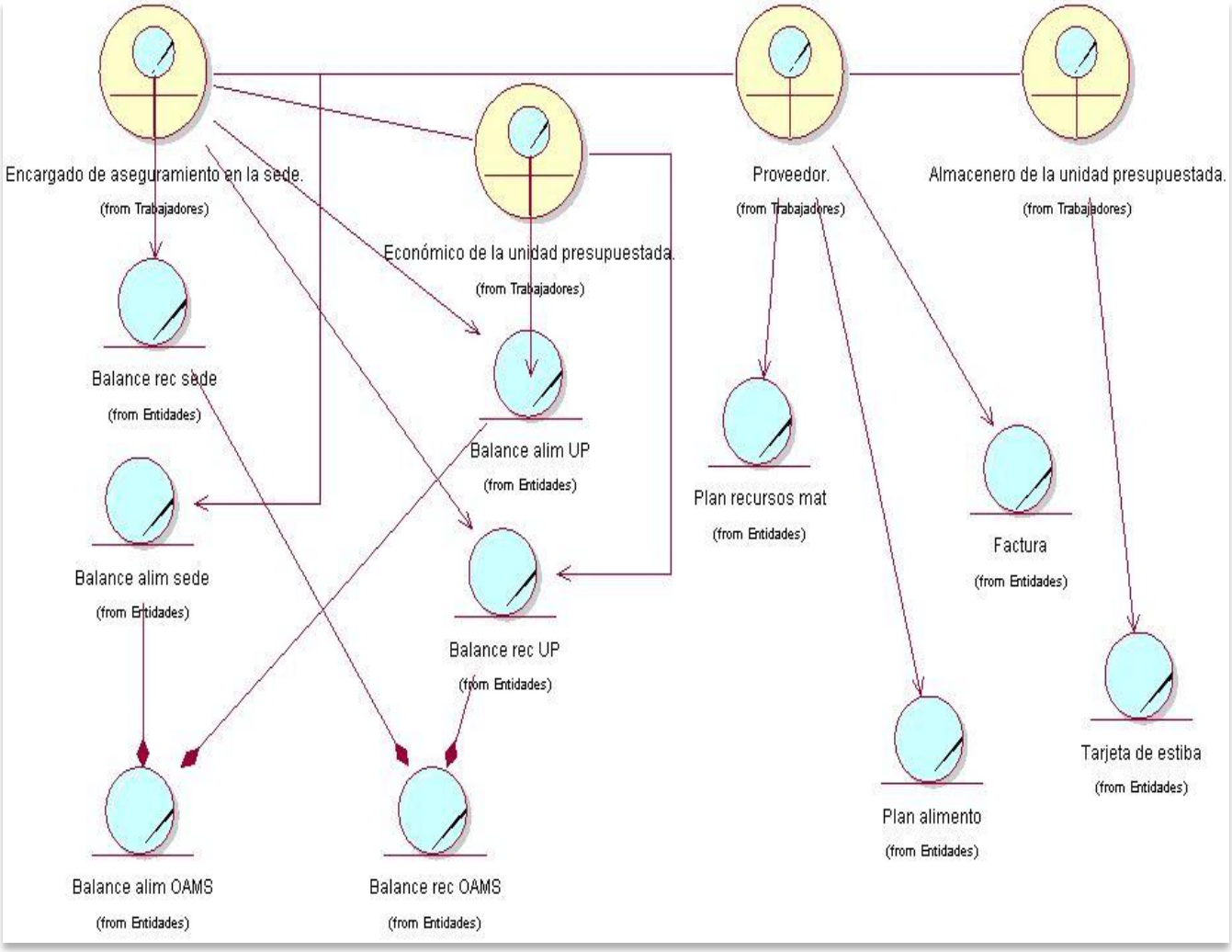


Figura 3: Diagrama de objetos del negocio.

2.3.5 Descripción de los casos de uso del negocio

Caso de uso del negocio	Presentar plan de solicitud de materiales de aseo e insumos.
Actores	Subdirector de aseguramiento de la OAMS.
Resumen	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS recibe el balance material de cada unidad presupuestada, con esta información planifica que cantidad de materiales de aseo e insumos necesita cada unidad presupuestada, teniendo en cuenta la cantidad en existencia de cada producto, el caso de uso finaliza cuando este plan es entregado a las unidades proveedoras con las que la OAMS tiene contrato.
Acción del actor	**Respuesta del proceso de negocio**

	1. El encargado de aseguramiento en la sede envía el balance material de cada unidad presupuestada al subdirector de aseguramiento de la OAMS, que le permite conocer la existencia real de cada producto.
2. El subdirector de aseguramiento de la OAMS, una vez conocida la existencia real de cada producto, elabora un plan donde le asigna a cada unidad presupuestada una cantidad determinada de cada producto teniendo en cuenta el tamaño de la unidad, la cantidad de oficinas, la cantidad de estudiantes y trabajadores.	
3. El subdirector de aseguramiento de la OAMS entrega el plan a los proveedores. 4. Finaliza el caso de uso del negocio (CUN).	

Tabla 3: Descripción del CUN Presentar plan de solicitud de materiales de aseo e insumos

Caso de uso del negocio	Presentar plan de solicitud de productos alimenticios.
Actores	Subdirector de aseguramiento de la OAMS.
Resumen	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS recibe el balance material de cada unidad presupuestada, con esta información planifica que cantidad de productos alimenticios necesita cada unidad presupuestada, teniendo en cuenta la cantidad en existencia de cada alimento, el caso de uso finaliza cuando este plan es entregado a las unidades proveedoras con las que la OAMS tiene contrato.
Acción del actor	**Respuesta del proceso de negocio**
	1. El encargado de aseguramiento en la sede envía el balance material de cada unidad presupuestada al subdirector de aseguramiento de la OAMS, que le permite conocer la existencia real de cada producto alimenticio.
2. El subdirector de aseguramiento de la OAMS, una vez conocida la existencia real de cada producto alimenticio, elabora un plan donde le asigna a cada unidad	

presupuestada una cantidad determinada de dichos alimentos teniendo en cuenta la norma, la frecuencia y la cantidad de comensales.	
3. El subdirector de aseguramiento de la OAMS entrega el plan de productos alimenticios a los proveedores. 4. Finaliza el CUN.	

Tabla 4: Descripción del CUN Presentar plan de solicitud de productos alimenticios.

Caso de uso del negocio	Realizar balance general.
Actores	Subdirector de aseguramiento de la OAMS (inicia).
Resumen	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS le pide al encargado de aseguramiento en la sede el balance general de las unidades presupuestadas y de la propia sede y termina cuando el subdirector de aseguramiento de la OAMS recibe el balance de cada sede (con el de cada unidad presupuestada) y con éste realiza el balance general de la OAMS.
Acción del actor	**Respuesta del proceso de negocio**
1. El subdirector de aseguramiento de la OAMS le pide al encargado de aseguramiento en la sede el balance general de las unidades presupuestadas.	2. El encargado de aseguramiento en la sede le pide entonces el balance general al económico de la unidad presupuestada.
	3. El económico de la unidad presupuestada con las entradas y salidas del mes confecciona el balance general y se lo envía al encargado de aseguramiento en la sede.
	4. El encargado de aseguramiento en la sede con los datos de los balances generales de las unidades presupuestadas que se le subordinan, confecciona un balance general de la sede y se lo envía al subdirector de aseguramiento de la OAMS.
5. El subdirector de aseguramiento de la OAMS con los datos del balance general de las sedes confecciona el balance general de la OAMS. 6. Finaliza el CUN.	

Tabla 5: Descripción del CUN Realizar balance general.

Caso de uso del negocio	Solicitar productos
Actores	Subdirector de aseguramiento de la OAMS (inicia).
Resumen	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS informa al encargado de aseguramiento en la sede, que ya puede solicitar a los proveedores contratados por la OAMS, los productos que necesita para una sede en un determinado período de tiempo, teniendo en cuenta el plan entregado por el subdirector de aseguramiento de la OAMS y termina cuando la unidad presupuestada recibe el producto.
Acción del actor	**Respuesta del proceso de negocio**
1. El subdirector de aseguramiento de la OAMS le informa al encargado de aseguramiento en la sede que ya puede realizar la solicitud.	2. El encargado de aseguramiento en la sede solicita a los proveedores contratados por la OAMS los productos que necesita para una sede en un determinado período de tiempo.
	3. El proveedor verifica la existencia del producto y que el pedido esté en correspondencia con el plan entregado por el subdirector de aseguramiento de la OAMS.
	4. El proveedor traslada a la unidad presupuestada el producto.
	5. El almacenero de la unidad presupuestada recibe los productos y anota en la tarjeta de estiba los datos de la factura.
	6. El almacenero le entrega una copia de la factura al proveedor y otra al económico de la unidad presupuestada. 7. Finaliza el CUN.

Tabla 6: Descripción del CUN Solicitar productos.

2.3.6 Diagramas de actividades de los casos de uso del negocio

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

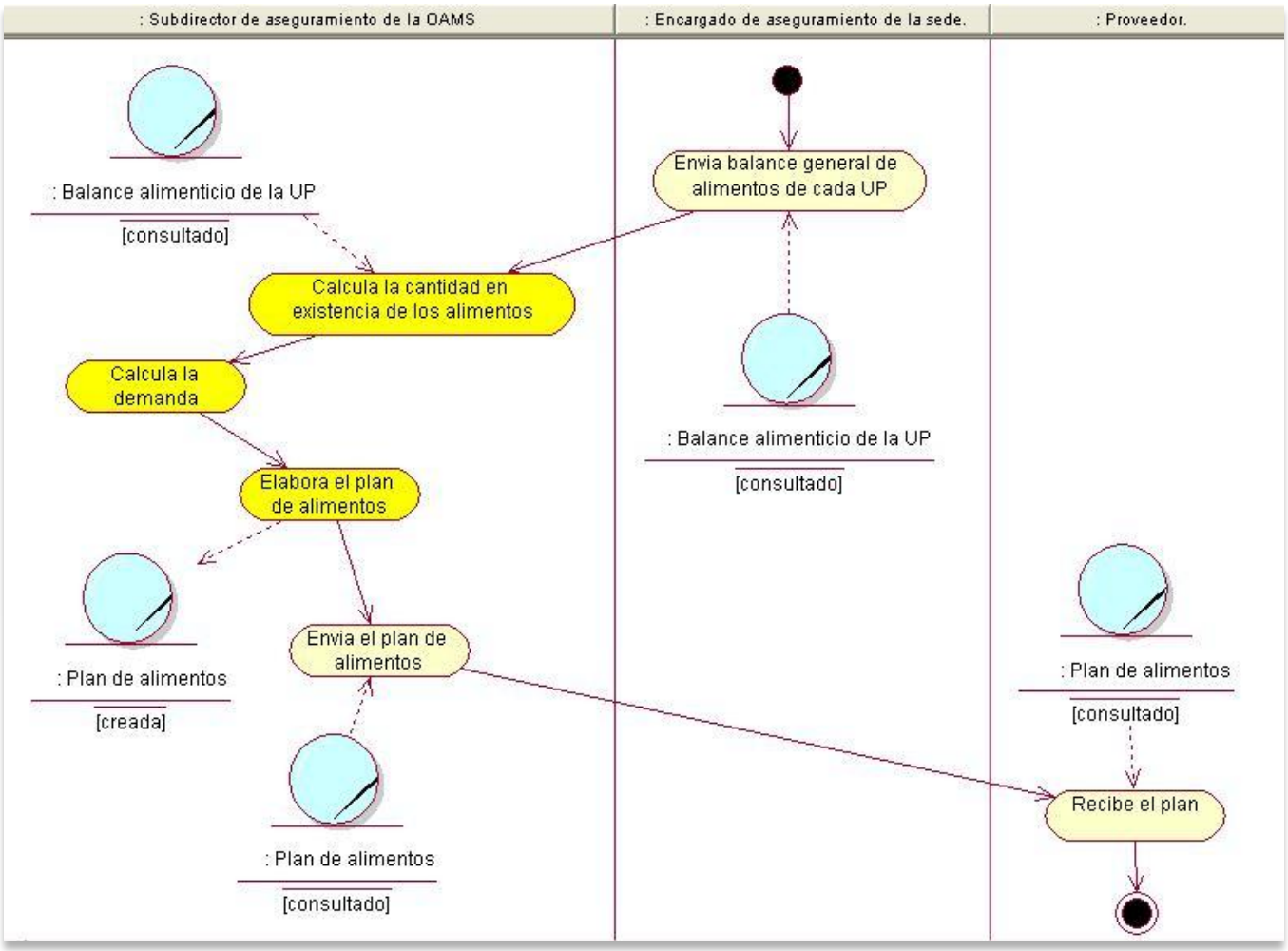


Figura 4: Diagrama de actividades del CUN Presentar plan de solicitud de productos alimenticios.

Los restantes diagramas de actividades se encuentran en: **¡Error! No se encuentra el origen de la referencia.**.

2.3.7 Reglas del negocio

Una regla del negocio es la declaración de políticas y restricciones de negocio que el sistema debe cumplir.

