PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS

PARA TÍTULOS PROFESIONALES DE TERCER NIVEL
DECLARACIÓN y AUTORIZACIÓN

Yo, **ANA CRISTINA MATHEU VASCO** C.I. **1802543080** autor del trabajo de graduación intitulado: "**DESARROLLO E IMPLEMENTACIÓN DEL PROCESO QUE MANEJA EL DEPARTAMENTO DE NUTRICIÓN DEL HOSPITAL DE NIÑOS BACA ORTIZ"**, previa a la obtención del título profesional de **INGENIERA DE SISTEMAS Y COMPUTACIÓN** en la Facultad de **INGENIERÍA**:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de graduación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 25 de Octubre del 2011

Ana Cristina Matheu Vasco
C.I. 1802543080

FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS

PARA TÍTULOS PROFESIONALES DE TERCER NIVEL
DECLARACIÓN y AUTORIZACIÓN

Yo, **CARLOS RAMIRO CARGUA QUISHPE** C.I. **1717481384** autor del trabajo de graduación intitulado: **"DESARROLLO E IMPLEMENTACIÓN DEL PROCESO QUE MANEJA EL DEPARTAMENTO DE NUTRICIÓN DEL HOSPITAL DE NIÑOS BACA ORTIZ"**, previa a la obtención del título profesional de **INGENIERA DE SISTEMAS Y COMPUTACIÓN** en la Facultad de **INGENIERÍA**:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de graduación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 25 de Octubre del 2011

Carlos Ramiro Cargua Quispe
C.I. 1717481384

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

"DESARROLLO E IMPLEMENTACIÓN DEL PROCESO QUE MANEJA EL DEPARTAMENTO DE NUTRICIÓN DEL HOSPITAL DE NIÑOS BACA ORTIZ"

CRISTINA MATHEU
CARLOS CARGUA

DIRECTOR: ING. EDDY SANCHEZ

QUITO, OCTUBRE DEL 2011

Dedicatoria

A mis padres que me han apoyado siempre, en todos los momentos de mi vida, a mi hermana Daniela que con palabra firme me ha ayudado a no olvidar y tener presente mis objetivos, a mi hermano Dieguito para que este logro sirva de motivación en su vida académica.
A toda mi familia por confiar en mí y ayudarme de una u otra forma.

Cristina Matheu

Agradecimiento

Agradezco a Dios por haberme permitido culminar exitosamente esta etapa de mi vida.
A mi familia por motivarme siempre y apoyarme en mi carrera profesional.
A todas las personas que colaboraron con la elaboración de este proyecto.
Especialmente al Ingeniero Fabián de la Cruz que ha sido un pilar fundamental en el desarrollo de este proyecto, gracias por su paciencia, su tiempo y por compartir sus conocimientos.
A mi compañero de tesis Carlos por su apoyo, perseverancia y optimismo durante el desarrollo del proyecto.
A mis amigas y amigos que han estado acompañándome, por su amistad incondicional.
Al personal del Departamento de Nutrición del Hospital "Baca Ortiz" por la apertura y confianza que tuvieron con nosotros.

Cristina Matheu

Dedicatoria

A mis padres Jorge y Germania, por regalarme el don de la vida.
A mi hermano Ludwig, por ser mi fuente de inspiración y admiración.
Y a todas aquellas personas que a lo largo de mi vida me han ayudado en los momentos más difíciles, gracias por su amistad y apoyo sincero.

Carlos Cargua

Agradecimiento

Agradezco a Dios, por bendecirme con todas mis facultades mentales que me permitieron culminar con éxito una etapa importante en mi vida.

A toda mi familia, gracias por el cariño y apoyo.

A mis grandes amigos, quienes me enseñaron el valor de la amistad.

Un agradecimiento muy especial para Cristina, por ser mi compañera de tesis, quien con su iniciativa hizo posible que este proyecto se llevara a cabo.

Un profundo agradecimiento a la Pontificia Universidad Católica del Ecuador, por formarme íntegramente.

Al Ingeniero Fabián de la Cruz por el apoyo brindado en la realización de este trabajo.

A la Ecónoma Grace Moreno, Directora del Departamento de Nutrición del Hospital de Niños "Baca Ortiz", por facilitarnos la información que hizo posible el desarrollo de todo este trabajo.

Carlos Cargua

ÍNDICE DE CONTENIDOS

Índice de Ilustraciones

Índice de Tablas

RESUMEN

El Departamento de Nutrición del Hospital de Niños "Baca Ortiz" es la organización encargada de planificar, ejecutar, supervisar, evaluar, controlar actividades e impartir directivas y disposiciones pertinentes para la elaboración de la alimentación de los pacientes. En este departamento se identificaron varios problemas que pueden resolverse gracias a la implementación e implantación de una solución computacional que permita llevar una mejor gestión. Esta disertación se enfocará en el desarrollo de un sistema informático capaz de manejar el proceso de Preparación y Elaboración de las dietas alimenticias; uno de los procesos habilitantes del departamento y en el cual intervienen los microprocesos; revisión del Menú y cálculo de cantidades de cada alimento.

Para la implementación del sistema es crucial aplicar ingeniería de software que consiste en definir el proceso de desarrollo de software que incorpora principios, modelos de ciclo de vida, métodos, metodologías y enfoques. La norma IS0 12207 es un principio que establece un marco de referencia para el proceso de ciclo de vida que incluye actividades y tareas que se llevarán a cabo desde la definición y compresión de los requerimientos hasta su finalización. Puede ser aplicada utilizando únicamente un subconjunto de actividades. El ciclo de vida lineal es usado como un marco de trabajo que permite estructurar, planificar y controlar un proyecto de desarrollo con altas posibilidades de alcanzar el éxito. El ciclo de trabajo determina el orden de las etapas involucradas en las fases principales: definición de requerimientos, diseño e implementación y mantenimiento. La metodología tradicional es una colección de métodos que se aplican a lo largo del ciclo de vida de desarrollo de software coherentes entre si y que siguen una filosofía o enfoque de desarrollo de software orientado a objetos.

Luego de identificar los pasos a seguir dentro del proceso de desarrollo de software se seleccionó las herramientas tecnológicas que permitieron generar el sistema informático. Estos recursos tecnológicos son un conjunto de elementos que facilitaron la generación del producto de software. Entre las herramientas útiles que se consideraron están: la herramienta de modelado, el gestor de la base de datos, el lenguaje de programación y el IDE de desarrollo. Una herramienta de modelado permite crear un "simulacro" del sistema, a bajo costo y riesgo mínimo. Un gestor de base de datos permite el manejo de la información dentro de una organización, manteniendo los datos claros y ordenados cumpliendo con objetivos básicos en cuanto al nivel de seguridad, independencia, consistencia, tiempo de respuesta y manejo de transacciones. Con ayuda de Netbeans; un IDE de desarrollo, se codificó, depuró y generó el sistema en lenguaje JAVA.

INTRODUCCIÓN

En nuestros días la automatización de los procesos ha permitido que las organizaciones sean muy eficientes. La mayoría de empresas y organizaciones han evaluado la ineficiencia que representa las organizaciones funcionales, con su visión tradicional, individualista y con una orientación interna, potenciando el concepto de proceso con una visión de integración, de trabajo en equipo, compromiso con metas de la organización y sobre todo orientada al cliente quien es el eje que impulsa a la empresa a mejorar continuamente.

Los procesos representan como una empresa crea y entrega valor a sus clientes internos o externos; además, un proceso es generado a través de una secuencia de actividades orientadas a generar dicho valor añadido sobre una o varias entradas para conseguir un resultado o salida que satisfaga los requerimientos del cliente.

Estas ventajas pueden ser implementadas en el departamento de Nutrición del Hospital de Niños "Baca Ortiz", mediante la implementación de una solución informática que controle el proceso de dimensionamiento de raciones alimenticias, identificando los reales requerimientos para mejorar la eficiencia de dicho departamento y de esta forma mejorar su servicio.

Se seguirá un proceso de desarrollo de software, aplicando conocimientos de Ingeniería de Software. Esperándose como resultado un sistema informático capaz de manejar la información del departamento en donde se permitirá el ingreso de un número de personas con un menú determinado y se obtendrá la cantidad total de ingredientes necesarios para cada momento del día (régimenes alimenticios: desayuno, colación, almuerzo, merienda, cena).

Objetivo General

- Desarrollar un sistema informático automatizado capaz de manejar el dimensionamiento de las comidas del Hospital de Niños "Baca Ortiz", que agilitará y mejorará el servicio del Departamento de Nutrición.

Objetivos Específicos

Identificar los procesos que se llevan a cabo para manejar el servicio de dimensionamiento de porciones para los pacientes y personal del Hospital de Niños "Baca Ortiz".

- Definir el proceso de desarrollo de software más apropiado incorporando principios, modelos de ciclo de vida, métodos, metodologías, enfoques.
- Seleccionar los recursos tecnológicos transversales que permitirán generar el sistema informático.
- Implementar la solución informática siguiendo el proceso de desarrollo de software y haciendo uso de los recursos tecnológicos.
- Implantar el producto de software en un ambiente real.

CAPITULO I

CARACTERIZACIÓN DEL DEPARTAMENTO DE NUTRICIÓN DEL HOSPITAL DE NIÑOS “BACA ORTIZ”

1. CARACTERIZACIÓN DEL DEPARTAMENTO DE NUTRICIÓN DEL HOSPITAL DE NIÑOS "BACA ORTIZ"

El Departamento de Nutrición del Hospital de niños "Baca Ortiz" es el órgano encargado de planificar, ejecutar, supervisar, evaluar, controlar actividades e impartir directivas y disposiciones pertinentes para la elaboración de la alimentación de los pacientes. En este departamento se han identificados funciones, características y procesos que aun no han sido automatizados por una aplicación informática. Es por ello, que se identificaron varios problemas que pueden resolverse gracias al desarrollo e implementación de una solución computacional que permita llevar una mejor gestión. Esta disertación se enfocará en uno de los procesos habilitantes que componen el macroproceso.

1.1 Datos de la Organización o Institución

1.1.1 Nombre

Departamento de Nutrición del Hospital de Niños "Baca Ortiz".

1.1.2 Actividad

El departamento brinda servicio los 365 días del año. Se encarga de la preparación de 5 regímenes alimenticios diarios (desayuno, colación, almuerzo, merienda y cena) debidamente planificados y elaborados basándose en dietas balanceadas y diseñadas por nutricionistas.

1.1.3 Ubicación

El Departamento de Nutrición del Hospital de Niños "Baca Ortiz" se encuentra ubicado, en la Av. Colón y 6 de Diciembre.

1.1.4 Características del departamento

El Departamento de Nutrición está a cargo de personal capacitado, que tiene la atribución de planificar, ejecutar, supervisar, evaluar, controlar las actividades e impartir las directivas y disposiciones pertinentes para la elaboración de menús.

El departamento de Nutrición tiene a su cargo:

- Llevar un control de los ingredientes que se almacenan en bodega. Procurando que estén correctamente refrigerados si lo ameritan, en un lugar fresco y limpio.
- Llevar un control de los proveedores de ingredientes. La Institución se enfoca en la adquisición de ingredientes al por mayor para ahorrar costos.
- Llevar un control de las finanzas para adquirir más ingredientes que intervendrán en posteriores dietas.
- Analizar técnica y analíticamente la preparación de dietas balanceadas para los pacientes y personal del Hospital.
- Planificar una serie de menús que se distribuirán durante todo el mes, los siete días a la semana en un horario establecido.
- Preparar y elaborar las dietas del menú diario siguiendo controles de calidad e higiene.
- Distribuir las raciones de alimento a los pacientes hospitalizados de acuerdo a principios técnicos de preparación y de la terapia dietética.
- Distribuir las raciones de alimento al personal de la Institución.

1.1.5 Estatutos

1.1.5.1 Misión

El Hospital de Niños “Baca Ortiz” es una Unidad Operativa del Ministerio de Salud Pública que brinda atención médica infantil en treinta y siete especialidades pediátricas, en prevención, diagnóstico, tratamiento, y rehabilitación de patologías clínicas y quirúrgicas, con alta calidad profesional, ética, calidez y humanismo, integrando a la familia y los profesionales de la salud en el cuidado general de la salud de niñas y niños del Ecuador, todo en armonía tal que precautele la conservación óptima del medio ambiente.

1.1.5.2 Visión

El Hospital de Niños “Baca Ortiz”, perteneciente al Ministerio de Salud Pública, será líder nacional en atención médica en todas las especialidades pediátricas. Con atención de la más alta calidad, por poseer un talento humano de excelencia académica, con una tecnología de vanguardia. Enmarcando su trabajo diario en los principios de universalidad, eficiencia, eficacia, calidad, equidad y solidaridad.

1.2 Problemas presentados en el departamento

Debido a que el departamento tiene una visión funcional se pudo identificar una serie de problemas que se detallan a continuación:

Como primer problema detectado es que el manejo de la información es realizada de forma manual lo que implica que toda la información sea propensa a errores, lo cual minimiza el desempeño del departamento a mediano y largo plazo.

El iniciar la preparación de las recetas dependerá de la determinación exacta del número de pacientes y personal del hospital. La persona encargada por ende, no puede elaborar ni estimar el total de recetas a prepararse en la "Central de preparaciones" del hospital. Toda la información que recibe esta persona es susceptible de cambios debido a la naturaleza de la institución. Cabe mencionar que el manejo de la información manual también está sujeto al error humano ya que la persona encargada puede digitar erróneamente una cantidad de personas o alimentos que no es la solicitada.

Luego de conocer el número de personas y la dieta a ser preparada, la cantidad de alimentos totales que se van a utilizar en cada dieta son estimados de manera aproximada y confiando en la experiencia de la persona encargada, por lo que el resultado final de esta estimación es propenso a errores.

Posteriormente esta persona registra los ingredientes necesarios en una lista donde se especifican todos los alimentos; con su respectiva unidad que serán utilizados en la preparación de las recetas.

En consecuencia surge otro inconveniente puesto que al ser el control manual, no se tiene una certeza del cien por ciento de que los totales registrados sean precisos lo cual puede acarrear el exceso o la falta física de los alimentos necesarios para la elaboración del menú, sin embargo si la cantidad de algún alimento no fuera suficiente se trata de reemplazarlo por uno de similares características.

1.3 Identificación de los procesos

1.3.1 Macroproceso

El proceso empieza cuando el Departamento de Nutrición recibe una lista con el número de pacientes internados en cada sala del hospital.

El personal encargado, dependiendo del número de pacientes y personal, estima y registra de forma manual la cantidad de todos los ingredientes con sus respectivas unidades de medida basándose en el menú planificado. Este proceso se realiza para cada uno de los régimenes alimenticios (desayuno, colación, almuerzo, merienda y cena) que son preparados diariamente en el hospital. Posteriormente se verifica la existencia de todos los alimentos en bodega, si un alimento está agotado se trata de reemplazarlo por alguno de similares características.

El Departamento de Nutrición también realiza la adquisición de los diferentes insumos que son necesarios en la elaboración y preparación de las dietas alimenticias. Para ello una persona encargada verifica el inventario de los insumos y procede a realizar la correspondiente solicitud de ingredientes a los proveedores, el cual es enviado al departamento financiero para su posterior análisis y aprobación de presupuesto.

1.3.2 Identificación de los procesos principales

El departamento de nutrición se encarga de:

- Liderar la gestión del Departamento de Nutrición mediante el direccionamiento estratégico y el establecimiento de políticas normas y procedimientos que contribuyen en la elaboración de las dietas alimenticias.
- Analizar y planificar técnica y analíticamente las dietas alimenticias para los pacientes y personal del Hospital.
- Gestionar costos e inventario de insumos.
- Gestionar el aprovisionamiento de insumos.
- Preparar las recetas alimenticias.
- Elaborar y distribuir las raciones alimenticias a los pacientes y personal del hospital.

1.3.3 Clasificación según el tipo de proceso

1.3.3.1 Procesos Creador de Valor

Elaborar y distribuir las raciones alimenticias a los pacientes y personal del Hospital.

1.3.3.2 Procesos Habilitantes

- Analizar y planificar técnica y analíticamente las dietas alimenticias para los pacientes y personal del Hospital.
- Gestionar costos e inventario de insumos.
- Gestionar el aprovisionamiento de insumos.
- Preparar las recetas alimenticias.

1.3.3.3 Procesos Gobernantes

- Liderar la gestión del Departamento de Nutrición mediante el direccionamiento estratégico y el establecimiento de políticas normas y procedimientos que contribuyen en la elaboración de las dietas alimenticias.

1.3.4 Mapa de procesos

A continuación se muestra el mapa de los procesos encontrados en el Departamento de Nutrición:

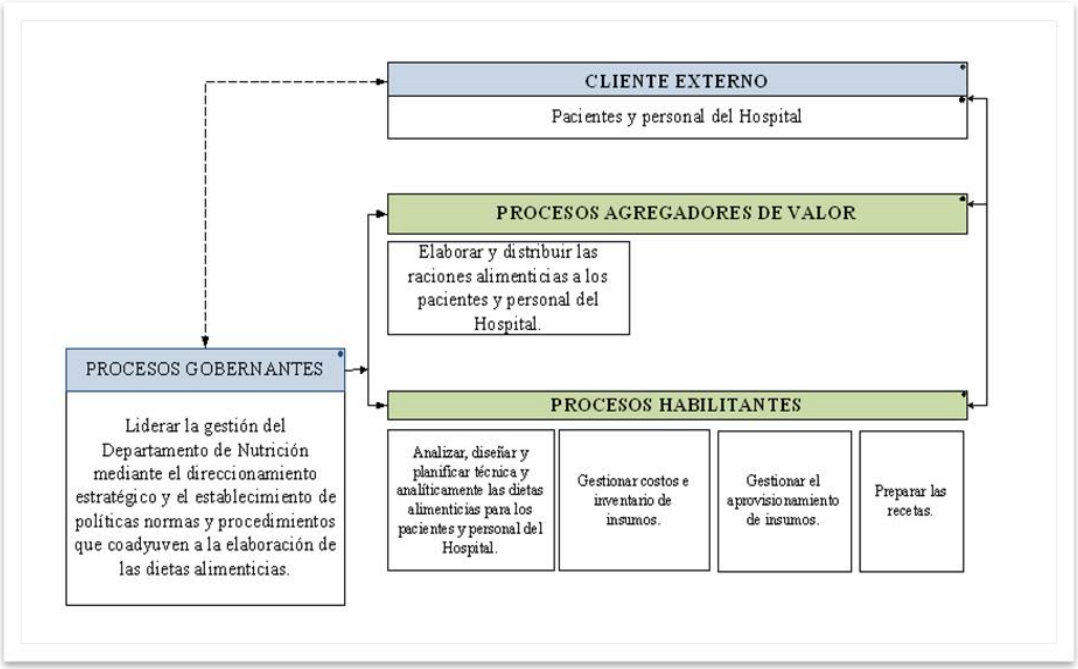


Ilustración 1-1: Mapa de procesos. [A]

1.3.5 Descripción de los Subprocesos

1.3.5.1 Cadena de valor del macroproceso

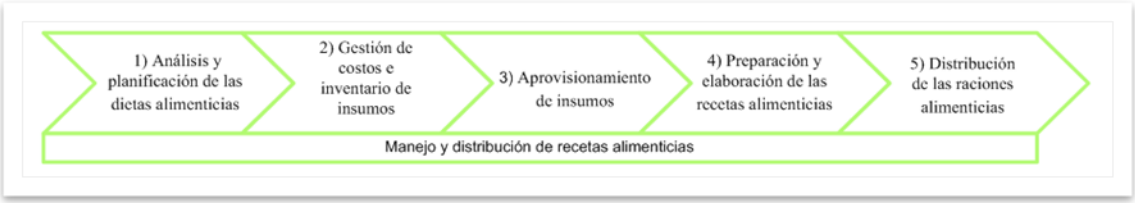


Ilustración 1-2: Cadena de valor del macroproceso. [A]

❖ Análisis y planificación de las dietas alimenticias

- El departamento de nutrición del Hospital de Niños Baca Ortiz cada año planifica y analiza los menús con las dietas.
- Después de analizar los menús con las dietas anteriores, diseñan y formulan los nuevos menús.

Ilustración 1-3: Subprocesos del análisis y planificación de las dietas alimenticias. [A]

❖ Gestión de costos e inventario de insumos

- La cantidad total utilizada de cada ingrediente es registrada en una lista en la que posteriormente se utilizará para llevar el inventario.
- Al ser el proceso de cálculo realizado de manera manual se corre el riesgo de cometer errores y no realizar correctamente los cálculos con las cantidades correctas de cada alimento o peor aun no registrar todos los alimentos utilizados, lo que hará que se dé un desfase entre las existencias reales de los alimentos y las que se encuentran registradas.
- Después de registrar la cantidad total de cada uno de los ingredientes necesarios para la preparación de la receta se verifica la existencia de todos los alimentos en bodega, si un alimento está agotado se trata de reemplazarlo por alguno de similares características.
- A la finalización de cada mes se procede a realizar una solicitud al departamento financiero con el presupuesto para la compra de los insumos faltantes.
- Con el desembolso del departamento financiero se procede a la adquisición de los alimentos.

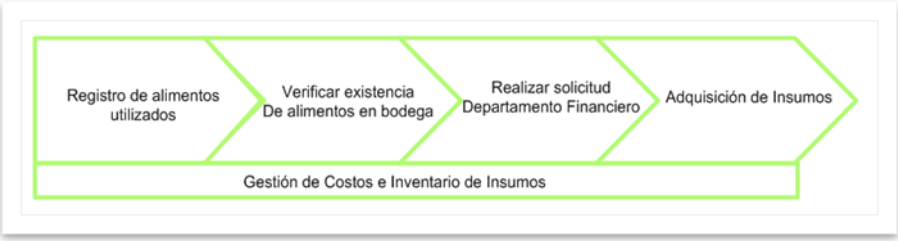


Ilustración 1-4: Subprocesos gestión de costos e inventario de insumos. [A]

❖ Aprovisionamiento de Insumos

- Los proveedores entregan los productos al departamento de nutrición.
- Las cantidades de los productos adquiridos son registradas en una lista para llevar el control de existencias.

Ilustración 1-5: Subprocesos Aprovisionamiento de insumos. [A]

❖ Preparación de las recetas alimenticias

- El proceso empieza cuando el departamento de nutrición recibe una lista con el número de pacientes por departamentos de todo el hospital.
- Conociendo el número tanto de pacientes como de personal del hospital se procede a revisar el menú que los nutricionistas de departamento han planificado.
- Posteriormente la persona encargada realiza el cálculo de cada uno de los ingredientes que se necesitarán para la elaboración de la receta a preparar.
- Existen listas de todos los alimentos utilizados en el hospital que de manera manual y basándose en el menú se irán registrando de uno en uno las cantidades totales en la unidad de medida correspondiente.

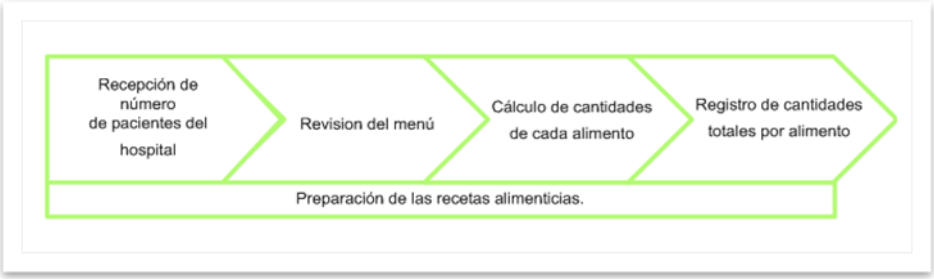


Ilustración 1-6: Subprocesos de la preparación de las recetas alimenticias. [A]

❖ Elaboración y Distribución de las raciones alimenticias

- Después de registrar la cantidad total de cada uno de los ingredientes necesarios para la preparación de la receta se verifica la existencia de todos los alimentos en

bodega, si un alimento está agotado se trata de reemplazarlo por alguno de similares características.

- Con los alimentos disponibles el personal del departamento de nutrición elabora las recetas.
- Luego de haber elaborado las recetas el personal del departamento de nutrición distribuye tanto al personal como a los pacientes de las diferentes áreas las raciones de comida.

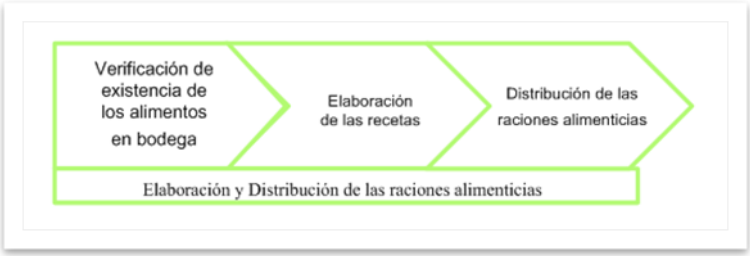


Ilustración 1-7: Subprocesos de la elaboración y distribución de las raciones alimenticias. [A]

1.4 Intervención de la solución informática en un proceso

Luego de enumerar los procesos y los subprocesos es necesario detallar la intervención de la solución informática para la mejora del proceso.