Reglas del negocio:

- Para poder realizar el balance de la sede es necesario que todas las UPs hayan entregado los balances.
- La demanda de un alimento es la norma multiplicada por la frecuencia y la cantidad de comensales.
- A la suma del costo, precio y la cantidad de productos de un mismo tipo que entran a una UP se le denomina entradas, y a la cantidad saliente se le denomina salidas.

- La existencia final de un producto, es la cantidad inicial sumada con las entradas menos las salidas.
- Para poder realizar el balance de la OAMS es necesario que todas las sedes hayan entregado el suyo.
- A una sede no se le puede aceptar el pedido de una UP si ésta ya ha consumido la cantidad de productos establecidos en el plan.
- Para el proveedor poder suministrar un producto a una sede debe tener el plan de asignación de recursos.
- Si llegada la fecha de cierre alguna UP no ha entregado su balance, la sede le notifica.
- Si llegada la fecha de cierre alguna sede no ha entregado su balance, la OAMS le notifica.
- A la asociación entre las entradas y las salidas de un mes, se denomina balance.
- A la asignación de una cantidad de recursos determinados para cierta UP se le denomina plan.

2.4 Descripción del sistema

Los procesos anteriormente descritos son los que se deben automatizar para dar solución al problema planteado.

Se propone el desarrollo de una aplicación de escritorio, que permita gestionar la información que se genera en los almacenes de las UPs subordinadas a la OAMS, y de esta forma facilitar la gestión de la información de la misma.

Esta aplicación debe dar la posibilidad de realizar un balance en las UPs donde se registren los datos de entrada y salida del mes de cada producto, de esta forma, se conocerá cuánto se ha consumido respecto a recursos financieros y recursos materiales en la unidad, debe permitir además, que se calcule la cobertura de un producto en dependencia de la existencia final del mismo. La aplicación debe permitir también realizar un balance a nivel nacional, en el que se utilicen los datos de los balances de las unidades, para conocer el consumo en moneda nacional y en divisa de la OAMS.

Por otra parte, debe posibilitar que la OAMS asigne los recursos existentes a las UPs a través de planes, que contendrán la cantidad a asignar de cada recurso en dependencia de la cantidad en existencia del mismo en el almacén de la unidad en cuestión.

Debido a que el sistema se encontrará distribuido en distintas unidades sin conexión entre sí, la aplicación debe permitir exportar o importar los datos necesarios entre las UPs y la OAMS. En momentos determinados se deben conocer datos provenientes de los balances o los planes, por tanto, se hace necesario la realización de reportes, como por ejemplo: reportes de balances provinciales (a nivel de UP), de planes, y de balances nacionales.

La aplicación debe contar además con diversas funcionalidades a las cuáles no todas las personas tendrán acceso, por lo que es necesario definir roles y usuarios, por tanto, cuando alguien acceda a la aplicación deberá autenticarse, y en dependencia del rol que éste desempeñe, se determinará su acceso a las mismas.

Para la realización de los planes y balances es imprescindible la gestión de los nomencladores que se detectaron en el estudio del negocio: categorías, alimentos, modelos, insumos, UPs y unidades de medidas.

2.4.1 Especificación de los requerimientos

2.4.1.1 Requerimientos funcionales:

RF 1 Gestionar alimentos.

RF 1.1 Insertar datos de alimentos: descripción, norma, categoría, unidad de medida.

RF 1.2 Modificar datos de alimentos: descripción, norma, categoría, unidad de medida.

RF 1.3 Eliminar alimento.

RF 2 Gestionar materiales de aseo e insumos.

RF 2.1 Insertar datos del material: modelo, descripción, unidad de medida.

RF 2.2 Modificar material: modelo, descripción, unidad de medida.

RF 2.3 Eliminar material.

RF 3 Gestionar unidad presupuestada.

RF 3.1 Insertar datos de la unidad: nombre, total de estudiantes, total de trabajadores.

RF 3.2 Modificar datos de la unidad: nombre, total de estudiantes, total de trabajadores.

RF 3.3 Eliminar unidad.

RF 4 Gestionar plan de alimentos.

RF 4.1 Crear plan: unidad presupuestada, fecha, frecuencia para estudiantes, frecuencia para trabajadores, plan real, demanda total.

RF 4.2 Modificar plan: frecuencia para estudiantes, frecuencia para trabajadores, plan real, demanda total.

RF 4.3 Eliminar plan.

RF 5 Gestionar plan de insumos.

RF 5.1 Crear plan: unidad presupuestada, fecha, cantidad.

RF 5.2 Modificar plan: cantidad.

RF 5.3 Eliminar plan.

RF 6 Gestionar balance de alimentos.

RF 6.1 Crear balance: unidad presupuestada, fecha, entradas por factura, entradas por devolución, entradas por transferencias, entradas por sobrantes, salidas por consumo, salidas por transferencias, salidas por merma, salidas por faltante, existencia inicial, total entradas, total salidas, precio en moneda nacional, precio en divisa, importe en moneda nacional, importe en divisa, cobertura.

RF 6.2 Modificar balance: entradas por factura, entradas por devolución, entradas por transferencias, entradas por sobrantes, salidas por consumo, salidas por transferencias, salidas por merma, salidas por faltante, existencia inicial, total entradas, total salidas, precio en moneda nacional, precio en divisa, importe en moneda nacional, importe en divisa, cobertura.

RF 6.3 Eliminar balance.

RF 7 Gestionar balance de insumos.

RF 7.1 Crear balance: unidad presupuestada, fecha, entradas por factura, entradas por devolución, entradas por transferencias, entradas por sobrantes, salidas por consumo, salidas por transferencias, salidas por merma, salidas por faltante, existencia inicial, total entradas, total salidas, precio en moneda nacional, precio en divisa, importe en moneda nacional, importe en divisa.

RF 7.2 Modificar balance: unidad presupuestada, fecha, entradas por factura, entradas por devolución, entradas por transferencias, entradas por sobrantes, salidas por consumo, salidas por transferencias, salidas por merma, salidas por faltante, existencia inicial, total entradas, total salidas, precio en moneda nacional, precio en divisa, importe en moneda nacional, importe en divisa.

RF 7.3 Eliminar balance.

RF 8 Realizar balance nacional: fecha.

RF 9 Gestionar categoría.

RF 9.1 Insertar datos de la categoría: nombre.

RF 9.2 Modificar categoría: nombre

RF 9.3 Eliminar categoría.

RF 10 Generar reportes.

RF 10.1 Reportar balance de alimentos.

RF 10.2 Reportar balance de insumos.

RF 10.3 Reportar balance nacional.

RF 10.4 Reportar plan de alimentos.

RF 10.5 Reportar plan de insumos.

RF 11 Gestionar modelos.

RF 11.1 Insertar datos del modelo: nombre, período.

RF 11.2 Modificar modelo: nombre, período.

RF 11.3 Eliminar modelo.

RF 12 Autenticar usuario.

RF 12.1 Permitir que el usuario entre al sistema dado el usuario y la contraseña.

RF 12.2 Permitir que el usuario cambie su contraseña una vez autenticado.

RF 12.3 Permitir que el usuario sólo pueda acceder a las funcionalidades determinadas por su rol.

RF 13 Gestionar roles.

RF 13.1 Crear rol: usuario, permisos.

RF 13.2 Modificar rol: usuario, permisos.

RF 13.3 Eliminar rol.

RF 14 Gestionar usuario.

RF14.1 Insertar usuario: nombre, usuario, contraseña y rol.

RF14.2 Modificar datos de un usuario: nombre, usuario, contraseña y rol.

RF14.3 Eliminar cuenta de usuario.

RF 15 Gestionar unidades de medidas.

RF 15.1 Crear unidad de medida: nombre, abreviatura.

RF 15.2 Modificar unidad de medida: nombre, abreviatura.

RF 15.3 Eliminar unidad de medida.

RF 16 Importar y exportar datos: alimentos, productos, categorías, modelos, unidades presupuestadas, unidades de medida, planes de alimentos, planes de productos, balances de alimentos, balances de productos.

2.4.1.2 Requerimientos no funcionales

Usabilidad:

El sistema debe ser usable por cualquier tipo de usuario con experiencia básica, media o avanzada en los procesos de análisis de la información proveniente de almacenes.

La información debe mostrarse de forma lógica y correctamente estructurada.

El sistema debe ser fácil de instalar.

Los usuarios deben pasar una capacitación de una semana.

Apariencia o interfaz externa:

El sistema debe tener un menú de herramientas para que el usuario pueda disponer de él en la medida de sus necesidades, visible todo el tiempo para propiciar el fácil acceso a las funcionalidades del *software*. Debe tener además un ambiente agradable y sencillo, que permita que el usuario se adapte con facilidad.

Rendimiento:

El sistema debe garantizar un tiempo de respuesta rápido, que variará en dependencia de la complejidad del servicio.

Seguridad:

Se deben aplicar las reglas de la "programación segura", mediante el tratamiento de excepciones, en cuanto a tipos de entradas, campos vacíos, entre otros.

El sistema debe contar con la acción de autenticación, para regular la entrada al mismo y las acciones que los usuarios puedan realizar, además se debe garantizar la seguridad de la contraseña mediante un método de encriptación.

Hardware:

Como mínimo se debe contar con computadoras *Intel Celeron* a 2.80 *GHz*, superior o equivalente, disco duro de 40 *GB* y 512 *MB* de *RAM*.

Software:

Las estaciones de trabajo donde se encuentre instalado el *software*, deben tener como sistema operativo versiones superiores o iguales a *WindowsXP*, como gestor de base de datos *PostgreSQL8.4* y además el marco de trabajo de *.NET* en su versión 3.5.

Ayuda:

El sistema debe contar con la opción de ayuda, que estará visible todo el tiempo para informar al usuario acerca de un procedimiento, en caso de que éste tenga alguna duda.

Confiabilidad:

La información manejada debe estar protegida de acceso no autorizado y de divulgación.

Soporte:

El servidor de BD debe soportar grandes volúmenes de datos y tener buena velocidad de procesamiento.

2.4.2 Modelo de casos de uso del sistema

Un modelo de casos de uso describe los requerimientos funcionales de un actor partiendo de las interacciones que éste ejecuta con el sistema.

2.4.2.1 Descripción de los actores del sistema

Actor	Descripción
Subdirector de aseguramiento de la OAMS.	Es el encargado de planificar y controlar los recursos destinados a las unidades presupuestadas.
Económico de la unidad presupuestada.	Es el encargado de llevar la contabilidad en la unidad presupuestada.

Administrador	Es el encargado de gestionar los diferentes roles con que contará el sistema, de crear los usuarios y asignarles roles en dependencia de la labor que realicen.
Usuario	Realiza las acciones comunes para los demás actores del sistema.
Usuario auxiliar	Realiza acciones comunes para el subdirector de aseguramiento de la OAMS y para el económico de la UP.

Tabla 7: Actores del sistema.

2.4.2.2 Diagrama de casos de uso del sistema

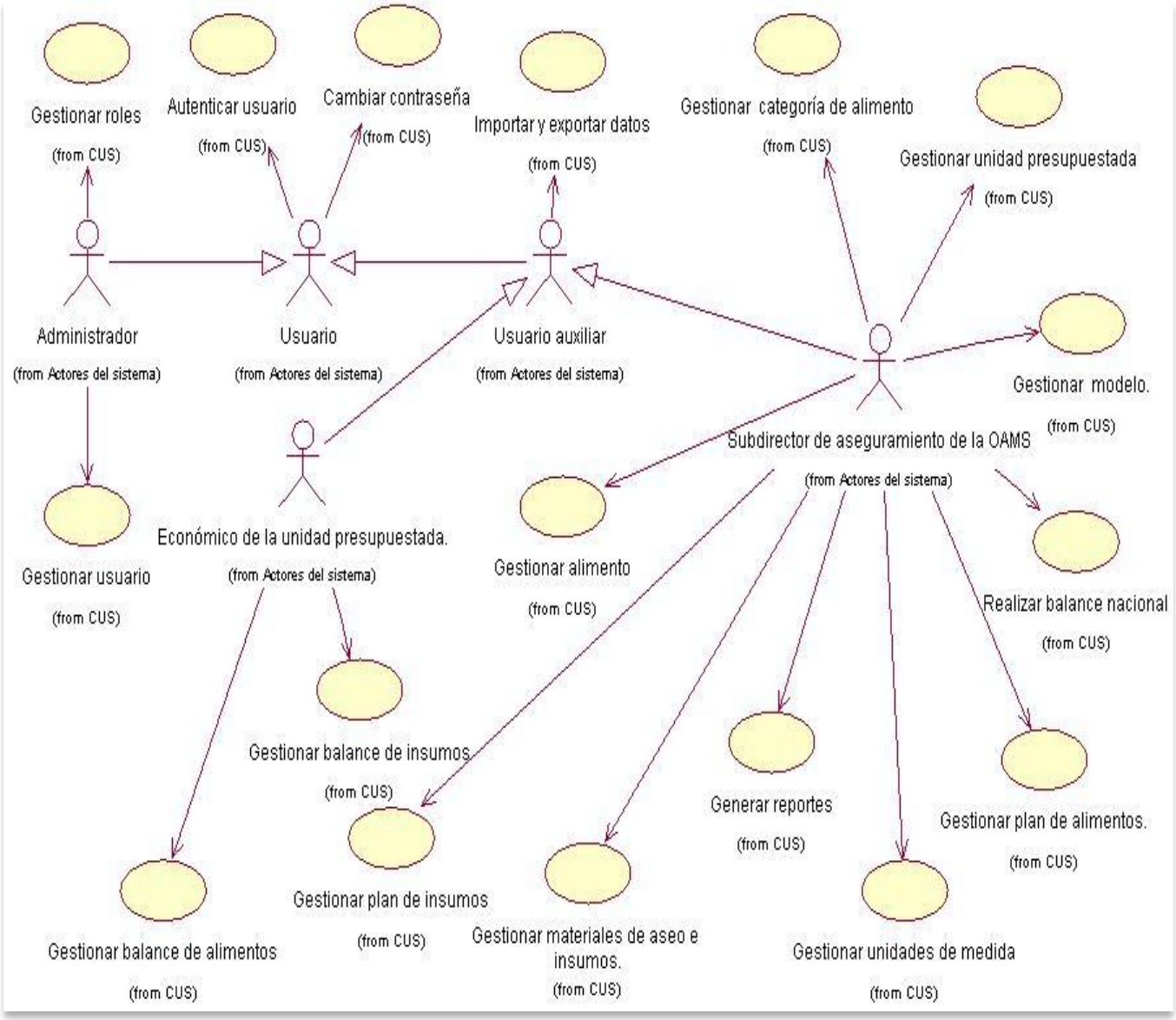


Figura 5: Diagrama de casos de uso del sistema.