El sistema se aplicará en el proceso de Preparación y Elaboración de las recetas alimenticias en el cual intervienen los subprocesos: recepción de número de pacientes del hospital, revisión del menú, cálculo de cantidades de cada alimento y registro de cantidades totales por alimento. La revisión del menú y el cálculo de cantidades de cada alimento serán mejorados al guardarse en un sistema computacional. Esto implicará que posterior a la selección de un menú planificado de un solo día, en el sistema se realizarán cálculos de dimensionamiento de cada ingrediente necesario para la elaboración de cada receta para un determinado número de personas.

CAPITULO 2
PROCESO DE DESARROLLO DE SOFTWARE

2. PROCESO DE DESARROLLO DE SOFTWARE

Parte crucial dentro de la ingeniería de software consiste en definir el proceso de desarrollo de software. Este proceso incorpora principios, modelos de ciclo de vida, métodos, metodologías, enfoques.

La norma IS0 12207 es una norma que establece un marco de referencia para el proceso de ciclo de vida que incluye actividades y tareas que se llevarán a cabo desde la definición y compresión de los requerimientos hasta su finalización. Puede ser aplicada por una organización utilizando únicamente un subconjunto de actividades.

El ciclo de vida es usado como un marco de trabajo que permite estructurar, planificar y controlar un proyecto de desarrollo con altas posibilidades de alcanzar el éxito. El ciclo de trabajo determina el orden de las etapas involucradas en las fases principales: definición de requerimientos, diseño e implementación y mantenimiento.

Los métodos son formas explicitas para realizar una o varias actividades requeridas. Es un proceso riguroso y sistemático para generar un conjunto de modelos que describen varios aspectos de un sistema de software en desarrollo. Los métodos especifican como construir técnicamente el software.

Las metodologías son una colección de métodos aplicados a lo largo del ciclo de vida de desarrollo de software coherentes entre si y que siguen una filosofía o enfoque de desarrollo de software.

Una forma particular de ver y organizar los pasos contenidos en el ciclo de desarrollo de software es lo que se define como enfoque.

2.1 Especificación de Metodología

2.1.1 Definición

Es un conjunto de pasos o procedimientos que ayudan a la creación del producto utilizando técnicas y herramientas como también la realización de la respectiva documentación, de esta forma se logrará controlar el desarrollo de cada fase o etapa del software a desarrollarse, existiendo una mayor probabilidad de tener éxito.

La finalidad de las metodologías es formalizar un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo predecible, eficiente y libre de errores.

Una metodología comprende un conjunto de métodos que guían en la planificación y en el desarrollo del software. Permiten definir cómo y cuando se deberá realizar determinada tarea optimizando el proceso y el producto final.

El propósito de una metodología es comprender la realidad, enfocándose en ciertos principios generales que ayuden en la resolución de un problema. Combina ciertos métodos que poseen técnicas, herramientas y tareas que, de acuerdo a un enfoque metodológico aportan en soluciones óptimas.

"En la actualidad se busca métodos que permitan optimizar el trabajo y al mismo tiempo ofrezcan escalabilidad y seguridad en la entrega de un producto de software." [1]

2.2 Clasificación

Según la filosofía de desarrollo se pueden clasificar en Metodologías tradicionales y Metodologías ágiles.

2.2.1 Metodología Tradicional

Metodología basada en la documentación y planificación y el cumplimiento de esta durante el desarrollo del proyecto.

[1]Tesis "Estudio De las metodologías de cuarta generación para desarrollo del software y propuesta de aplicaciones en empresa de la ciudad de quito." , Autora: Proaño Mosquera María Isabel , Quito - 2005

2.2.2 Metodología Ágil

Metodología basada en la adaptabilidad de los procesos de desarrollo. Dando relevancia a la capacidad de respuesta a un cambio y no al seguimiento estricto de un plan. En esta metodología el desarrollo de software es incremental, cooperativo, sencillo y adaptable.

2.3 Enfoques o paradigmas de desarrollo de software

2.3.1 Definición de enfoque

Es una manera particular de ver y organizar los pasos contenidos en el ciclo de desarrollo de software, se puede definir también como una metodología para la construcción del software que puede ser: Orientado por procedimientos o estructurados y Orientado por objetos.

2.3.2 Tipos de enfoque

2.3.2.1 Orientado por procedimientos o estructurados

Crean modelos del sistema orientados a procesos, flujos y estructura de datos de manera descendente o jerárquica, descomponiendo la finalidad del sistema. Comienza con una visión general del problema con un nivel alto de abstracción hasta un nivel de abstracción sencillo.

Se utilizan varias herramientas. Por ejemplo:

- Diagramas de flujo de datos (DFD).
- Diccionario de datos.
- Diagramas de transición de estados.
- Diagramas entidad-relación.

❖ Diagrama de flujo de datos (DFD)

Es una técnica utilizada para la representación gráfica de un algoritmo o proceso que está compuesto por operaciones, decisiones lógicas y ciclos repetitivos, y es representado mediante símbolos con significados definidos y que indican cada uno de los pasos de un algoritmo en el orden en el que es ejecutado mediante flechas que conectan los puntos desde

su inicio hasta su fin. Puede ser utilizado en varias disciplinas como en programación, economía, procesos industriales y psicología cognitiva.
Los diagramas de flujo facilitan a otras personas la comprensión de la secuencia que se lleva a cabo para la solución de un problema o representación de pasos de un proceso.

Una de las características clave de este tipo de diagrama es que poseen únicamente un punto de inicio y uno de final.

Un algoritmo se compone por operaciones, decisiones lógicas y ciclos repetitivos que se representan gráficamente por medio de símbolos estandarizados por ISO y ANSI (American National Standards Institute).

Los principales símbolos para elaborar los diagramas de flujo son:

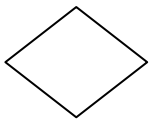
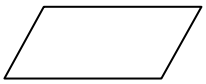
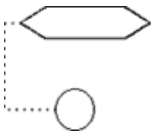
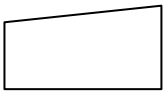
GRAFICO	DESCRIPCIÓN	GRAFICO	DESCRIPCIÓN
 Inicio/Final	Indica el inicio y el final de un diagrama. Sólo puede salir una línea de flujo. El final sólo debe llegar una línea.	 Decisión	Indica la comparación de dos datos y dependiendo del resultado lógico (falso o verdadero) se toma la decisión de escoger la una opción o la otra.
 Entrada General	Entrada/Salida de datos.	 Iteración	Indica que una instrucción o grupo de instrucciones deben ejecutarse varias veces.
 Entrada por teclado	Instrucción de entrada de datos por teclado. Indica que el computador debe esperar a que el usuario teclee un dato que se	 Salida Impresa	Indica la presentación de uno o varios resultados en forma impresa.

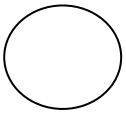
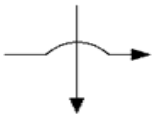
GRAFICO	DESCRIPCIÓN	GRAFICO	DESCRIPCIÓN
	guardará en una variable o constante.		
 Llamada a subrutina	Indica la llamada a una subrutina o procedimiento determinado.	 Salida en Pantalla	Instrucción de presentación de mensajes o resultados en pantalla.
 Acción/Proceso General	Indica una acción o instrucción general que debe realizar el computador (cambios de valores de variables, asignaciones, operaciones aritméticas, etc.)	 Conector	Muestra el enlace de dos partes de un diagrama dentro de la misma página.
 Flujo	Indica el seguimiento lógico del diagrama y el sentido de ejecución de las operaciones.	 Conector	Indica el enlace de dos partes de un diagrama en páginas diferentes.

Tabla 2-1: Principales símbolos para elaborar un DFD. [1]

❖ Diccionario de Datos

Registro de todos los datos pertinentes al sistema y que se encuentran en los diagramas de flujo, con definiciones precisas para que el usuario y el analista tengan el mismo punto de vista y de esta forma un entendimiento común de todas las entradas, salidas y componentes.

❖ Especificaciones de Proceso

Es la descripción de cada uno de los pasos definidos en los diagramas de flujo en el nivel más bajo. Su propósito es definir lo que debe hacerse para transformar entradas en salidas.

Se puede realizar de varias formas entre las que se pueden mencionar pseudocódigo, con tablas de decisión o en un lenguaje de programación.

Deberá ser especificado en una forma que pueda ser comunicado efectivamente al público que se encuentre involucrado como por ejemplo usuarios administradores, auditores, personal de control de calidad y otros, puesto que la persona que lea la especificación del proceso sea capaz de entenderla sin dificultades. La especificación del proceso debe ser expresada de tal manera que puedan verificar tanto el usuario como el analista.

➢ *Diagramas entidad-relación*

Son utilizados para representar problemas, también permiten representar de manera global un sistema. Sus elementos fundamentales como su nombre lo indica son las entidades y las relaciones.

➢ *Entidad*

Es la representación de un tipo de objeto, real o abstracto, del problema a ser modelado. Toda entidad posee nombre y atributos definidos en un dominio determinado. En el diagrama Entidad - Relación las entidades son representadas mediante rectángulos.

Toda entidad debe ser identificada y distinguible mediante un atributo denominado identificador, clave principal o primaria. Puede haber varios posibles identificadores para una misma entidad, en tal caso se escogerá uno de ellos como identificador principal siendo el resto identificadores alternativos.

➢ *Relación*

Es una asociación matemática entre dos o más entidades. Las relaciones son nombradas con verbos. Se representan en el diagrama Entidad-Relación mediante flechas y rombos. Cada entidad interviene en una relación con una determinada cardinalidad.

- *Cardinalidad*

Número de instancias o elementos de una entidad que pueden ser relacionadas a un elemento de otra.

Son posibles las siguientes cardinalidades:

- Una a una (1:1): Una entidad A se relaciona únicamente con una entidad B y viceversa.
- Una a muchas (1: N): Una entidad A se relaciona con cero o muchas entidades en B. Pero una entidad B se relaciona con una única entidad A.
- Muchas a una (N: 1): Una entidad A se relaciona exclusivamente con una entidad B. Pero una entidad B se puede relacionar con 0 o muchas entidades A.
- Muchas a muchas (N: N): Una entidad en A se puede relacionar con 0 o muchas entidades en B y viceversa.

Los elementos del modelo Entidad Relación se corresponden con almacenes de datos en el DFD.

2.3.2.2 Enfoque orientado a objetos

Este enfoque identifica los tipos de datos que hay que utilizar, que características poseen y como están relacionados.

La orientación a objetos supone un paradigma distinto del tradicional que enfoca la atención en las estructuras de datos.

El primer lenguaje orientado a objetos fue Simula67, desarrollado por Kristen Nggaard y Ole-Johan Dahl. Varios años después de su desarrollo, la mayor parte de los lenguajes modernos comenzaron a utilizar los principios de orientación a objetos de Simula67 el cual se basa en componentes facilitando la reutilización de código desarrollado por terceras personas. Adicionalmente, el mantenimiento de un sistema desarrollado con esta metodología se lo puede realizar de una manera más ágil puesto que permite localizar donde se debe realizar un cambio.

- Objetos

Un objeto es una entidad que está compuesta por una estructura de datos y de una colección de métodos (procedimientos o funciones) que manipulan estos datos. Los datos definidos dentro de un objeto son sus atributos. Un objeto solo puede ser manipulado a través de su interfaz, es decir una colección de funciones que implementa y que son visibles al exterior.

- Atributos

Los atributos son datos de una clase. Un atributo está compuesto por un nombre y el valor que se le asigne a este. Dependerá si la declaración del atributo es pública ya que esto significará que el elemento definido pueda utilizarse libremente desde cualquier otro punto del programa.

- Métodos

Los métodos de una clase son funciones propias que forman parte de la definición y describen el comportamiento asociado a un objeto. La ejecución de un método puede conducir a cambiar el estado del objeto o dato local del objeto. Todos los métodos que alteran o acceden a los datos de un objeto se definen dentro del objeto.

Cada método tiene un nombre y un cuerpo que realiza la acción o comportamiento asociado con el nombre de éste. No se pueden modificar los datos (atributos) de otros objetos directamente, sino que se ha de llamar a los métodos de dichos objetos para que los modifiquen.

- Clase

La clase es un modelo o prototipo que define las variables y métodos comunes a todos los objetos de cierta clase. Se puede decir que una clase es una plantilla genérica para un conjunto de objetos de similares características. Siendo los objetos y las instancias la representación de una clase.

Las características básicas se enumeran a continuación:

- Herencia

Relación entre clases, características comunes entre varias clases, la clase que contiene los atributos comunes es la clase antecesora denominada Superclase o clase Padre y clases que se especializan en particularidades, subclases o clases hijas.

- Polimorfismo

Es la propiedad que da la posibilidad de acceder a un variado rango de funciones distintas a través de la misma interfaz. Siendo el polimorfismo la capacidad de los objetos de una clase para responder de manera diferente frente a un mismo evento dependiendo a los parámetros utilizados.