2.4.2.3 Descripción de los casos de uso del sistema

Caso de Uso:	Gestionar alimento.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a crear, modificar o eliminar un alimento. El caso de uso del sistema (CUS) finaliza cuando se ejecuta alguna de estas acciones.

Tabla 8: Resumen del CUS Gestionar alimento.

Caso de Uso:	Gestionar materiales de aseo e insumos.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a crear, modificar o eliminar un producto. El CUS finaliza cuando se ejecuta alguna de estas acciones.

Tabla 9: Resumen del CUS Gestionar producto de aseo e insumos.

Caso de Uso:	Gestionar modelo.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a crear, modificar o eliminar un modelo. El CUS finaliza cuando se ejecuta alguna de estas acciones.

Tabla 10: Resumen del CUS Gestionar modelo.

Caso de Uso:	Gestionar unidad presupuestada.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a crear, modificar o eliminar una unidad presupuestada. El CUS finaliza cuando se ejecuta alguna de estas acciones.

Tabla 11: Resumen del CUS Gestionar UP.

Caso de Uso:	Gestionar plan de alimentos.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a confeccionar, modificar o eliminar un plan de alimentos. El CUS finaliza cuando el plan queda confeccionado, modificado o eliminado.

Tabla 12: Resumen del CUS Gestionar plan de alimentos.

Caso de Uso:	Gestionar plan de insumos.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a

	confeccionar, modificar o eliminar un plan de insumos. El CUS finaliza cuando el plan queda confeccionado, modificado o eliminado.

Tabla 13: Resumen del CUS Gestionar plan de insumos.

Caso de Uso:	Gestionar balance de alimentos.
Actores:	Económico de la unidad presupuestada (inicia).
Resumen:	El caso de uso comienza cuando el económico de la unidad presupuestada accede a crear, modificar o eliminar un balance de alimentos. El CUS finaliza cuando el balance mensual queda confeccionado, modificado o eliminado.

Tabla 14: Resumen del CUS Gestionar balance de alimentos.

Caso de Uso:	Gestionar balance de insumos.
Actores:	Económico de la unidad presupuestada (inicia).
Resumen:	El caso de uso comienza cuando el económico de la unidad presupuestada accede a crear, modificar o eliminar un balance de insumos. El CUS finaliza cuando el balance mensual queda confeccionado, modificado o eliminado.

Tabla 15: Resumen del CUS Gestionar balance de insumos.

Caso de Uso:	Realizar balance nacional.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a realizar el balance. El CUS finaliza cuando el balance mensual queda confeccionado.

Tabla 16: Resumen del CUS Realizar balance nacional.

Caso de Uso:	Gestionar categoría de alimentos.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a la opción categorías de alimentos para crear, modificar o eliminar una categoría. El CUS finaliza cuando el actor crea, modifica o elimina una categoría.

Tabla 17: Resumen del CUS Gestionar categoría de alimentos.

Caso de Uso:	Generar reportes.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso se inicia cuando el subdirector de aseguramiento de la OAMS solicita obtener un reporte dada una fecha. El sistema permite obtener el reporte correspondiente en formato *pdf* e imprimirlo, finalizando el caso de uso.

Tabla 18: Resumen del CUS Generar reportes.

Caso de Uso:	Autenticar usuario.
Actores:	Usuario (inicia).
Resumen:	El caso de uso comienza cuando un usuario accede a ejecutar la aplicación. El CUS finaliza cuando el usuario logra acceder a las funcionalidades del sistema a las que tiene

	permiso.

Tabla 19: Resumen del CUS Autenticar usuario.

Caso de Uso:	Cambiar contraseña.
Actores:	Usuario (inicia).
Resumen:	El caso de uso comienza cuando el usuario accede la opción cambiar contraseña. El CUS finaliza cuando la contraseña es modificada.

Tabla 20: Resumen del CUS Cambiar contraseña.

Caso de Uso:	Gestionar roles.
Actores:	Administrador (inicia).
Resumen:	El caso de uso comienza cuando el administrador accede la opción roles, para crear, modificar o eliminar un rol. El CUS finaliza cuando el rol es creado, modificado o eliminado.

Tabla 21: Resumen del CUS Gestionar roles.

Caso de Uso:	Gestionar usuarios.
Actores:	Administrador (inicia).
Resumen:	El caso de uso comienza cuando el administrador accede la opción usuarios, para crear, modificar o eliminar un usuario. El CUS finaliza cuando el usuario es creado, modificado o eliminado.

Tabla 22: Resumen del CUS Gestionar usuarios.

Caso de Uso:	Gestionar unidades de medida.
Actores:	Subdirector de aseguramiento de la OAMS (inicia).
Resumen:	El caso de uso comienza cuando el subdirector de aseguramiento de la OAMS accede a crear, modificar o eliminar una unidad de medida. El CUS finaliza cuando se ejecuta alguna de estas acciones.

Tabla 23: Resumen del CUS Gestionar unidades de medida.

Caso de Uso:	Importar y exportar datos.
Actores:	Usuario auxiliar (inicia).
Resumen:	El caso de uso comienza cuando el usuario auxiliar accede a importar o exportar datos. El CUS finaliza cuando se ejecuta alguna de estas acciones.

Tabla 24: Resumen del CUS Importar y exportar datos.

Las descripciones textuales de los CUS se encuentran en: **¡Error! No se encuentra el origen de la referencia.**.

2.5 Conclusiones

En este capítulo se llevaron a cabo acciones que facilitarán el desarrollo de una aplicación, que satisfaga las necesidades y exigencias del cliente. Se realizó la modelación del negocio, el levantamiento de requisitos y la modelación del sistema, incluyendo un conjunto de diagramas que permiten una mayor visibilidad de la solución propuesta.

Capítulo 3 Análisis y diseño del sistema

3.1 Introducción

En este capítulo se realiza una breve descripción de las clases que se definen en el modelo de análisis y en el modelo de diseño. También se describen los artefactos que se generan en cada uno de estos modelos, entre los que se encuentran los diagramas de clases del análisis, que describen gráficamente las especificaciones de las clases de *software* y de las interfaces de la aplicación, los diagramas de interacción, que reflejan el comportamiento y la trazabilidad de los mensajes entre los objetos que se definen en el modelo conceptual, a través de diagramas de colaboración.

Una vez realizado el modelo de análisis, se procede a realizar el modelo de diseño, que tendrá como artefactos principales los diagramas de clases del diseño de cada caso de uso. También se hará referencia al estilo arquitectónico y los patrones de diseño que se utilizarán. Además, se mostrará el diagrama de despliegue, el que representa la relación física de los nodos donde se implantará el sistema.

3.2 Modelo del análisis

El modelo de análisis ayuda a refinar los requisitos, representa la estructura global del sistema, describe la realización de casos de uso, sirve como una abstracción del modelo de diseño, se centra en los requerimientos funcionales y permite razonar sobre los aspectos internos del sistema, incluidos sus recursos compartidos internos. Se centra además en aspectos tales como la flexibilidad ante los cambios y la reutilización. Esta estructura se utiliza como entrada en las actividades del diseño. Este modelo consiste en obtener una visión del sistema que se preocupa de ver qué se debe hacer, sin tener que preocuparse por cuestiones propias del lenguaje.

3.2.1 Diagrama de clases del análisis

RUP propone clasificar a las clases del análisis en:

- Interfaz: modelan la interacción entre el sistema y sus actores.

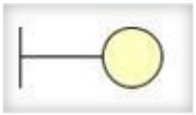


Figura 6: Estereotipo de la clase interfaz del análisis.

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

Figura 7: Estereotipo de la clase controladora del análisis.

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

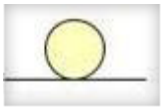


Figura 8: Estereotipo de la clase entidad del análisis.

3.3 Diagrama de clases

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

A continuación se muestran los diagramas de clases del análisis correspondientes a los CUS:

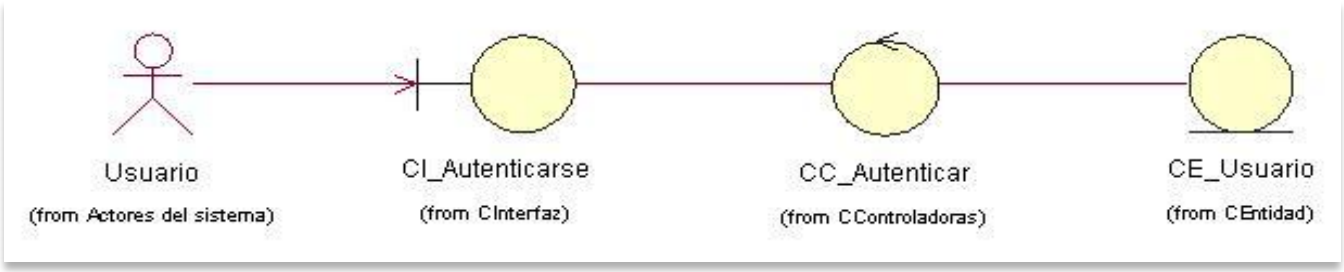


Figura 9: Diagrama de clases del análisis del CU Autenticar usuario.

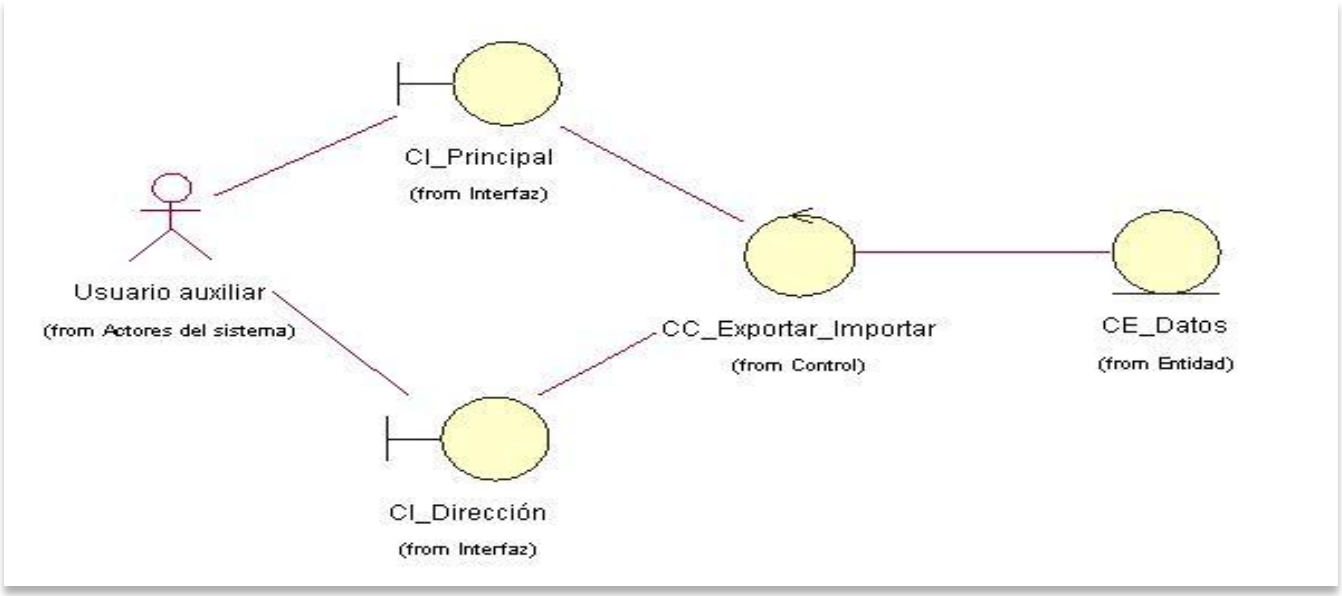


Figura 10: Diagrama de clases del análisis del CU Exportar/Importar datos.