- Encapsulamiento

Se encarga de mantener ocultos los procesos internos permitiendo identificar únicamente las características básicas simplificando el uso y/o modificación.

El encapsulamiento ofrece la facilidad para manejar la complejidad, puesto que las Clases se las puede comparar como cajas negras que únicamente permiten conocer el comportamiento pero no los detalles internos, por lo que se conocerá únicamente qué hace la Clase pero no será necesario saber cómo lo hace.

- Abstracción

Propiedad que permite captar las características más importantes de un objeto de tal manera que se realicen las funciones requeridas sin necesidad de conocer el proceso interno, centrándose en la vista externa de un objeto para de esta forma separar las características esenciales de las que no lo son.

2.4 Lenguaje unificado de modelado UML (Unified Modeling Language)

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group).

UML es un lenguaje de modelado visual utilizado para la especificación, construcción y documentación de un sistema. Permite modelar la estructura de las aplicaciones, su comportamiento y arquitectura. También permite modelar procesos de negocio y estructura de datos.

UML es considerado una técnica para especificar los diferentes sistemas en todas sus etapas facilitando el entendimiento de procesos tanto grandes y complejos como simples y pequeños.

UML constituye uno de los pilares fundamentales en las etapas de diseño ya que ofrece la creación de un "plano" del sistema (modelo).

Es importante tener en cuenta que UML no es una metodología de desarrollo; es decir, no indica que actividad realizar, que orden seguir, o que modelo de ciclo de vida usar.

2.4.1 Diagramas

En UML2.0 existen 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente, como se muestra en la figura.

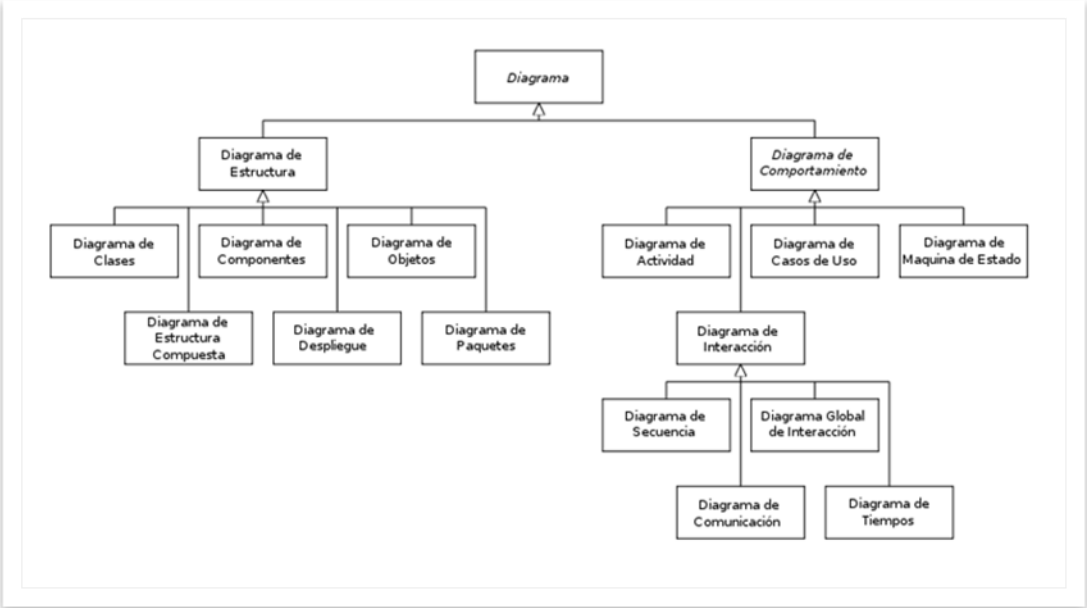


Ilustración 2-1: Jerarquía de los diagramas UML 2.0 mostrados como un diagrama de clases. [2]

2.4.1.1 Diagramas de estructura

Se enfocan en los elementos estáticos que deben existir en el sistema a modelarse:

- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de componentes.
- Diagrama de despliegue.
- Diagrama de paquetes.
- Diagrama de estructura compuesta (UML 2.0).

❖ Diagrama de clases

La principal finalidad de los diagramas de clases es indicar tanto las relaciones existentes entre ellas como también los tipos de objetos que componen a cada una de las clases.

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

- *Asociaciones entre clases*

Representan tipos de compartición entre clases, o relaciones semánticas.

- Asociación: Indica relaciones de mandato direccionales.
- Composición: significa utilizar objetos dentro de otros objetos.
- Herencia: Decir que una clase hereda de otra quiere decir que esa clase obtiene los mismos métodos y propiedades de la otra clase.
- Agregación: representa una relación del tipo "tener un" entre clases.
- Uso: es un refinamiento de la asociación donde se especifica cuál es el cliente y cual el servidor de ciertos servicios.

- *Características*

- Clase: Es un contenedor de uno o más datos (variables o propiedades miembro) junto a las operaciones de manipulación de dichos datos (métodos). Las clases habitualmente se denotan con sustantivos en singular.
- Variables miembro: las propiedades o atributos son características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Habitualmente, las variables miembro son privadas al objeto (siguiendo las directrices de diseño del Principio de ocultación) y su acceso se realiza mediante propiedades o métodos que realizan comprobaciones adicionales. Suelen denominarse con nombres.
- Métodos en las clases: Implementan la funcionalidad asociada al objeto. Los métodos son el equivalente a las funciones en los lenguajes estructurados. Se diferencian de ellos en que es posible acceder a las variables de la clase de forma implícita. Cuando se desea realizar una acción sobre un objeto, se dice que se le manda un mensaje invocando a un método que realizará la acción. Habitualmente, los métodos suelen ser verbos.
- Propiedades: Las propiedades son un tipo especial de métodos. Debido a que suele ser común que las variables miembro sean privadas para controlar el acceso y mantener la coherencia, surge la necesidad de permitir consultar o modificar su valor mediante pares de métodos: GetVariable y SetVariable.

El nivel de acceso a las variables y métodos puede ser:

- Public (+): El atributo o método puede ser visto dentro y fuera de la clase.
- Private (-): El atributo o método será únicamente accedido desde la clase.
- Protected (#): El atributo o método no podrá ser accesado desde fuera de la clase pero si desde los métodos de la clase o de las clases q se derivan de ésta.

2.4.1.2 Diagrama de Objetos

Son un subtipo de un diagrama de clase que hace énfasis en mostrar los objetos y las relaciones existentes entre ellos. Ejemplificando una situación que puede darse en un momento determinado. Los diagramas de objetos expresan la parte estática de una interacción. Los diagramas de objetos no tienen compartimientos a diferencia de los diagramas de clase que poseen tres compartimientos.

2.4.1.3 Diagrama de componentes

Se centran en la identificación del escenario inicial de la arquitectura ya que permiten modelar los componentes de software de alto nivel; lo cual permite que la organización de actividades de desarrollo sea más fácil.

Existen dos estrategias fundamentales para el desarrollo de un modelo de componentes: "top down" (de arriba hacia abajo) y "bottom up" (de abajo hacia arriba).

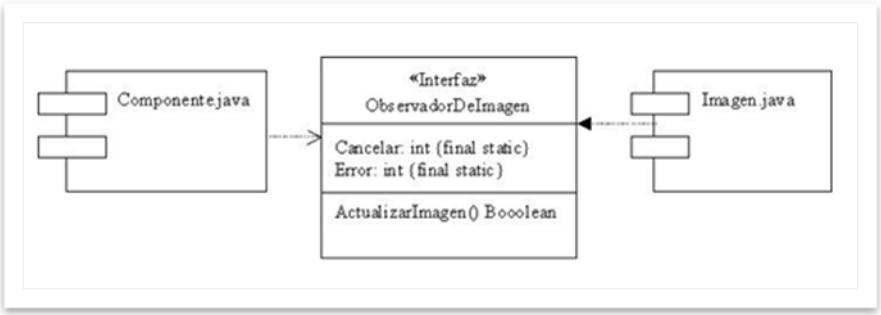


Ilustración 2-2: Ejemplo de un diagrama de componentes [3]

2.4.1.4 Diagrama de Despliegue

Muestran el hardware y software del sistema instalado en el hardware y el software intermedio para conectar las máquinas entre sí. También representan una vista estática del tiempo de ejecución de los nodos de procesamiento y los componentes que se ejecutan.

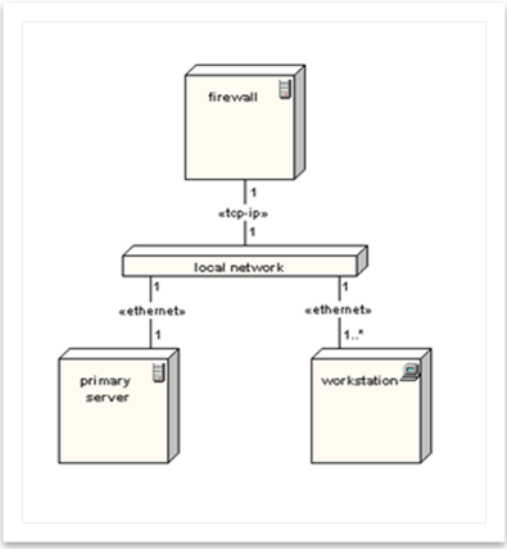


Ilustración 2-3: Ejemplo de un diagrama de despliegue. [4]

2.4.1.5 Diagrama de paquetes

Consiste en el agrupamiento de clases en un nivel más alto.
Los diagramas de paquetes son usados para organizar elementos del modelo en grupos, haciéndolos más simples y fáciles de entender.

El diagrama de paquetes es utilizado generalmente para agrupar el diagrama de clases y los diagramas de casos de uso puesto que estos tienden a crecer. Se representan como carpetas de archivos.

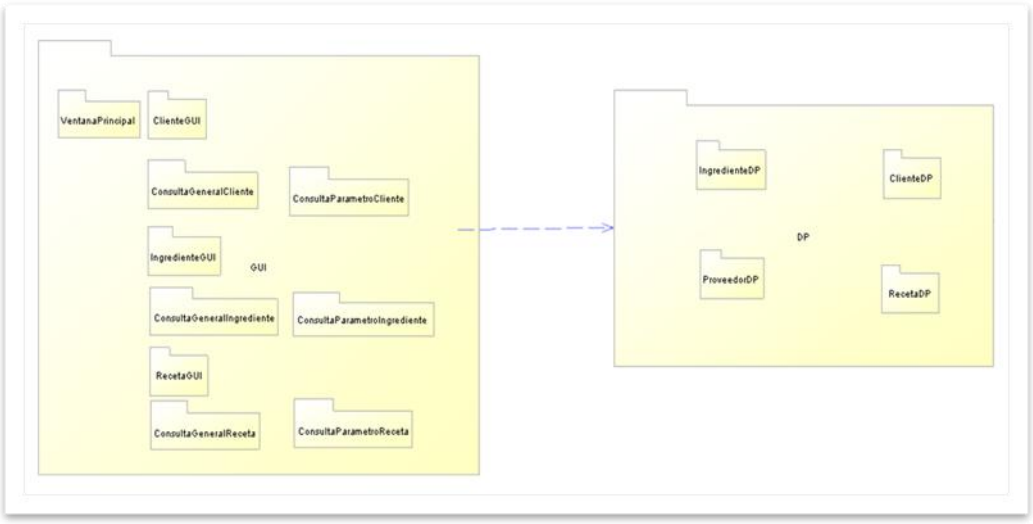


Ilustración 2-4: Ejemplo de un diagrama de paquetes. [A]

2.4.1.6 Diagrama de estructura compuesta (UML 2.0)

Muestra la estructura interna de una clase y las colaboraciones que esta estructura hace posibles.

Una estructura compuesta es un conjunto de elementos interconectados que colaboran en tiempo de ejecución para lograr algún propósito. También se definen como un conjunto de elementos donde cada uno posee un rol definido para colaborar en tiempo de ejecución y alcanzar un propósito.

❖ Características

- Parte: Elemento que representa un rol jugado en tiempo de ejecución por una instancia de una clase. La parte puede incluir un factor de multiplicidad (cardinalidad).
- Puerto: Conecta clasificadores estructurados con sus partes y con el ambiente. Se representa por un rectángulo.
- Conector: Relaciona dos o más entidades, permitiéndoles interactuar en tiempo de ejecución. Un conector es representado por una flecha que une una combinación.
- Colaboración: Definen un conjunto de roles usados colectivamente para ilustrar una funcionalidad específica. Ésta es mostrada como un óvalo sin relleno conteniendo los roles que las instancias pueden jugar en la colaboración.
- Clasificador estructurado: Representa una clase, frecuentemente una clase abstracta, cuyo comportamiento puede ser completa o parcialmente descrito mediante interacciones entre partes.

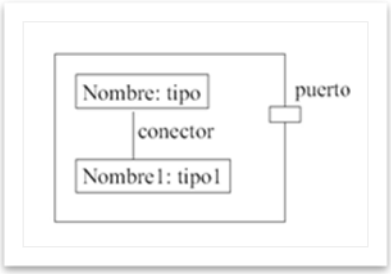


Ilustración 2-5: Ejemplo de un diagrama de estructura compuesta. [A]

2.4.1.7 Diagramas de comportamiento

Muestran el comportamiento en tiempo de ejecución del sistema:

- Diagrama de actividades.
- Diagrama de casos de uso.
- Diagrama de estados.

❖ Diagrama de actividades

Permite visualizar y comprender la dinámica de un proceso ayudando a los miembros del equipo de desarrollo a entender como es utilizado el sistema y cómo reacciona en determinados eventos y las distintas rutas que pueden irse desencadenando. Un diagrama de actividades puede manejar procesos paralelos y secuenciales.

➢ *Características*

- Carriles (Swimlanes): Un diagrama de actividades utiliza carriles dando la posibilidad de asignar responsabilidades de las actividades de un proceso. Para el uso de carriles se organizan las clases de manera vertical y separada por líneas.
- Inicio: Está representado por un círculo.
- Actividad: Acciones que serán realizadas.
- Transacción: Cambio de una actividad a otra, representada por una flecha.

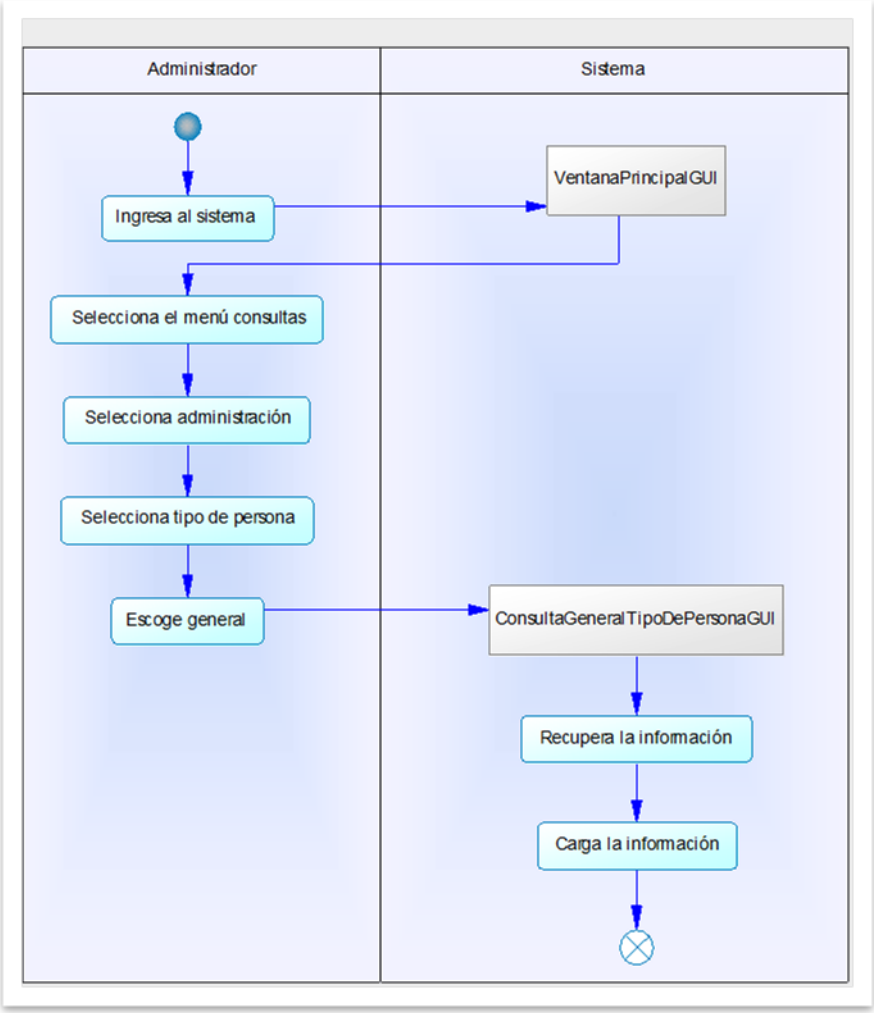


Ilustración 2-6: Ejemplo de un diagrama de actividades. [A]

❖ Diagrama de casos de uso

Los casos de uso se basan en la descripción de escenarios (casos) en los cuales los usuarios interactúan con el sistema que se ha definido para alanzar un objetivo o para cumplir una tarea en particular.

Los casos de uso son una técnica para entender cómo funcionará un sistema conociendo que operaciones pueden ser repetitivas y si existen situaciones opcionales que el usuario pueda ejecutarlas.

Los casos de uso dan la posibilidad de analizar y comprender el sistema. Es importante mencionar que los casos de uso describen que hace el sistema, mas no como lo hace.

➢ *Características*

- Inicio: cuándo y qué actor lo produce. La interacción: mensajes que son recibidos después de las acciones realizadas por el usuario.
- Fin: cuándo se produce y qué valor es devuelto por el sistema.
- Actor: conjunto de roles que el usuario juega en los casos de uso al interactuar con ellos.
- Casos de Uso: comportamiento o acción específica que se realiza luego de una orden realizada por el actor o por la petición de otro caso de uso.
- Relaciones:
- Generalización: (generalization): Es una relación que amplía la funcionalidad de un Caso de Uso o refina su funcionalidad original mediante el agregado de nuevas operaciones y/o atributos y/o secuencias de acciones.
- Inclusión (include): La relación mediante la cual el comportamiento de un caso de uso debe ser incluido, se ejecuta todo el comportamiento del casos de uso incluido para luego continuar con el caso de uso original es decir se re-usa un Caso de Uso encapsulándolo en distintos contextos a través de su invocación desde otros Casos de Uso.
- Extensión (extend): Es usado cuando un caso de uso es similar a otro pero que amplía la funcionalidad de este por medio de la extensión de las secuencias de acciones.

Ilustración 2-7: Ejemplo de un diagrama de casos de uso. [A]

❖ Diagrama de estados

Una máquina de estados es un comportamiento que especifica las secuencias de estados por las que pasa un objeto durante su vida, en respuesta a eventos, junto con sus respuestas a esos eventos.

El diagrama de estados muestra la sucesión de estados por los que pasa un caso de uso o un objeto a lo largo de su vida, pudiendo ser también todo el sistema. En éste se indican qué eventos o acciones deben darse para pasar de un estado a otro y cuáles son las respuestas que genera.

- *Características:*

  - Estado: está representado con una caja redondeada con el nombre del estado en su interior.
  - Transición: se representa como una flecha desde el estado origen al estado destino.
  - Estado inicial de creación y estado final de destrucción: el estado inicial se muestra como un círculo sólido y el estado final como un círculo sólido rodeado de otro círculo. El objeto no puede "estar" en estos estados, pero sirven para conocer cuáles son las transiciones inicial y final(es).

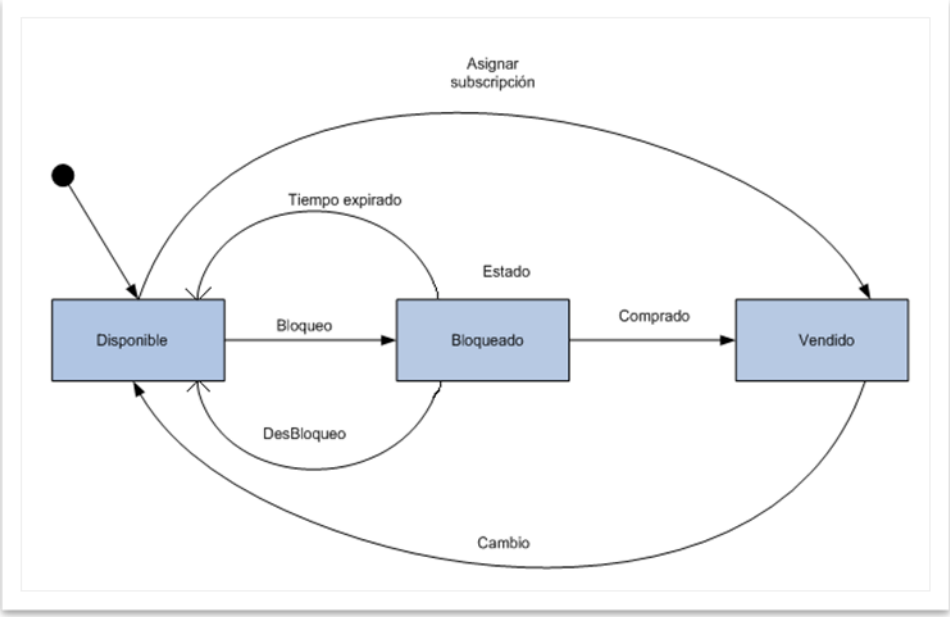


Ilustración 2-8: Ejemplo de un diagrama de estados. [A]

2.4.1.8 Diagramas de interacción

Son derivados de los diagramas de comportamiento, se enfocan en el flujo de control y de datos entre los elementos del sistema.

- Diagrama de secuencia.
- Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración (UML 1.x).
- Diagrama de tiempos (UML 2.0).
- Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0).

❖ Diagrama de Secuencia

Como su nombre lo indica muestra la sucesión e intercambiados entre el usuario y el sistema poniendo énfasis en el orden cronológico en el que son enviados y recibidos los mensajes. Los diagramas de secuencia ayudan a definir y entender el flujo de información de una clase a otra y el comportamiento que se quiere obtener.

➢ *Características:*

- Estructura: los diagramas de secuencia están compuestos por el eje vertical en el que está representado el tiempo, y los mensajes representados con flechas que son enviados de un objeto a otro con los nombres de la operación y los parámetros. Los objetos que inician la interacción se encuentra en el lado izquierdo del diagrama, las clases son colocadas a la derecha.
- Línea de vida del objeto: Representa la existencia de un objeto en el transcurso del tiempo simbolizado con una línea entrecortada vertical. Los objetos que se alinean al tope del diagrama son aquellos que existen durante toda la interacción.
- Foco de control: Muestra el período de tiempo durante el cual un objeto realiza una acción en forma directa o a través de un proceso subordinado, está representado por un largo y delgado rectángulo.

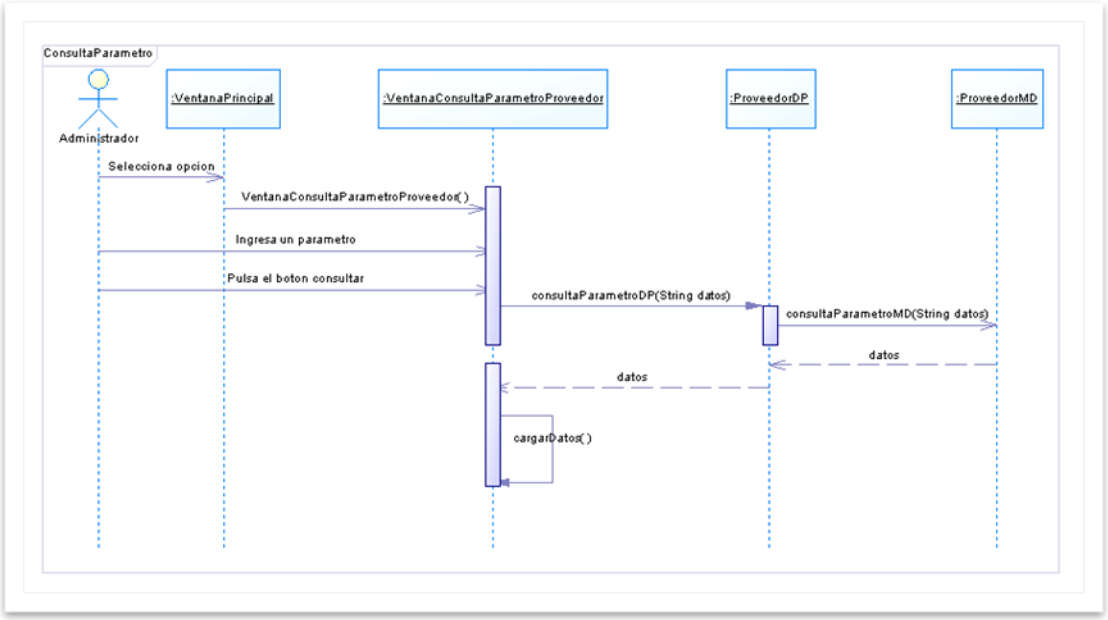


Ilustración 2-9: Ejemplo de un diagrama de secuencia. [A]

❖ Diagrama de comunicación

Los diagramas de comunicación antes conocidos como diagramas de colaboración se enfocan en la organización de los objetos que participan en una interacción, mostrando el comportamiento de varios objetos que colaboran juntos para lograr un objetivo en común. El diagrama de comunicación muestra las instancias de las clases, y el flujo de mensajes que son recibidos y enviados entre ellas.