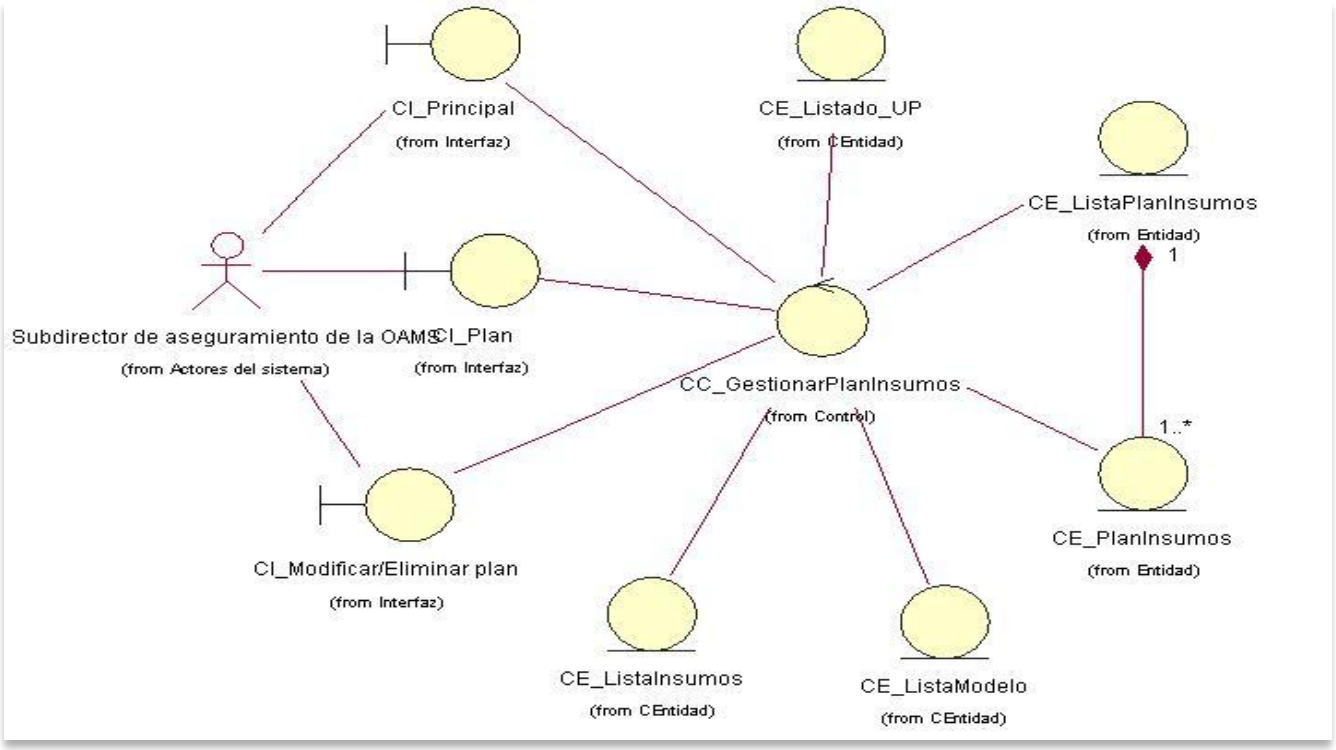


Figura 11: Diagrama de clases del análisis del CU Gestionar plan de insumos.

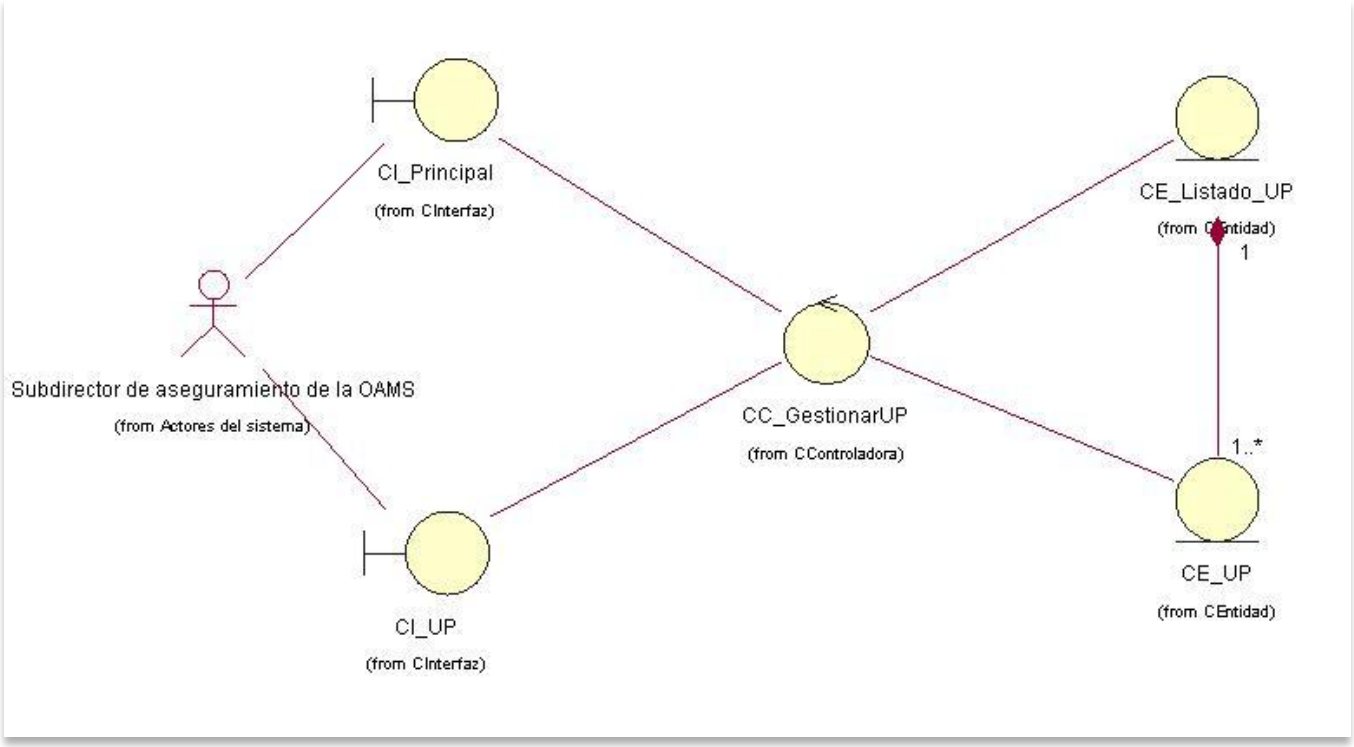


Figura 12: Diagrama de clases del análisis del CU Gestionar UP.

Los restantes diagramas de clases del análisis se encuentran en: Anexo 3: Diagramas de clases del análisis.

3.3.1 Diagrama de interacción del análisis (colaboración)

Los diagramas de interacción incluyen la interacción de los mensajes entre los objetos que se definen en el modelo conceptual y otras clases de objetos.

Un diagrama de colaboración específicamente muestra una interacción organizada de los objetos que efectúan operaciones.

A continuación se muestran los diagramas de colaboración correspondientes a los CUS:

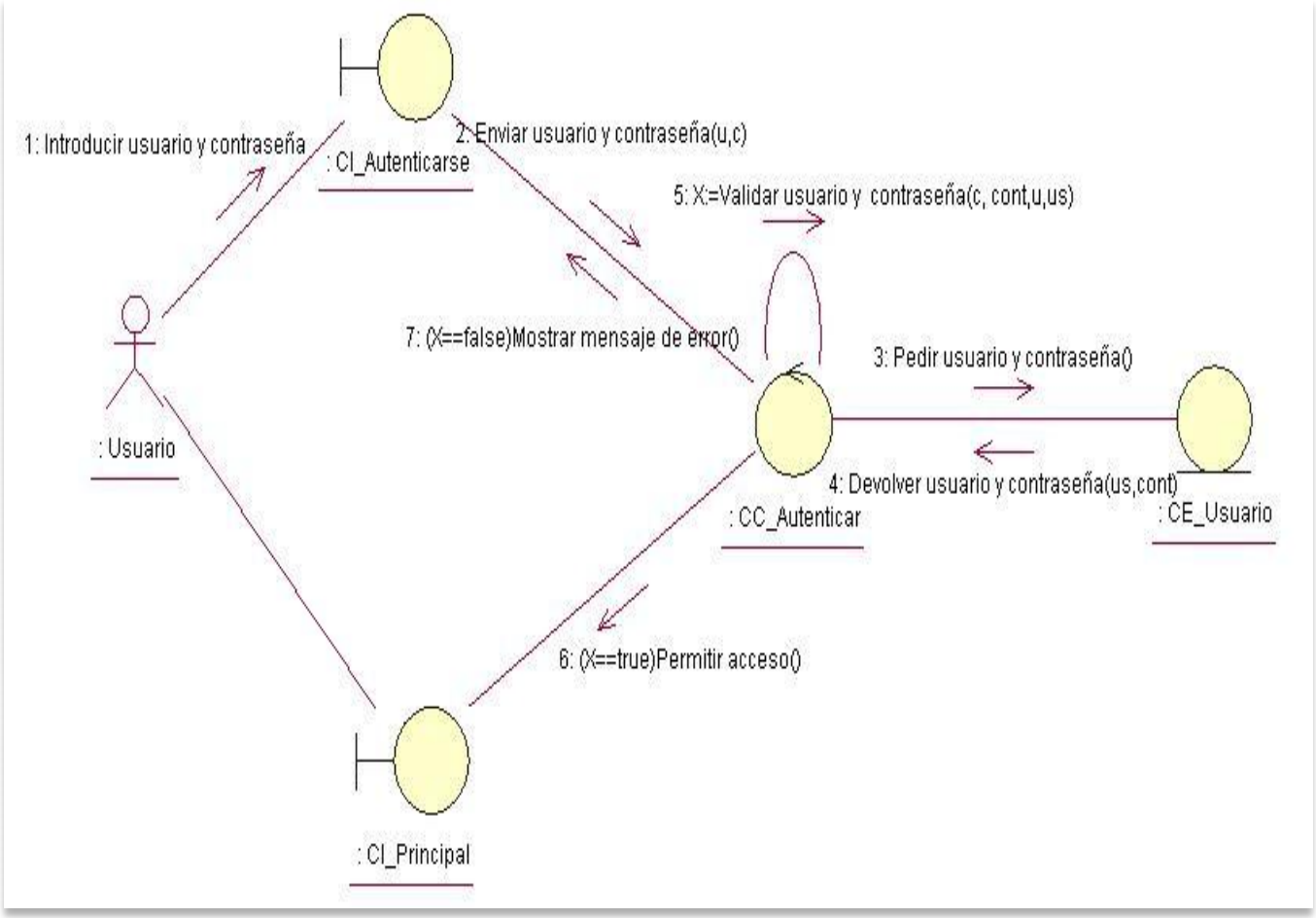


Figura 13: Diagrama de colaboración del CU Autenticar usuario.