➢ *Características:*

- Presentación de un camino.
- Secuencia numérica que indicará el orden en el tiempo de los mensajes.

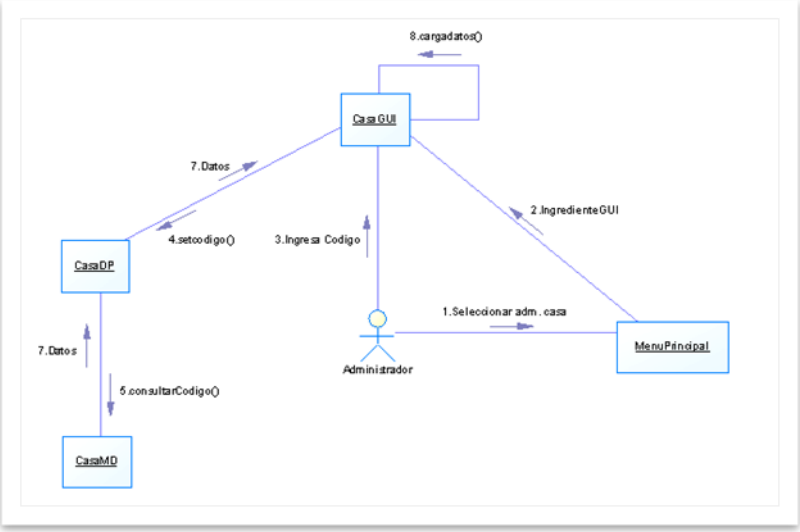


Ilustración 2-10: Ejemplo de un diagrama de comunicación. [A]

❖ Diagrama de tiempos (UML 2.0)

Los diagramas de tiempo son uno de los más recientes artefactos agregados a UML 2.0. Estos diagramas son usados para estudiar el comportamiento de uno o más objetos a lo largo de un período de tiempo determinado y mostrando el cambio de valor o estado que este pudo haber tenido en respuesta a eventos externos.

➢ *Línea de vida del estado*

- Indica el cambio de estado de ítem en el tiempo. De manera horizontal se muestra el tiempo trascurrido en cualquier unidad que se elija mientras que de manera vertical se nombra con una lista de estados proporcionados.

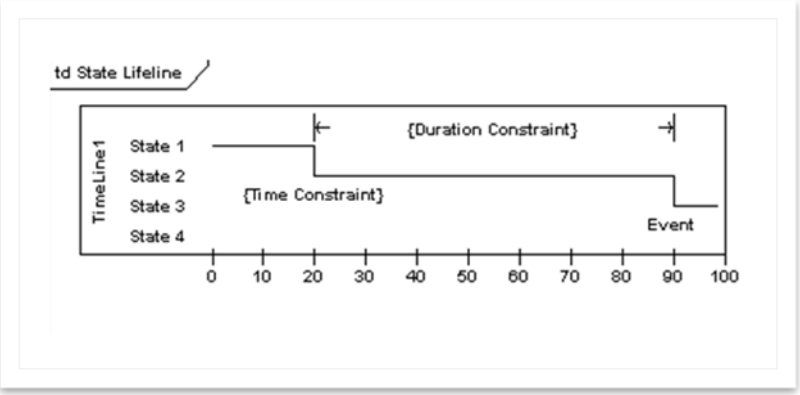


Ilustración 2-11: Ejemplo de la línea de vida del estado de un diagrama de tiempos. [5]

- *Línea de vida del valor*

  - Muestra el cambio del valor de un ítem en el tiempo. De manera horizontal se puede observar el tiempo transcurrido en cualquier unidad que se elija, al igual que para la línea de vida del estado. El valor se muestra entre el par de líneas horizontales que se cruzan en cada cambio del valor.

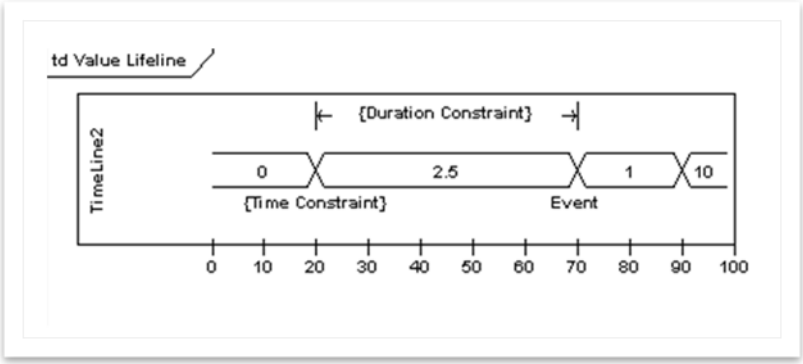


Ilustración 2-12: Ejemplo de línea de vida del valor de un diagrama de tiempos [5]

Las líneas de vida y del estado se pueden ubicar una arriba de otro en cualquier combinación. Los mensajes se pueden pasar de una línea de vida a otra. Ambos gráficos pueden combinarse para brindar una mejor comprensión e información.

❖ Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0)

"Es una variante del Diagrama de Actividad, muestra un panorama general del flujo de control dentro del sistema o proceso de negocio." 2

Un diagrama global de interacciones combina elementos de los diagramas de actividad con los diagramas de secuencia para mostrar el flujo de ejecución del programa.

Los modelos de interacción pueden llegar a ser muy grandes para sistemas complejos. Por lo que si el número de líneas de vida participantes y el número de mensajes intercambiados excede un cierto parámetro, se puede "modularizar" las interacciones y dividir en partes pequeñas y manejables, de acuerdo a principios universales del diseño de sistemas, que también pueden ser visualizadas con la ayuda de otros diagramas como los diagramas de secuencias.

[2] http://www.jtentor.com.ar/post/UML-Diagramas.aspx

2.5 ISO 12207

2.5.1 Introducción

ISO/IEC 12207 es una norma que establece un marco de referencia para el proceso de ciclo de vida de software que incluye las actividades y tareas que se llevaran a cabo desde la definición y comprensión de requerimientos hasta la finalización de su uso.

El propósito principal de este estándar brindar una visión general y común tanto para desarrolladores, personal de mantenimiento, personas próximas a adquirir un sistema y que de esta forma estos puedan usar un lenguaje común.
La norma ISO 12207 puede ser usada por una organización dependiendo de las necesidades utilizando únicamente un subconjunto de actividades que se acoplen a ellas.

2.5.2 Limitaciones

Se deberá tener en cuenta que la norma describe los procesos, actividades y tareas que se llevarán a cabo, pero no especifica el cómo realizarlas.

La norma no establece un modelo de ciclo de vida concreto sino más bien una guía donde serán las o la parte involucrada quien decida y sea responsable de seleccionar y aplicar los métodos de desarrollo de software y realizar actividades y tareas que se consideren necesarias.

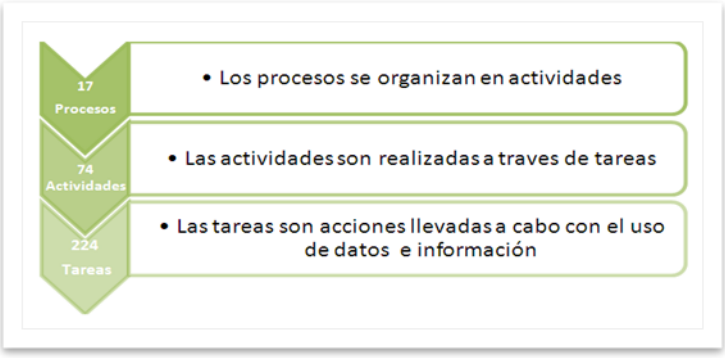


Ilustración 2-13: Procesos, actividades y tareas de la norma ISO 12207. [B]

Las actividades se agrupan en cinco procesos principales, ocho procesos de soporte, cuatro procesos organizativos, y un proceso de adaptación.

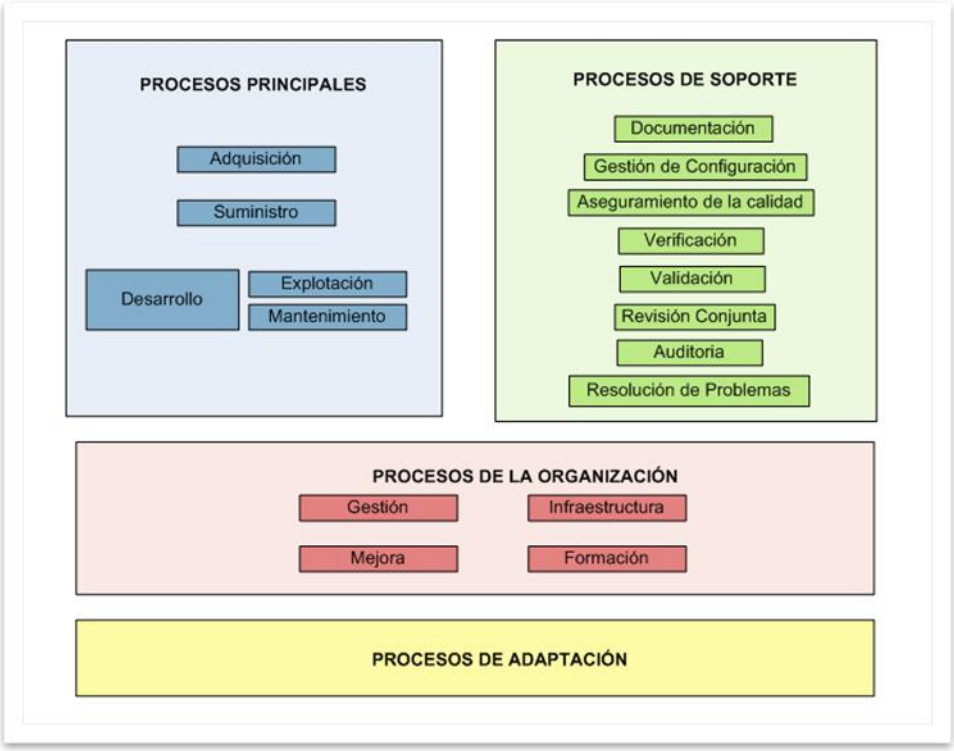


Ilustración 2-14: Procesos principales, de apoyo y de la organización de la Norma ISO/IEC 12207. [6]

2.5.3 Procesos principales

2.5.3.1 Adquisición

Contiene actividades y tareas del usuario que está dispuesto a adquirir un sistema. Empezando por la definición de necesidades por las que el usuario adquiere el sistema, posteriormente existe una preparación y publicación de propuestas, donde el adquiriente seleccionará al proveedor.

❖ Actividades:

- Inicio.
- Preparación de la solicitud de Propuestas.
- Preparación y actualización del contrato.
- Seguimiento del proveedor.
- Aceptación y finalización.

2.5.3.2 Suministro

Este proceso está enfocado a las actividades y tareas llevadas a cabo por el proveedor, empezando desde la creación de una propuesta o la firma de contrato con el adquiriente, posteriormente se procederá a determinar recursos necesarios, planificación ejecución de planes hasta la entrega al usuario del producto o servicio.

❖ Actividades:

- Inicio.
- Preparación de la respuesta.
- Contrato.
- Planificación.
- Ejecución y control.
- Revisión y evaluación
- Entrega y finalización.

2.5.3.3 Desarrollo

Este proceso está compuesto por las actividades de análisis de requerimientos, diseño, codificación, integración pruebas, instalación y aceptación relacionadas con productos de software.

❖ Actividades:

- Implementación del proceso.
- Análisis de los requerimientos del sistema.
- Diseño de la arquitectura del sistema.
- Análisis de los requerimientos software.
- Diseño de la arquitectura del sistema.
- Diseño detallado del software.
- Codificación y pruebas del software.
- Integración del software.
- Pruebas de calificación del software.
- Integración del sistema.
- Pruebas de calificación del sistema.

- Instalación del software.
- Apoyo a la aceptación del software.

2.5.3.4 Mantenimiento

Este proceso se da cuando existe la necesidad de modificar el producto de software ya sea por problemas o por la necesidad de mejorarlo.

- Implementación del proceso.
- Análisis de problemas y modificaciones.
- Implementación de las modificaciones.
- Revisión/ aceptación del mantenimiento.
- Migración.
- Retirada del software.

2.5.4 Procesos de apoyo

2.5.4.1 Documentación

Proceso que consiste en el registro de la documentación existente de un proceso o actividad del ciclo de vida. Incluyendo actividades para planificar, diseñar, desarrollar, producir, modificar, siendo esta documentación necesaria para los involucrados como gerentes, proveedores, usuarios.