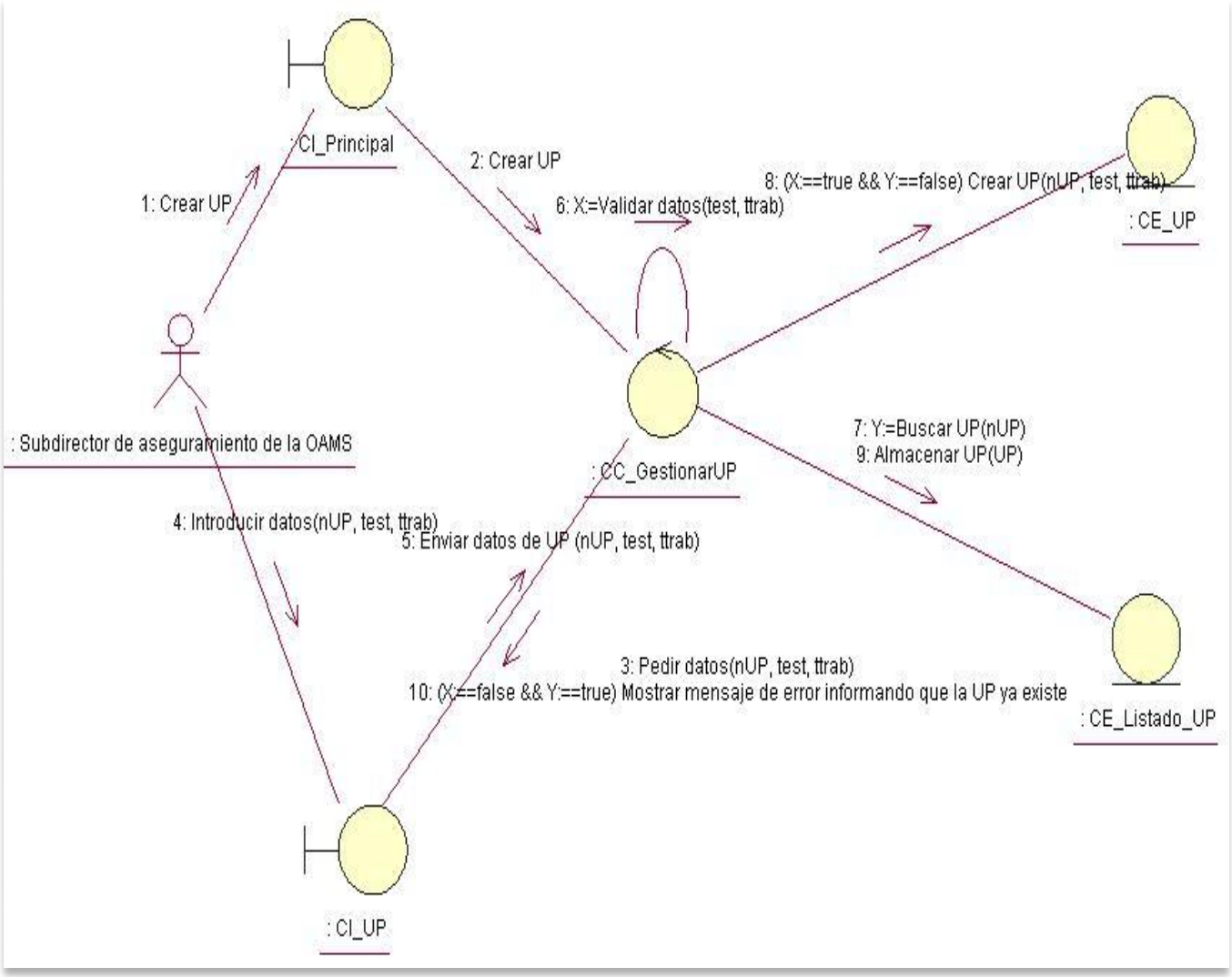


Figura 14: Diagrama de colaboración del CU Gestionar UP para el escenario crear UP.

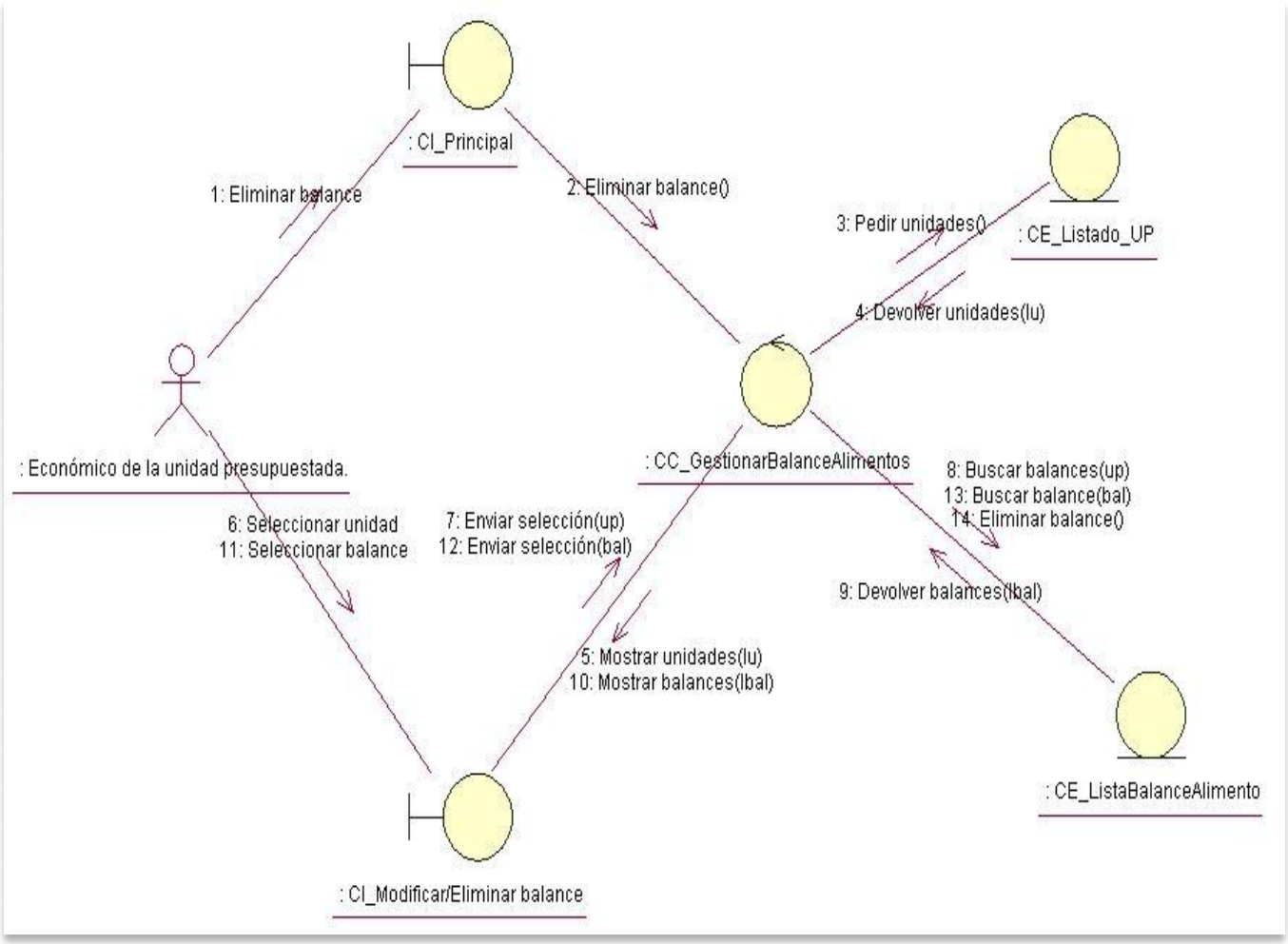


Figura 15: Diagrama de colaboración del CU Gestionar balance de alimentos para el escenario eliminar balance.

Los restantes diagramas de colaboración se encuentran en: **¡Error! No se encuentra el origen de la referencia.**.

3.4. Patrones de diseño a utilizar

"En la tecnología orientada a objetos, un patrón es una descripción del problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos". (7)

Para el diseño de la aplicación se hizo uso de los Patrones de Principios Generales para Asignar Responsabilidades (GRASP). Los patrones de GRASP que se utilizaron fueron: experto, creador, alta cohesión y bajo acoplamiento.

- Experto:

• Problema: ¿qué principio usar para asignar responsabilidades a los objetos?

- Solución: asignar una responsabilidad al experto en información (la clase que tiene la información necesaria para realizar la responsabilidad).

Si esta actividad se realiza bien, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y existen más oportunidades para reutilizar componentes en futuras aplicaciones.

El experto en información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos.

"El experto expresa la "intuición" común de que los objetos hacen las cosas relacionadas con la información que tienen. Se debe notar que el cumplimiento de la responsabilidad a menudo requiere información que se encuentra dispersa por diferentes clases de objetos". (7)

En el sistema se utilizó este patrón en el diseño de las clases que representan los objetos del dominio, que son las que tienen los atributos que describen dichos objetos, y a través de ellas se realizan operaciones sobre los mismos.

Beneficios:

1. Se mantiene el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas. Normalmente, esto conlleva un bajo acoplamiento, lo que da lugar a sistemas más robustos y más fáciles de mantener.
2. Se distribuye el comportamiento entre las clases que contienen la información requerida, por tanto, se estimula las definiciones de clases más cohesivas y "ligeras" que son más fáciles de entender y mantener. Se soporta normalmente una alta cohesión.

➢ Creador

- Problema: ¿quién debería ser el responsable de la creación de una nueva instancia de alguna clase?
- Solución: asignar a la clase B la responsabilidad de crear una instancia de la clase A si se cumple una o más de los casos siguientes:
    - B agrega objetos de A.
    - B contiene objetos de A.
    - B registra instancias de objetos de A.
    - B utiliza más estrechamente objetos de A.
    - B tiene los datos de inicialización que se pasará a un objeto de A cuando sea creado (por tanto, B es un experto con respecto a la creación de A).

Si se asigna bien la responsabilidad de creación, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización.

Este patrón se utilizó para diseñar las clases que controlan todos los procesos del negocio, que a la vez se encargan de instanciar las clases que representan los objetos del dominio y darles sus datos de inicialización.

- Alta cohesión:
- Problema: ¿cómo mantener la complejidad manejable?
- Solución: asignar una responsabilidad de manera que la cohesión permanezca alta.

La cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. Estos elementos pueden ser clases, subsistemas, entre otros.

Una clase con baja cohesión hace muchas cosas no relacionadas, o hace demasiado trabajo. Tales clases no son convenientes, adolecen de los siguientes problemas:

1. Difíciles de entender.
2. Difíciles de reutilizar.
3. Difíciles de mantener.
4. Delicadas, constantemente afectadas por los cambios.

"Como regla empírica, una clase con alta cohesión tiene un número relativamente pequeño de métodos, con funcionalidad altamente relacionada, y no realiza mucho trabajo. Colabora con otros objetos para compartir el esfuerzo si la tarea es extensa". (7)

Para diseñar las clases de acceso a datos y las controladoras, fue necesario el uso de este patrón para lograr tener en las mismas la menor cantidad de operaciones posibles, y que estas operaciones estuvieran relacionadas, por ejemplo en las clase controladoras, la mayoría de las funcionalidades se centran en gestionar los objetos del dominio, y en las de acceso a datos, en realizar las consultas necesarias para tratar los datos de estos objetos.

- Bajo acoplamiento:
  - Problema: ¿cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?
  - Solución: asignar una responsabilidad de manera que el acoplamiento permanezca bajo.

El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos. Un elemento con bajo acoplamiento no depende de demasiados otros elementos.

Una clase con alto acoplamiento confía en muchas otras clases. Tales clases podrían no ser deseables; algunas adolecen de los siguientes problemas:

1. Los cambios en las clases relacionadas fuerzan cambios locales.
2. Son difíciles de entender de manera aislada.
3. Son difíciles de reutilizar puesto que su uso requiere la presencia adicional de las clases de las que depende.

En general, las clases que son inherentemente muy genéricas por naturaleza, y con una probabilidad de reutilización alta, debería tener un acoplamiento especialmente bajo.

"El caso extremo de bajo acoplamiento es cuando no existe acoplamiento entre clases. Esto no es deseable porque una metáfora central de la tecnología de objetos es un sistema de objetos conectados que se comunican mediante el paso de mensajes. Si el bajo acoplamiento se lleva al extremo, producirá un diseño pobre porque dará lugar a unos pocos objetos inconexos, saturados, y con actividad compleja que hacen todo el trabajo, con muchos objetos muy pasivos, sin acoplamiento que actúan como simple repositorio de datos". (7)

Para el diseño de la clase conexión, se aplicó este patrón con el objetivo de lograr que esta clase no dependiera de ninguna otra, y que pudiera ser utilizada en la implementación de todos los casos de uso.

3.5 Arquitectura del sistema

"La arquitectura de una aplicación es la vista conceptual de la estructura de ésta. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como está distribuido este código". (8)

La arquitectura a usar en el diseño del sistema es la arquitectura de n_capas (3 capas). Esta arquitectura "se enfoca en la distribución de roles y responsabilidades de forma jerárquica, brindando una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada". (8)

La arquitectura en capas se encuentra centrada básicamente en dividir un problema en pequeñas partes que puedan ser manejadas e implementadas de forma independiente, dichas partes poseerán responsabilidades específicas que no dependan del funcionamiento de las otras, o al menos que su dependencia sea mínima. Este aspecto constituye una ventaja considerable, pues proporciona una amplia reutilización de las clases implementadas, al hacer abstracciones de las distintas funcionalidades o responsabilidades del sistema agrupándolas en capas.

Los sistemas o arquitecturas en capas, constituyen una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior, y se sirve de las prestaciones que le brinda la inmediatamente inferior.

Las tres capas son:

- Capa de presentación: esta capa resuelve la presentación de datos al usuario, o sea, representa la interacción entre el usuario y el sistema. En esta capa están contenidas todas las interfaces a través de las cuales el usuario puede decirle al sistema que almacene determinados datos, o pedirle datos ya almacenados y que él necesita.
- Capa de lógica del negocio: esta capa resuelve la lógica de la aplicación. Contiene los algoritmos, validaciones y coordinación necesaria para resolver la problemática. Los elementos fundamentales de esta capa son los objetos de dominio. Estos objetos se representan como los objetos principales del negocio y contienen la lógica del mismo. Esta capa contiene las soluciones de los problemas planteados en el estudio del negocio.
- Capa de acceso a datos: esta capa resuelve el acceso a datos, abstrayendo a su capa superior de la complejidad del acceso e interacción con los diferentes orígenes de datos. Es la encargada de darle a la capa de negocio lo que el usuario necesita o sea, que en esta capa existe una lógica que administra la comunicación entra la base de datos y la capa del negocio.

Algunos de los principios que se aplican cuando se diseña para usar este estilo de arquitectura incluyen:

- Funcionalidad claramente definida: el diseño claramente define la separación entre la funcionalidad de cada capa. Capas superiores como la capa de presentación, envían comandos a las capas inferiores como la capa de negocios y la capa de datos, y los datos fluyen hacia y desde las capas en cualquier sentido.
- Alta cohesión: cada capa contiene funcionalidad directamente relacionada con la tarea de dicha capa.
- Reutilizable: las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios.

3.6 Modelo de diseño

El diseño ocupa el final de la fase de elaboración y el comienzo de las iteraciones de construcción. Contribuye a una arquitectura estable y sólida. En el diseño se modela el sistema teniendo en cuenta la arquitectura, para lograr que soporte todos los requisitos, funcionales no funcionales y las restricciones que se le suponen.

3.6.1 Diagrama de clases del diseño

Los diagramas de clases del diseño son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones. Son el pilar básico del modelado con UML, siendo utilizados para mostrar cómo puede ser construido el sistema.

A continuación se muestran los diagramas de clases del diseño correspondientes a los CUS:

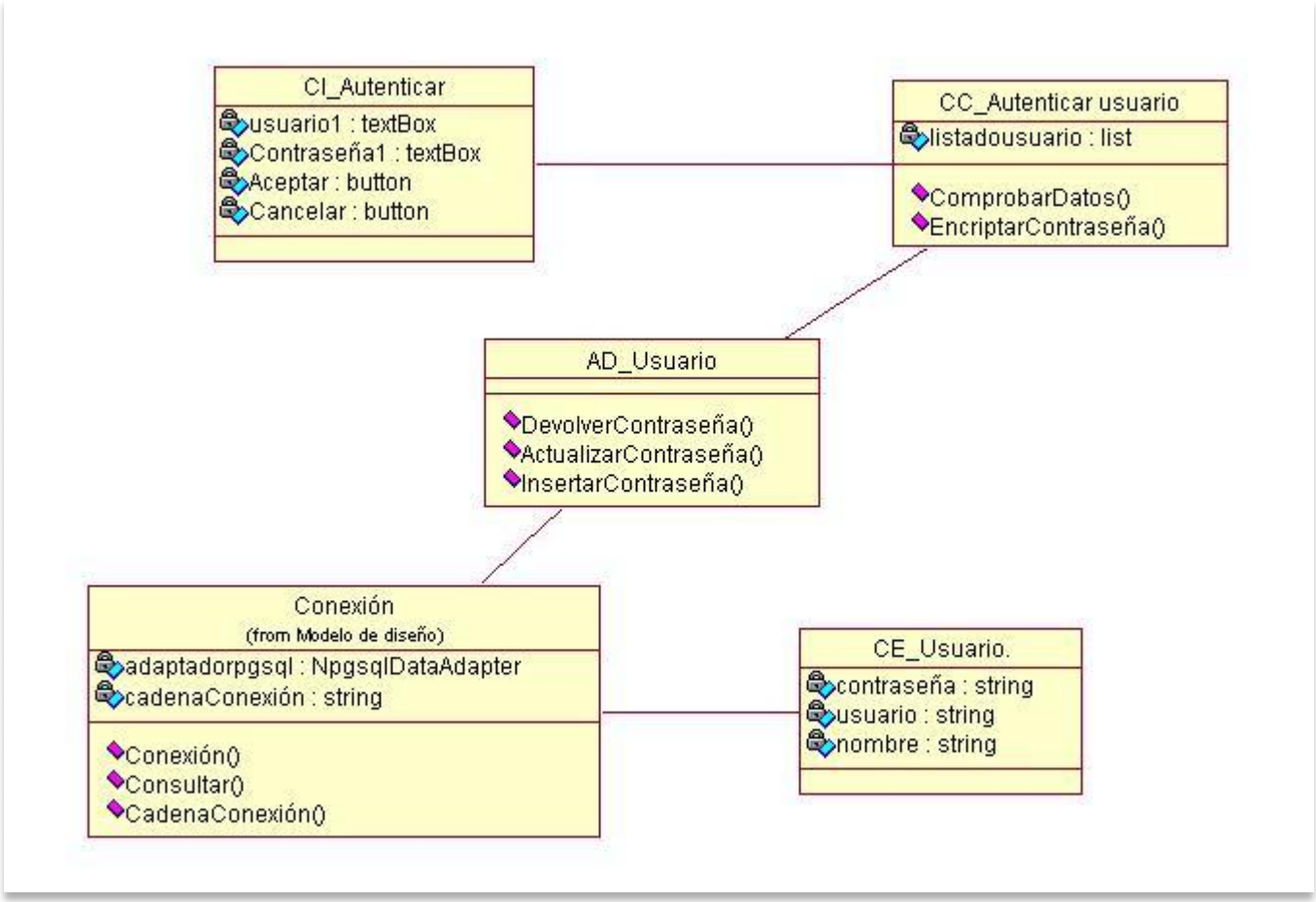


Figura 16: Diagrama de clases del diseño para el CU Autenticar usuario.

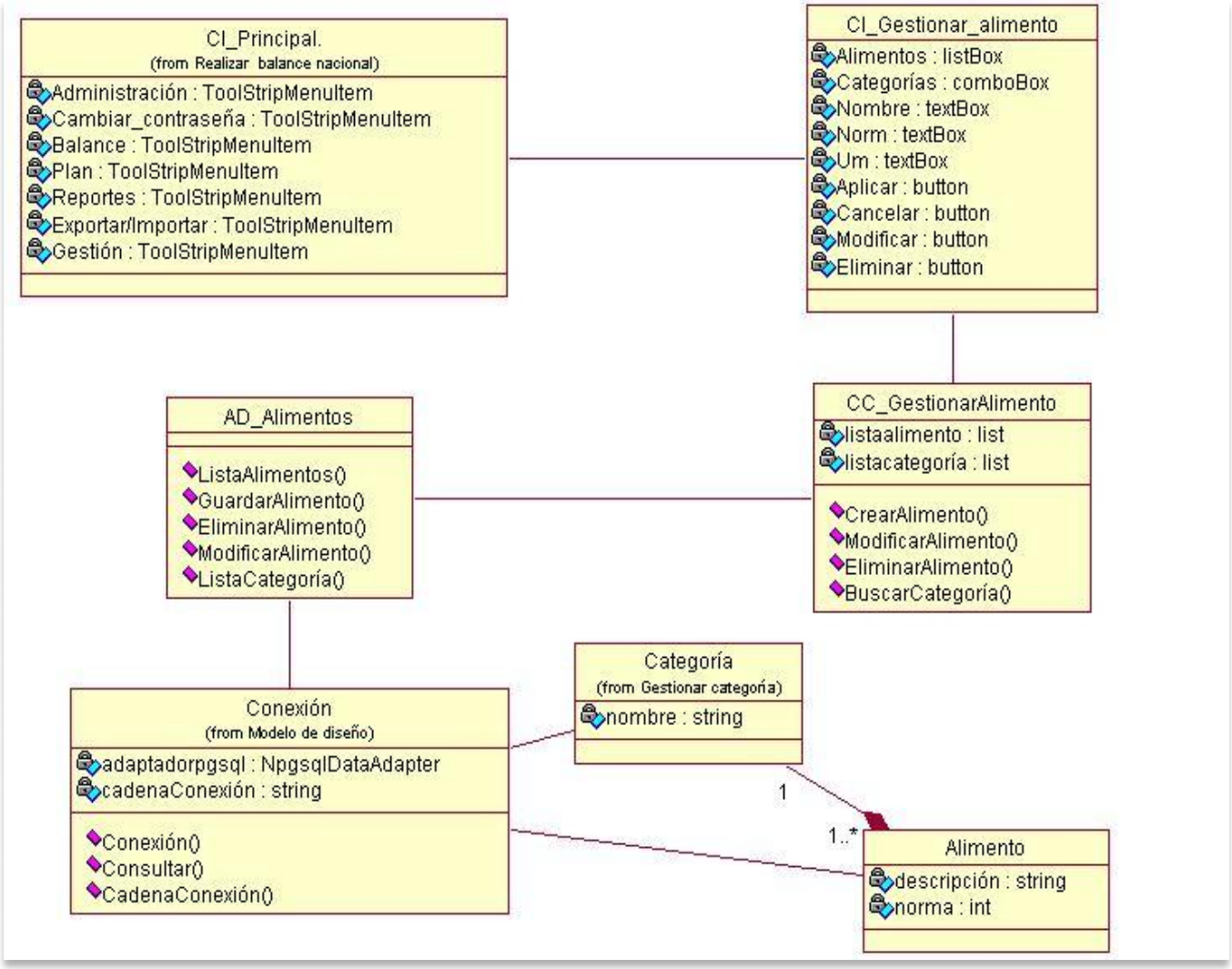


Figura 17: Diagrama de clases del diseño para el CU Gestionar alimento.

Los restantes diagramas de clases del diseño se encuentran en: **¡Error! No se encuentra el origen de la referencia.**.

3.6.1.1 Descripción de las clases del diseño

Descripción de las clases controladoras (CC):

Nombre:	CC_Autenticar_usuario	
Tipo de clase:	Controladora	
Atributo:		**Tipo:**
usuario		Usuario
Para cada responsabilidad:		
Nombre:	EncriptarContraseña()	
Descripción:	Este método permite encriptar la primera contraseña que se le va a poner al sistema.	

Nombre:	ComprobarDatos()
Descripción:	Este método permite comprobar si los datos de autenticación de un usuario son correctos.

Tabla 25: Descripción de la CC_Autenticar_usuario.

Nombre:	CC_Gestionar_alimento
Tipo de clase:	Controladora
Atributo:	**Tipo:**
listaalimento	list
listacategoría	list
Para cada responsabilidad:	
Nombre:	Insertar_alimento(Alimentos ali)
Descripción:	A través de este método se introducen todos los datos de un alimento en el sistema.
Nombre:	Modificar_alimento(Alimentos ali)
Descripción:	A través de este método se modifican los datos que el usuario desee de un alimento en el sistema.
Nombre:	Eliminar_alimento(int id)
Descripción:	A través de este método se elimina del sistema el alimento que el usuario desee.
Nombre:	BuscarCategoría()
Descripción:	Este método permite buscar la categoría que el usuario seleccione.

Tabla 26: Descripción de la CC_Gestionar_alimento.

Nombre:	CC_Gestionar_categoría
Tipo de clase:	Controladora
Atributo:	**Tipo:**
lista_categoria	list
Para cada responsabilidad:	
Nombre:	Insertar_categoría(Categoría cat)
Descripción:	A través de este método se introducen todos los datos de una categoría en el sistema.
Nombre:	Modificar_categoría(Categoría cat)
Descripción:	A través de este método se modifican los datos que el usuario desee de una categoría en el sistema.
Nombre:	Eliminar_categoría(int id)
Descripción:	A través de este método se elimina del sistema la categoría que el usuario desee.

Tabla 27: Descripción de la CC_Gestionar_categoría.

Nombre:	CC_Gestionar_modelo
Tipo de clase:	Controladora
Atributo:	**Tipo:**

lista_modelo	list
Para cada responsabilidad:	
Nombre:	Insertar_modelo(Modelo mod)
Descripción:	A través de este método se introducen todos los datos de un modelo en el sistema.
Nombre:	Modificar_modelo(Modelo mod)
Descripción:	A través de este método se modifican los datos que el usuario desee de un modelo en el sistema.
Nombre:	Eliminar_modelo(int idm)
Descripción:	A través de este método se elimina del sistema el modelo que el usuario desee.

Tabla 28: Descripción de la CC_Gestionar_modelo.

Nombre:	CC_Gestionar_producto
Tipo de clase:	Controladora
Atributo:	**Tipo:**
lista_modelo	list
lista_producto	list
Para cada responsabilidad:	
Nombre:	Insertar_producto(Producto pord)
Descripción:	A través de este método se introducen todos los datos de un producto en el sistema.
Nombre:	Modificar_producto(Producto pord)
Descripción:	A través de este método se modifican los datos que el usuario desee de un producto en el sistema.
Nombre:	Eliminar_producto(int idp)
Descripción:	A través de este método se elimina del sistema el producto que el usuario desee.
Nombre:	Buscar_modelo()
Descripción:	A través de este método el usuario puede obtener el modelo deseado.

Tabla 29: Descripción de la CC_Gestionar_producto.

Nombre:	CC_Gestionar_unidad_presupuestada
Tipo de clase:	Controladora
Atributo:	**Tipo:**
lista_unidades	list
Para cada responsabilidad:	
Nombre:	Insertar_unidad(Unidad_presupuestada uni)
Descripción:	A través de este método se introducen todos los datos de una UP en el sistema.
Nombre:	Modificar_unidad(Unidad_presupuestada uni)
Descripción:	A través de este método se modifican los datos que el usuario desee de una UP en el

	sistema.
Nombre:	Eliminar_unidad(int idu)
Descripción:	A través de este método se elimina del sistema la UP que el usuario desee.

Tabla 30: Descripción de la CC_Gestionar_unidad_presupuestada.