❖ Actividades:

- Implementación del proceso.
- Diseño y desarrollo.
- Producción.
- Mantenimiento.

2.5.4.2 Verificación

Este proceso consiste como su nombre lo indica verificaciones o revisiones tanto del contrato o convenio según sea el caso, procesos, requerimientos, diseño, código, integración, documentación teniendo en cuenta criterios definidos para cada actividad.

- Implementación del proceso.
- Verificación.

2.5.4.3 Validación

Este proceso determina si el producto de software cumple con el uso para el cual fue desarrollado. Este proceso podrá ser llevado por el personal que desarrollo el producto o por terceros, esto dependerá del grado de independencia que se requiera en este proceso, si este es el caso el proceso será denominado validación independiente.

- Implementación del proceso.
- Validación.

2.5.4.4 Revisión conjunta

Este proceso puede ser llevado por cualquiera de las dos partes donde la una revisa a la otra, y se revisará el estado y los productos de una actividad del proyecto.

- Implementación del proceso.
- Revisiones de la gestión del proyecto.
- Revisiones técnicas.

2.5.4.5 Auditoría

Puede ser ejecutado por cualquiera de las dos partes, donde se verifica el cumplimiento con requerimientos planes y contrato.

❖ Actividades:

- Implementación del proceso.
- Auditoría.

2.5.4.6 Solución de Problemas

El propósito de este proceso es analizar y resolver problemas detectados durante la ejecución de los procesos de desarrollo, operación mantenimiento, documentando y asegurándose que el problema será resuelto

- Implementación del proceso.
- Solución de problemas.

2.5.5 Procesos de la organización

Los procesos organizativos se enfocan en las personas responsables de cada proceso que serán quienes los usaran.

2.5.5.1 Gestión

Administra actividades y tareas que pueden ser empleadas por cualquier otro proceso.

❖ Actividades:

- Inicio y definición del alcance.
- Planificación.
- Ejecución y control.
- Revisión y evaluación.
- Finalización.

2.5.5.2 Infraestructura

Este proceso define las actividades para establecer y mantener las infraestructura (hardware, software, estándar, herramientas, etc.) necesaria por otros procesos.

❖ Actividades:

- Implementación del Proceso.
- Establecimiento de la Infraestructura.
- Mantenimiento de la Infraestructura.

2.5.5.3 Mejora de proceso

Provee de actividades para establecer, evaluar, medir, controlar y mejorar un proceso de ciclo de vida del software.

❖ Actividades:

- Establecimiento del proceso.
- Evaluación del proceso.
- Mejora del Proceso.

2.5.5.4 Recursos humanos

Es un proceso para preparar y mantener capacitado al personal, puesto que es necesario que los desarrolladores de la aplicación conozcan del tema, para una correcta realización de la aplicación y posterior implementación, suministro, operación y mantenimiento.

❖ Actividades:

- Implementación del Procesos.
- Desarrollo del Material de Formación.
- Implementación del Plan de Formación.

2.6 Modelos de ciclo de vida de desarrollo de software

2.6.1 Definición de ciclo de vida

El ciclo de vida de desarrollo de software se puede definir como un marco de trabajo usado para estructurar, planificar y controlar un proyecto de desarrollo, que permite llevarlo a cabo con altas posibilidades de éxito. El ciclo de trabajo determina el orden de las etapas involucradas describiendo las fases principales de desarrollo de software ayudando a administrar el progreso del desarrollo.

Los modelos brindan una guía que ayuda a ordenar las diversas actividades en el proyecto, permiten estimar recursos, registrar y controlar tiempos de las diferentes actividades

realizadas y las actividades que todavía están pendientes, permitiendo así llevar un seguimiento del avance del desarrollo del software.

El ciclo de vida del software describe el desarrollo de software de manera ordenada, definiendo el estado de las fases por las que un proyecto atravesará desde la fase inicial hasta la fase final.

2.6.2 Tipos de ciclo de vida

Un modelo de ciclo de vida de desarrollo del software describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito es definir las distintas fases que se requieren para validar el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo.

Existen varios modelos de ciclo de vida de software donde se especifican las actividades que se realizan durante el desarrollo de éste, entre los más conocidos:

- Ciclo de vida lineal.
- Ciclo de vida en cascada puro.
- Ciclo de vida por prototipos.
- Ciclo de vida en espiral.
- Ciclo de vida incremental.
- Ciclo de vida evolutivo.

2.6.2.1 Ciclo de vida lineal

Este es el modelo más sencillo, consiste en la descomposición de todas las etapas de un proyecto de desarrollo y ejecutarlas de manera lineal, sin ningún tipo de iteración, es decir la realización de cada una de las etapas una seguida de la anterior.

Es importante recalcar que las etapas son independientes entre sí. Este modelo es el más sencillo en cuanto a la planificación y predicción de tiempos.

Se debe considerar que en cada etapa se conozca de manera precisa su contenido minimizando las posibilidades de cometer errores, y evitando al mínimo la necesidad de requerir información del cliente o usuario final.

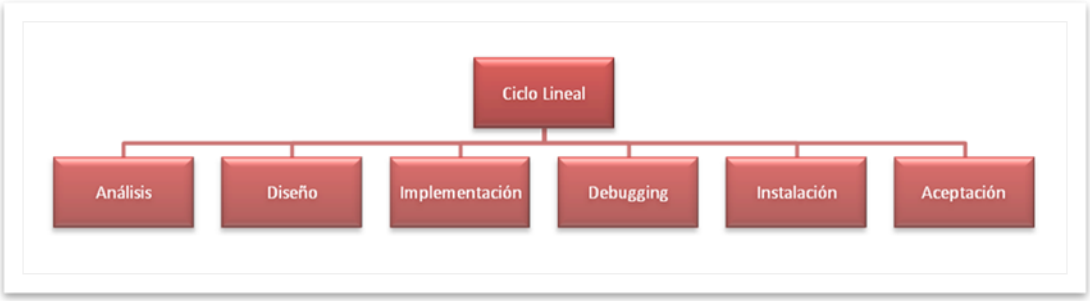


Ilustración 2-15: Fases del modelo de ciclo de vida lineal. [A]

2.6.2.2 Ciclo de vida en cascada puro

Sigue de manera secuencial etapas o fases del ciclo de vida del software, el inicio de cada etapa debe esperar la finalización de la inmediatamente anterior. De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo. Sus fases son definidas, no se podrá empezar una fase sin antes haber terminar la anterior.

Este modelo permite la iteración de sus fases es decir para poder corregir un error en una fase anterior a la que en ese momento el grupo de desarrollo se encuentra será necesario terminar todas las etapas del ciclo y empezar de nuevo desde la primera hasta llegar efectivamente a la etapa donde se encontró el error, para posteriormente continuar alterando en las siguientes etapas, según sea necesario, para que la corrección de dicho error sea modificado en las etapas siguientes.

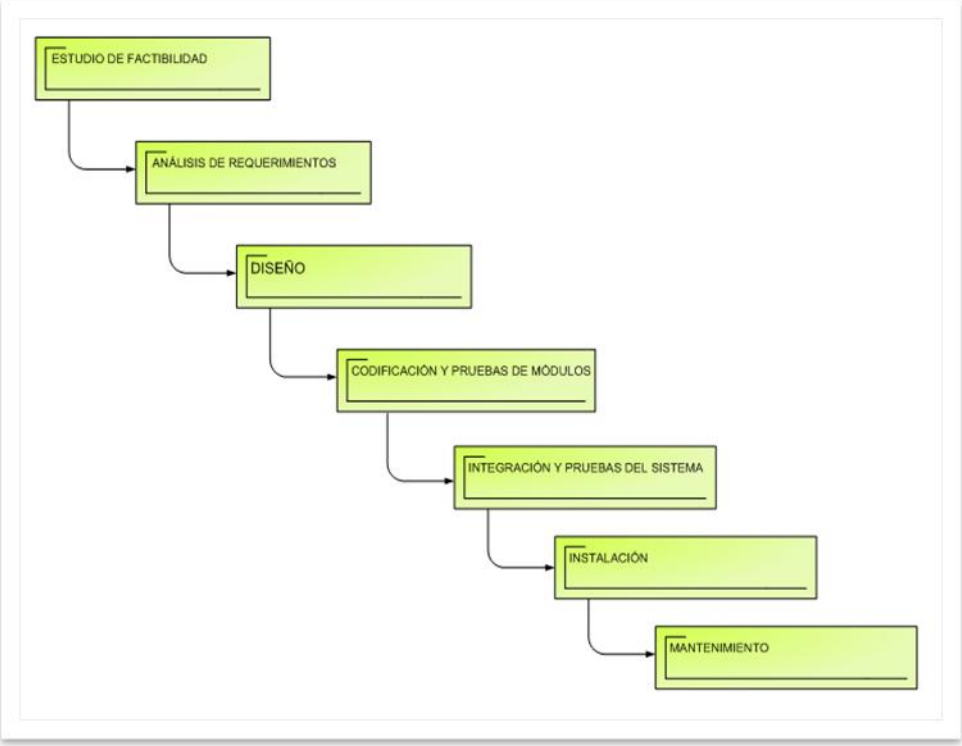


Ilustración 2-16: Fases del modelo de ciclo de vida en cascada puro. [A]

El ciclo de vida en cascada tiene diversas variaciones entre las que destacan:

- Ciclo de vida en cascada con retroalimentación.
- Ciclo de vida con componentes.
- Ciclo de vida en V.
- Ciclo de vida Sashimi.
- Ciclo de vida en cascada con subproyectos.
- Ciclo de vida iterativo.

2.6.2.3 Ciclo de vida por prototipos

Es un marco de trabajo que consiste en el diseño rápido de aspectos del software visibles para el cliente en un prototipo que será entregado como una primera versión al usuario quien podrá evaluar y opinar; realizando una retroalimentación de los requerimientos de, esta manera el desarrollador podrá conocer de mejor y de manera más detallada las necesidades del usuario y al mismo tiempo puede observar resultados a corto plazo. El desarrollador irá presentando versiones cada vez más completas refinando los requisitos hasta lograr satisfacer al cliente.

Es útil usar este modelo cuando el usuario no ha identificado todos los requerimientos del sistema ha implementarse y a través de las diversas versiones del producto podrá irlas definiendo; ayudando tanto al desarrollador de software como al cliente a entender cuál será el resultado cuando los requisitos sean satisfechos.

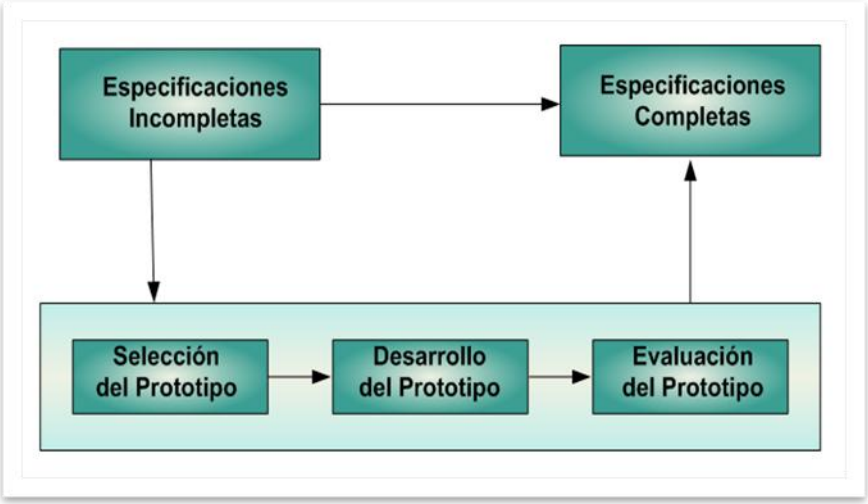


Ilustración 2-17: Fases del modelo de ciclo de vida por prototipos. [7]

2.6.2.4 Modelo de ciclo de vida en espiral

Este modelo de ciclo de vida comprende cuatro fases, que están representadas cada una por un cuadrante del plano cartesiano, se deberá atravesar dichas fases, y se podrá obtener un primer producto. El desarrollador puede repetir muchas veces el ciclo en forma espiral obteniendo como resultado prototipos que irán cumpliendo con los requerimientos del cliente. Se debe considerar que este modelo maneja un alto riesgo al desconocer todos los requerimientos del usuario que se conocerán en el desarrollo del producto.

Se detalla las actividades que se realizan en cada cuadrante:

- Etapa del que: es decir donde se determinarán los objetivos, las alternativas para el desarrollo del producto.
- Etapa del cómo: se evalúan las alternativas enumeradas en la etapa anterior de cómo lograr cumplir con todos los requerimientos enumerados por el cliente.
- Etapa de acción: se lleva a cabo la ejecución y el desarrollo del producto.
- Etapa de mejoramiento: se evalúa los resultados obtenidos en las etapas anteriores y se verifica si es necesario mejorar el producto resultante y si este cumple o no con los requerimientos si no lo hace se planificará como llevar a cabo el siguiente ciclo.