Nombre:	CC_Cambiar_contraseña	
Tipo de clase:	Controladora	
Atributo:		**Tipo:**
usuario		Usuario
Para cada responsabilidad:		
Nombre:	BuscarUsuario()	
Descripción:	Este método permite obtener el usuario que está usando la sección de trabajo.	
Nombre:	ComprobarContraseña()	
Descripción:	Este método permite comprobar que la contraseña sea la correcta.	
Nombre:	CambiarContraseña()	
Descripción:	Este método permite modificar la contraseña.	

Tabla 31: Descripción de la CC_Cambiar_contraseña.

Nombre:	CC_Realizar_balance_nacional	
Tipo de clase:	Controladora	
Atributo:		**Tipo:**
listadobalancealimento		list
listadobalanceinsumo		list
listadobalancenacional		list
Para cada responsabilidad:		
Nombre:	BuscarBalances()	
Descripción:	Este método permite obtener los balances de alimentos e insumos.	
Nombre:	SumarImportes()	
Descripción:	Este método permite sumar los importes de los alimentos e insumos.	
Nombre:	SumarCostos()	
Descripción:	Este método permite sumar los costos de los alimentos e insumos.	
Nombre:	CrearBalanceNacional()	
Descripción:	Este método permite crear un balance nacional.	

Tabla 32: Descripción de la CC_Realizar_balance_nacional.

Nombre:	CC_Generar_reporte	
Tipo de clase:	Controladora	
Atributo:		**Tipo:**

istaplaninsumo	list
listaplanalimento	list
listabalancealimento	list
listabalanceinsumo	list
listabalancenacional	list
Para cada responsabilidad:	
Nombre:	BuscarBalanceInsumo()
Descripción:	Este método permite obtener los balances de insumos.
Nombre:	BuscarBalanceAlimento()
Descripción:	Este método permite obtener los balances de alimento.
Nombre:	BuscarPlanInsumo()
Descripción:	Este método permite obtener los planes de insumos.
Nombre:	BuscarPlanAlimento()
Descripción:	Este método permite obtener los planes de alimentos.
Nombre:	BuscarBalanceNacional()
Descripción:	Este método permite obtener los balances nacionales.

Tabla 33: Descripción de la CC_Generar_reporte.

Nombre:	CC_Gestionar_balance_alimentos
Tipo de clase:	Controladora
Atributo	**Tipo**
listaplanalimento	list
listaup	list
listabalancealimento	list
Para cada responsabilidad:	
Nombre:	CrearBalanceAlimento()
Descripción:	Este método permite insertar los datos de un balance de alimentos.
Nombre:	ModificarBalanceAlimento()
Descripción:	Este método permite modificar los datos de un balance de alimentos.
Nombre:	EliminarBalanceAlimento()
Descripción:	Este método permite eliminar un balance de alimentos.
Nombre:	BuscarPlan()
Descripción:	Este método permite obtener los planes de una UP.
Nombre:	BuscarUP()
Descripción:	Este método permite obtener una UP.

Tabla 34: Descripción de la CC_Gestionar_balance_alimentos.

Nombre:	CC_Gestionar_plan_alimentos	
Tipo de clase:	Controladora	
Atributo		**Tipo**
listaplanalimento		list
listadocategoría		list
listaalimento		list
Para cada responsabilidad:		
Nombre:	CrearPlanAlimento()	
Descripción:	Este método permite insertar los datos de un plan de alimentos.	
Nombre:	ModificarPlanAlimento()	
Descripción:	Este método permite modificar los datos de un plan de alimentos.	
Nombre:	EliminarPlanAlimento()	
Descripción:	Este método permite eliminar un plan de alimentos.	
Nombre:	BuscarCategoría()	
Descripción:	Este método permite obtener la categoría deseada.	
Nombre:	BuscarAlimento()	
Descripción:	Este método permite obtener los alimentos de una categoría dada.	
Nombre:	CalcularDemanda()	
Descripción:	Este método permite calcular la demanda de un alimento para una frecuencia dada.	
Nombre:	BuscarUP()	
Descripción:	Este método permite obtener la UP deseada.	

Tabla 35: Descripción de la CC_Gestionar_plan_alimentos.

Nombre:	CC_Gestionar_plan_insumos	
Tipo de clase:	Controladora	
Atributo:		**Tipo:**
ListaModelos		list
ListaInsumos		list
ListaPlanInsumos		list
ListaUP		list
Para cada responsabilidad:		
Nombre:	CrearPlanInsumo()	
Descripción:	Este método permite insertar los datos de un plan de insumos.	
Nombre:	ModificarPlanInsumo()	
Descripción:	Este método permite modificar los datos de un plan de insumos.	
Nombre:	EliminarPlanInsumo()	

Descripción:	Este método permite eliminar un plan de insumos.
Nombre:	BuscarModelos()
Descripción:	Este método permite obtener el modelo deseado.
Nombre:	BuscarInsumos()
Descripción:	Este método permite obtener los insumos de un modelo dado.
Nombre:	BuscarUP()
Descripción:	Este método permite obtener la UP deseada.

Tabla 36: Descripción de la CC_Gestionar_plan_insumos.

Nombre:	CC_Gestionar_balance_insumos
Tipo de clase:	Controladora
Atributo:	**Tipo:**
listaplaninsumo	list
listabalanceinsumo	list
listaup	list
Para cada responsabilidad:	
Nombre:	CrearBalanceInsumo()
Descripción:	Este método permite insertar los datos de un balance de insumos.
Nombre:	ModificarBalanceInsumo()
Descripción:	Este método permite modificar los datos de un balance de insumos.
Nombre:	EliminarBalanceinsumo()
Descripción:	Este método permite eliminar un balance de insumos.
Nombre:	BuscarUP()
Descripción:	Este método permite obtener la UP deseada.
Nombre:	BuscarPlan()
Descripción:	Este método permite obtener los planes de insumos de una UP.

Tabla 37: Descripción de la CC_Gestionar_balance_insumos.

Nombre:	CC_Gestionar_rol
Tipo de clase:	Controladora
Atributo:	**Tipo:**
listaroles	list
listadopermisos	list
Para cada responsabilidad:	
Nombre:	CrearRol()
Descripción:	Este método permite insertar los datos de un rol.
Nombre:	ModificarRol()

Descripción:	Este método permite modificar los datos de un rol.
Nombre:	EliminarRol()
Descripción:	Este método permite eliminar los datos de un rol.

Tabla 38: Descripción de la CC_Gestionar_rol.

Nombre:	CC_Gestionar_unidad_medida	
Tipo de clase:	Controladora	
Atributo		**Tipo**
listaum		list
Para cada responsabilidad:		
Nombre:	CrearUM()	
Descripción:	Este método permite insertar los datos de una unidad de medida.	
Nombre:	ModificarUM()	
Descripción:	Este método permite modificar los datos de una unidad de medida.	
Nombre:	EliminarUM()	
Descripción:	Este método permite eliminar los datos de una unidad de medida.	

Tabla 39: Descripción de la CC_Gestionar_ unidad_medida.

Nombre:	CC_Gestionar_usuario	
Tipo de clase:	Controladora	
Atributo		**Tipo**
listausuario		list
listaroles		list
Para cada responsabilidad:		
Nombre:	CrearUsuario()	
Descripción:	Este método permite insertar los datos de un usuario.	
Nombre:	ModificarUsuario()	
Descripción:	Este método permite modificar los datos de un usuario.	
Nombre:	EliminarUsuario()	
Descripción:	Este método permite eliminar los datos de un usuario.	
Nombre:	BuscarRoles()	
Descripción:	Este método permite obtener los roles existentes.	

Tabla 40: Descripción de la CC_Gestionar_usuario.

Descripción de las clases de tipo entidad (CE):

Nombre:	CE_Usuario	
Tipo de clase:	Entidad	
Atributo:		**Tipo:**

contraseña	string
usuario	string
nombre	string

Tabla 41: Descripción de la CE_Usuario.

Nombre:	CE_Alimento
Tipo de clase:	Entidad
Atributo:	**Tipo:**
descripción	string
norma	string
unidad_medida	string

Tabla 42: Descripción de la CE_Alimento.

Nombre:	CE_Categoría
Tipo de clase:	Entidad
Atributo:	**Tipo:**
nombre	string

Tabla 43: Descripción de la CE_Categoría.

Nombre:	CE_Modelo
Tipo de clase:	Entidad
Atributo:	**Tipo:**
descripción	string
período	string

Tabla 44: Descripción de la CE_Modelo.

Nombre:	CE_Producto
Tipo de clase:	Entidad
Atributo:	**Tipo:**
unidad_medida	string
descripción	string

Tabla 45: Descripción de la CE_Producto.

Nombre:	CE_Unidad_presupuestada
Tipo de clase:	Entidad
Atributo:	**Tipo:**
nombre_up	string
cant_estudiantes	int
cant_trabajadores	int

Tabla 46: Descripción de la CE_Unidad_preupuestada.

Nombre:	CE_Balance_producto
Tipo de clase:	Entidad
Atributo:	**Tipo:**
existencia_inicial	double
fecha	datetime
ef	double
ed	double
et	double
es	double
sc	double
st	double
sm	double
sf	double
totalentradas	double
totalsalidas	double
preciomn	double
preciocuc	double
importemn	double
importecuc	double
exist_final	double

Tabla 47: Descripción de la CE_Balance_producto.

Nombre:	CE_Balance_alimento
Tipo de clase:	Entidad
Atributo:	**Tipo:**
existencia_inicial	double
fecha	datetime
ef	double
ed	double
et	double
es	double
sc	double
st	double
sm	double
sf	double
totalentradas	double

totalsalidas	double
preciomn	double
preciocuc	double
importemn	double
importecuc	double
cobertura	double
exist_final	double

Tabla 48: Descripción de la CE_Balance_alimento.

Nombre:	CE_Balance_nacional
Tipo de clase:	Entidad
Atributo:	**Tipo:**
consumocuc	double
consumomn	double
fecha	datetime

Tabla 49: Descripción de la CE_Balance_nacional.

Nombre:	CE_Plan_producto
Tipo de clase:	Entidad
Atributo:	**Tipo:**
fecha	datetime
cant_prod	int

Tabla 50: Descripción de la CE_Plan_producto.

Nombre:	CE_Plan_alimento
Tipo de clase:	Entidad
Atributo:	**Tipo:**
fecha	datetime
de	double
dt	double
dtotal	double
pr	double
fe	double
ft	double

Tabla 51: Descripción de la CE_Plan_alimento.

Nombre:	CE_Rol
Tipo de clase:	Entidad
Atributo	**Tipo**

nombre	string
permiso	string

Tabla 52: Descripción de la CE_Rol.

Nombre:	CE_UM
Tipo de clase:	Entidad
Atributo	**Tipo**
nombre	string
abreviatura	string

Tabla 53: Descripción de la CE_UM.

3.7 Diagrama de despliegue

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

A continuación se muestra la vista de despliegue para la aplicación diseñada:

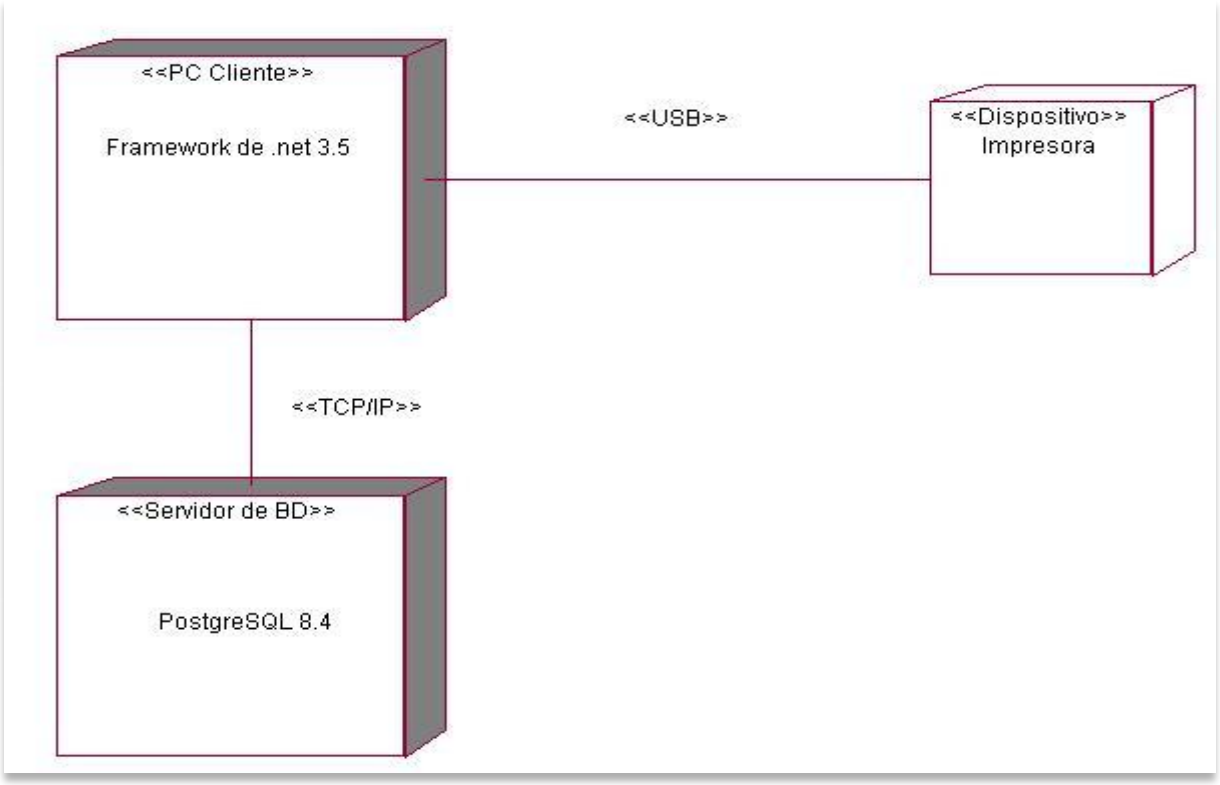


Figura 16: Diagrama de despliegue.

3.8 Diseño de la base de datos

Modelo de datos: es un lenguaje utilizado para la descripción de una base de datos. Permite describir las estructuras de datos de la base, las restricciones de integridad y las operaciones de manipulación de los datos, o sea, permite manipular los datos de un cierto mundo real que se desea almacenar en la base de datos.

El modelo entidad relación se encuentra en: Anexo 6: Modelo de datos.

El diagrama de clases persistentes se encuentra en: Anexo 7: Diagrama de clases persistentes.

3.9 Conclusiones

Con la realización del modelo de análisis y diseño, se transforman los requerimientos funcionales en vista a la futura implementación del sistema, y se establecen las bases para una arquitectura de *software* más robusta.

Conclusiones generales

Después de haber diseñado el sistema para la gestión de la información de los almacenes de las unidades presupuestadas que se subordinan a la Oficina de Atención a Misiones de la Salud, se pudo llegar a las siguientes conclusiones:

- Se realizó un análisis del estado del arte de algunos sistemas que tratan la gestión de la información de almacenes, permitiendo tomar de ellos experiencias para el diseño de la propuesta de solución.
- Se realizó un estudio de las herramientas, lenguajes y metodologías informáticas, lo que facilitó el diseño de la solución propuesta y adelantó el trabajo de los futuros desarrolladores.
- Se definieron los requerimientos funcionales, los que permitieron tener una guía de las necesidades del cliente.
- Se propuso una posible solución a la problemática planteada por el cliente en las diferentes entrevistas que se le realizaron.
- Se realizó el análisis y el diseño del sistema, lo que dio la posibilidad de obtener una guía para la futura implementación.

Recomendaciones

Una vez concluido el análisis y diseño del sistema para la gestión de la información de los almacenes de las unidades presupuestadas que se subordinan a la Oficina de Atención a Misiones de la Salud se recomienda:

- Comenzar la implementación del sistema, sobre la propuesta de solución brindada en el trabajo de diploma.
- Realizar el estudio de nuevos procesos que permitan un mayor control de la información para la Oficina de Atención a Misiones de la Salud.
- Comenzar el diseño de una nueva versión de la aplicación sobre plataforma *web,* para que pueda ser utilizado en los lugares donde exista conexión con la red de Infomed, lo que permitirá la agilización de las operaciones.

Referencias bibliográficas

1. **EterovicS, Yadran.** El Proceso Unificado (RUP): Técnicas Modernas para Desarrollar Aplicaciones. *El Proceso Unificado (RUP): Técnicas Modernas para Desarrollar Aplicaciones.* [En línea] [Citado el: 06 de 02 de 2010.] http://www2.ing.puc.cl/~iic3194/rupppt.pdf .

2. **Jiménez Garzón, Darwin.** [En línea] [Citado el: 11 de 01 de 2010.] http://codeticainge.googlepages.com/guiaing.pdf.

3. **Calero Solís, Manuel.** Una explicación de la programación extrema (XP). *Una explicación de la programación extrema (XP).* [En línea] [Citado el: 12 de 01 de 2010.] http://www.willydev.net/descargas/prev/ExplicaXp.pdf.

4. **Barrientos Enríquez, Aleida Mirian.** El desarrollo de sistemas de información empleando el lenguaje de modelado unificado UML. *El desarrollo de sistemas de información empleando el lenguaje de modelado unificado UML.* [En línea] [Citado el: 17 de 01 de 2010.] http://www.monografias.com/trabajos16/lenguaje-modelado-unificado/lenguaje-modelado-unificado.shtml.

5. **Rojas, Jesús.** [En línea] [Citado el: 14 de 02 de 2010.] http://www.slideshare.net/jesus25dite/lenguaje-sql-3259211.

6. **Alvarez, Rubén.** Qué es y para qué sirve el SQL. *Qué es y para qué sirve el SQL.* [En línea] [Citado el: 24 de 01 de 2010.] http://www.desarrolloweb.com/articulos/262.php.

7. **Lescano, German.** Patrones fundamentales en el diseño orientado a objetos (DOO). *Patrones fundamentales en el diseño orientado a objetos (DOO).* [En línea] 10 de 03 de 2010. [Citado el: 26 de 04 de 2010.] http://germanlescano.wordpress.com/2010/03/10/patrones-.

8. **Peláez, Juan Carlos.** Extractado de La Guía de Arquitectura. *Extractado de La Guía de Arquitectura.* [En línea] 29 de 5 de 2009. [Citado el: 25 de 05 de 2010.] http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx.

Bibliografía consultada

1. EterovicS, Yadran. El Proceso Unificado (RUP): Técnicas Modernas para Desarrollar Aplicaciones. *El Proceso Unificado (RUP): Técnicas Modernas para Desarrollar Aplicaciones.* [En línea] [Citado el: 06 de 02 de 2010.] http://www2.ing.puc.cl/~iic3194/rupppt.pdf .

2. **Jiménez Garzón, Darwin.** [En línea] [Citado el: 11 de 01 de 2010.] http://codeticainge.googlepages.com/guiaing.pdf.

3. **Calero Solís, Manuel.** Una explicación de la programación extrema (XP). *Una explicación de la programación extrema (XP).* [En línea] [Citado el: 12 de 01 de 2010.] http://www.willydev.net/descargas/prev/ExplicaXp.pdf.

4. **Barrientos Enríquez, Aleida Mirian.** El desarrollo de sistemas de información empleando el lenguaje de modelado unificado UML. *El desarrollo de sistemas de información empleando el lenguaje de modelado unificado UML.* [En línea] [Citado el: 17 de 01 de 2010.] http://www.monografias.com/trabajos16/lenguaje-modelado-unificado/lenguaje-modelado-unificado.shtml.

5. **Rojas, Jesús.** [En línea] [Citado el: 14 de 02 de 2010.] http://www.slideshare.net/jesus25dite/lenguaje-sql-3259211.

6. **Alvarez, Rubén.** Qué es y para qué sirve el SQL. *Qué es y para qué sirve el SQL.* [En línea] [Citado el: 24 de 01 de 2010.] http://www.desarrolloweb.com/articulos/262.php.

7. **Lescano, German.** Patrones fundamentales en el diseño orientado a objetos (DOO). *Patrones fundamentales en el diseño orientado a objetos (DOO).* [En línea] 10 de 03 de 2010. [Citado el: 26 de 04 de 2010.] http://germanlescano.wordpress.com/2010/03/10/patrones-.

8. EasyWMS. *EasyWMS.* [En línea] [Citado el: 01 de 02 de 2010.] http://www.mecalux.es/software-almacen/31117946-31117956-pd.html.

9. Por qué utilizar C#. *Por qué utilizar C#.* [En línea] 2010. [Citado el: 17 de 01 de 2010.] http://msdn.microsoft.com/es-es/library/aa287554(VS.71).aspx.

10. Rational Rose Enterprise Edition . *Rational Rose Enterprise Edition .* [En línea] 2002. [Citado el: 05 de 02 de 2010.] http://www.taringa.net/posts/downloads/883942/Rational-Rose-Enterprise-Edition-2002.html.

11. Rational Unified Process. *Rational Unified Process.* [En línea] 2010. [Citado el: 03 de 02 de 2010.] http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42&limit=1&limitstart=6.

12. Sistema de control logístico. *Sistema de control logístico.* [En línea] [Citado el: 24 de 01 de 2010.] http://www.logismarket.es/icp-logistica/sistema-de-control-logistico/614612401-1177375-p.html.

13. Software de gestión de almacenes (SGA). *Software de gestión de almacenes (SGA).* [En línea] 2010. [Citado el: 23 de 01 de 2010.] http://www.logismarket.es/software-gestion-almacenes-sga/1177375-cp.html.

14. Tabla de comparación (RUP vs XP). *Tabla de comparacion (RUP vs XP).* [En línea] 26 de 04 de 2008. [Citado el: 17 de 02 de 2010.] http://tabladecomparacion.blogspot.com/.

15. Visual Studio .NET Professional. La herramienta líder para el desarrollo en plataformas .NET. *Visual Studio .NET Professional. La herramienta líder para el desarrollo en plataformas .NET.* [En línea] [Citado el: 01 de 02 de 2010.] http://www.abox.com/productos.asp?pid=314.

16. **Arroyo, Cristian R.** PostgreSQL 8.4. *PostgreSQL 8.4.* [En línea] 2008. [Citado el: 05 de 02 de 2010.] http://www.vivalinux.com.ar/soft/postgresql-8.4.

17. **Casasola Romero, Oscar.** Introducción a UML. *Introducción a UML.* [En línea] 25 de 11 de 2002. [Citado el: 01 de 02 de 2010.] http://www.programacion.com/articulo/introduccion_a_uml_181/1.

18. **Cueva Lovelle, Juan Manuel.** Introducción a UML. Lenguaje para modelar objetos. *Introducción a UML. Lenguaje para modelar objetos.* [En línea] [Citado el: 14 de 02 de 2010.] http://gidis.ing.unlpam.edu.ar/downloads/pdfs/IntroduccionUML.PDF.

19. **Escribano, G. F.** Introducción a Extreme Programming. *Introducción a Extreme Programming.* [En línea] 2002. [Citado el: 22 de 02 de 2010.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf..

20. **Martín Escofet, Carmen.** El lenguaje SQL. *El lenguaje SQL.* [En línea] 17 de 1 de 2010. http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02149.pdf.

21. **Martínez R., Alejandro.** Guía a Rational Unified Process. *Guía a Rational Unified Process.* [En línea] 2003. [Citado el: 12 de 01 de 2010.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf..

22. **Mayo, Joseph.** *C# al descubierto.* Madrid : Pearson Educación, 2002. pág. 784.

23. **Mellon, I.** Itera: Ingeniería de Software. RUP. *Itera: Ingeniería de Software. RUP.* [En línea] 2009. [Citado el: 15 de 02 de 2010.] http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42..

24. **Pérez Chávez, Roger, y otros.** *Programación orientada a objetos con C#.* Matanzas : Universidad de Matanzas, 2003. pág. 215.

25. **Quiñones A, Ernesto.** INTRODUCCION A POSTGRESQL. *INTRODUCCION A POSTGRESQL.* [En línea] [Citado el: 11 de 02 de 2010.] http://www.eqsoft.net/presentas/introduccion_a_postgresql.pdf.

26. **Rojas Mesa, Yuniet.** De la gestión de información a la gestión del conocimiento. *De la gestión de información a la gestión del conocimiento.* [En línea] http://bvs.sld.cu/revistas/aci/vol14_1_06/aci02106.htm.

27. **Wile, Christoph.** *C#.* Madrid : Pearson Educación, 2001. pág. 208.

Glosario de términos

- Almacén: espacio físico donde se guardan los recursos de las unidades presupuestadas.
- Artefactos: resultados que se obtienen al terminar un flujo de trabajo.
- Atención primaria: atención que se brinda en la red de policlínicos y hospitales de Cuba.
- Balance: cantidad de recursos financieros y materiales consumidos por una unidad presupuestada en un período de tiempo.
- Cobertura: alcance en días de los productos según el consumo.
- Consumo: cantidad de productos que se utilizan por día.
- Gestión de la información: proceso mediante el cual, se obtiene la información procedente de los procesos que se realizan en los almacenes de la OAMS como: la asignación de recursos materiales, los diferentes sistemas de inventarios, la cobertura y consumo de alimentos, permitiendo realizar planes reales.
- Inventarios: relación detallada de los bienes que se encuentran en los almacenes subordinados a la OAMS.
- Plan: cantidad de un producto que se asigna a una unidad presupuestada.
- Programa de Formación de Médicos: proyecto que acoge a jóvenes de diferentes países para que cursen estudios de medicina en Cuba, con el objetivo de llevar la salud pública a todos los rincones del mundo.
- Sistema docente–asistencial: es el sistema que recoge todos los aspectos relacionados con la carrera de medicina.
- Unidad presupuestada: entidad física donde obtienen su preparación los estudiantes latinoamericanos de medicina, y donde los médicos internacionalistas realizan cursos de adiestramiento.
- Veracidad de la información: certeza de que la información que se recibe es real.