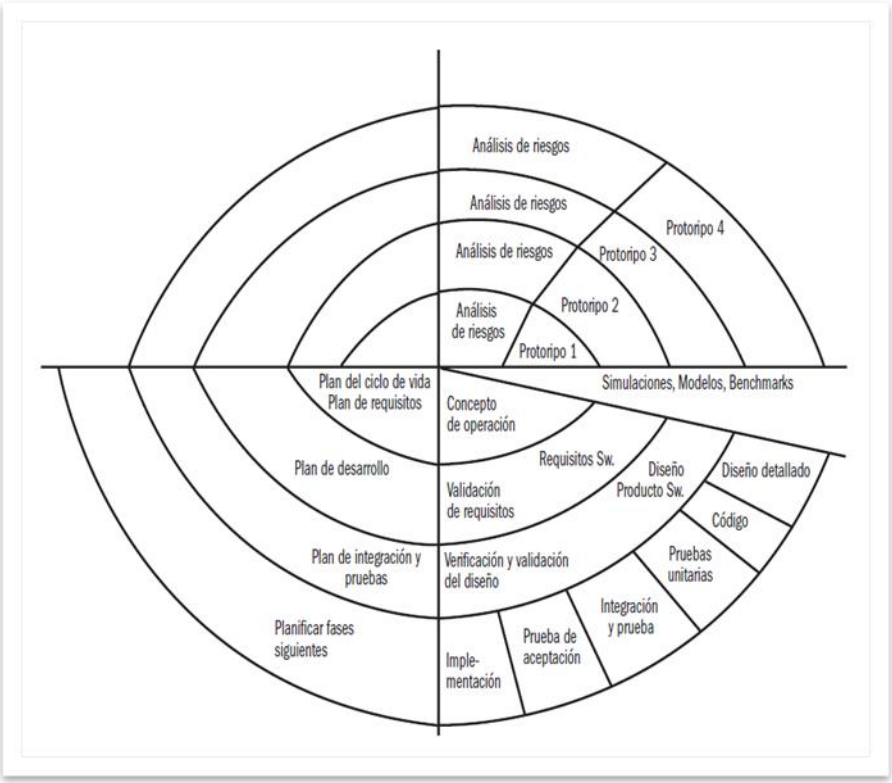


Ilustración 2-18: Modelo de ciclo de vida en espiral [8]

2.6.2.5 Ciclo de vida incremental

Este modelo trata de aplicar el ciclo de vida en cascada y el ciclo de vida por prototipos puesto que utiliza una serie de mini-cascadas, donde cada una de ellas es un módulo del sistema. De esta forma divide un sistema en módulos pequeños que pueden ser desarrollados por diferentes programadores. Cada módulo implementado cumple con las diferentes funcionalidades del sistema, incrementando gradualmente las funcionalidades del programa, pudiéndose realizar una entrega del sistema al cliente antes de terminarlo. Si es detectado el error simplemente se anulará la ultima iteración.

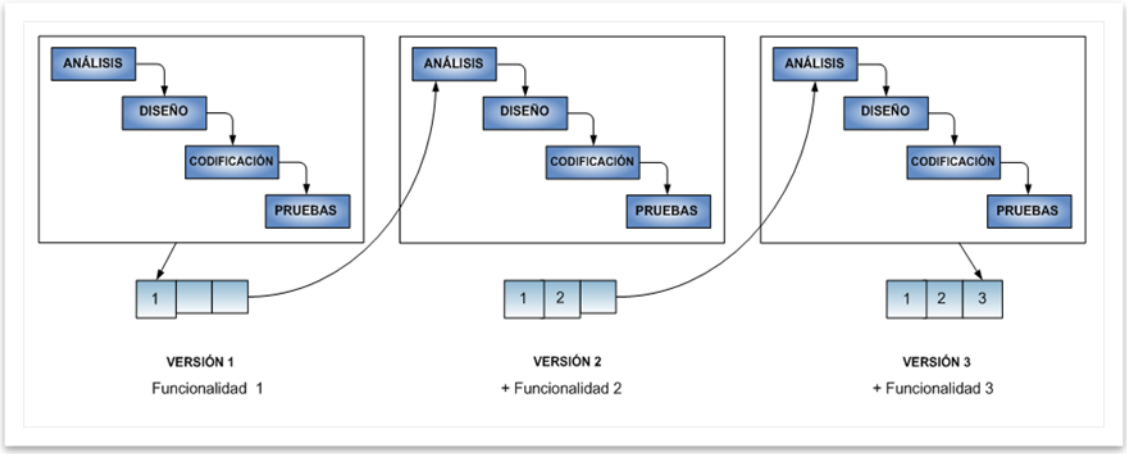


Ilustración 2-19: Fases del modelo de ciclo de vida incremental [9]

2.6.2.6 Modelo de ciclo de vida evolutivo

Este modelo da la posibilidad de que el usuario pueda opinar y de esta forma ser flexible a cambios que se den en el sistema conforme se va desarrollando. Los requisitos centrales son descritos por el cliente aunque para los desarrolladores todavía no se encuentren bien definidos, mientras se va dando el proceso de desarrollo se irá completando dichos requerimientos que se podrán observar en versiones siguientes.

2.7 Cuadros comparativos en el proceso de desarrollo de software

2.7.1 Comparación entre metodología ágil y tradicional

El desarrollo de software de calidad depende de un gran número de actividades y etapas, donde la elección de la metodología para la realización de un determinado proyecto es parte trascendental para el éxito del producto. Por lo que dependerá el tipo de proyecto a llevar a cabo para escoger una u otra metodología.

Se realizó una tabla comparativa para poder escoger el tipo de metodología que se adapte de mejor manera.

Metodologías ágiles	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Preparados para cambios durante el proyecto.	Cierta resistencia a cambios.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.
Pocos roles.	Mayor número de roles.
Grupos pequeños trabajando en el mismo lugar.	Grupos con mayor número de personas y posiblemente distribuidos.

Tabla 2-2: Comparación entre metodología tradicional y ágil. [A]

2.7.2 Resumen de los procesos de la norma ISO 12207 utilizados

2.7.2.1 Procesos principales

Proceso	Definición	Usado	Razón
Adquisición	Contiene actividades y tareas del usuario que está dispuesto a adquirir un sistema. Empezando por la definición de necesidades por las que el usuario adquiere el sistema, posteriormente existe una preparación y publicación de propuestas, donde el adquiriente seleccionará al proveedor.	NO	Este proceso no será aplicado puesto que la disertación no está enfocada en las actividades que realizará el usuario.

Proceso	Definición	Usado	Razón
Suministro	Este proceso está enfocado a las actividades y tareas llevadas a cabo por el proveedor, empezando desde la creación de una propuesta o la firma de contrato con el adquiriente, posteriormente se procederá a determinar recursos necesarios, planificación ejecución de planes hasta la entrega al usuario del producto o servicio.	NO	No aplica al no ser necesario utilizarlo al no tener que realizar una propuesta al usuario.
Desarrollo	Este proceso está compuesto por las actividades de análisis de requerimientos, diseño, codificación, integración pruebas e instalación y aceptación relacionadas con productos de software.	SI	Aplica al describir con mayor detalle las actividades y tareas que lo conforman.
Mantenimiento	Este proceso se da cuando existe la necesidad de modificar el producto de software ya sea por problemas o por la necesidad de mejorarlo.	NO	No aplica la actual disertación trata específicamente del desarrollo del proceso de dimensionamiento.

Tabla 2-3: Procesos principales utilizados de la norma ISO 12207. [A]

- Detalle de las actividades dentro del proceso de desarrollo

  - *Implementación del proceso*

    - *Selección del Modelo de Ciclo de vida*

      El desarrollador deberá definir un modelo de ciclo de vida que se adapte a las necesidades de la aplicación a ser desarrollada, analizando las características de este como complejidad, tiempo, número de personas involucradas.

"Se deberán seleccionar las actividades y tareas del proceso de desarrollo y establecer una correspondencia entre dichas tareas y el modelo de ciclo de vida"

- *Documentar*

  El desarrollador deberá documentar los resultados obtenidos siguiendo los procesos de documentación.

  De igual manera y si es el caso tomar en cuenta el proceso de la gestión de la configuración seleccionando una línea base y ejecutar el control de cambios de acuerdo a dicho proceso.

  Adicionalmente es importante documentar los problemas encontrados en el software y la solución que se encontró.

- *Selección de Herramientas para el desarrollo*

  El o los desarrolladores deberán elegir de manera adecuada las herramientas, lenguaje de programación, IDE, gestor de base apropiado para el desarrollo de la aplicación previo análisis de las características para tomar la decisión más adecuada en cuanto a las funcionalidades y bondades que ofrezcan las herramientas a utilizarse.

- *Planificación de Actividades*

  Parte fundamental del desarrollo de una aplicación es la planificación organizada de las tareas a llevarse a cabo, definiendo tiempos, responsabilidades y roles específicos para cada miembro del equipo de desarrollo.

- *Elementos Entregables y no Entregables*

  Es importante considerar que existe documentación que puede ser importante para el o los desarrolladores pero no para el usuario final, tomando en cuenta que dicha documentación es independiente de esta documentación, y si no lo es se deberán considerar entregables.

- *Análisis de los requerimientos del sistema*

  Consiste en la analizar los requerimientos y determinar si son viables y susceptibles de cumplirse.

- *Especificación de requerimientos del Sistema*

Es primordial conocer las funciones previstas que el nuevo sistema tendrá. La especificación de los requerimientos deberá incluir la descripción de las funciones, interfaces control de accesos. Es muy importante incluir en la documentación tanto las especificaciones de los requerimientos como las limitaciones que el sistema tendrá.

- *Evaluación de los Requerimientos*

  Una vez especificados los requerimientos es importante analizar y evaluar que efectivamente dichos requerimientos podrán ser desarrollados, es necesario documentar los resultados que se obtengan de las evaluaciones entre ellos se deberá incluir la capacidad de ser probados, analizar si es viable el diseño de la arquitectura de la aplicación y si es posible realizar un mantenimiento de esta.

- *Diseño de la arquitectura*

  - *Establecer arquitectura del Sistema*

    Es importante identificar los elementos tanto de hardware como de software y la configuración de estos. Es importante documentar la arquitectura del sistema y los requerimientos asignados.

  - *Evaluación de la Arquitectura*

    Al igual que la especificación de los requerimientos la arquitectura del sistema también deberá ser evaluada donde se deberá documentar la viabilidad de los elementos de software para poder cumplir con los requerimientos.

- *Análisis de los requerimientos de Software*

  Se debe tomar en cuenta los siguientes aspectos para documentarlos debidamente.

  - Funciones y capacidades del software.
  - Interfaces externas.
  - Requerimientos de calificación.
  - Especificaciones de seguridad tanto físicas como de acceso.
  - Especificaciones y requerimientos de bases de datos, instalación y aceptación del producto.
  - Documentación para el usuario.
  - Adicionalmente se deberá analizar y evaluar los requerimientos tomando en consideración la capacidad para realizar el seguimiento de los requerimientos y verificación de que existe una consistencia interna y externa. Como también que los requerimientos puedan ser probados. Y que debe haber posibilidad de realizar mantenimiento de la aplicación.

- *Diseño detallado del software*

  El o los desarrolladores deberán crear un diseño detallado para los componentes de software, se deberá refinar los componentes.

  Es necesario preparar y documentar el diseño detallado de:

  - Interfaces externas al software y entre los componentes.
  - Base de datos.
  - Adicionalmente el o los desarrolladores deberán documentar los requerimientos de prueba y planificar pruebas de las unidades.

  - *Evaluación*

    Es necesario también evaluar el diseño detallado del software teniendo en cuenta los criterios de viabilidad en la realización de pruebas operaciones y

mantenimiento, y de consistencia con el diseño de la arquitectura entre componentes de software.

- *Codificación y pruebas del software*

  - *Documentar*

    El o los desarrolladores deberán documentar:

    Las unidades de software y base de datos.
    Procedimientos que serán utilizados para probar las unidades de software y base de datos.

  - *Satisfacer Requerimientos*

    Los procedimientos antes de prueba deberán ser realizados para verificar que se satisfacen los requerimientos. Es necesario documentar los resultados obtenidos.

  - *Actualización*

    Si es el caso se deberá actualizar:

    - Documentos entregables al usuario.
    - Requerimientos de prueba y plan para integración.

  - *Evaluación*

    Es necesario evaluar el código de software y los resultados de las pruebas realizadas tomando en cuenta los criterios que se enumeran a continuación.

    - Viabilidad de la integración del software y de la realización de pruebas.
    - Viabilidad de realizar mantenimiento del software:

      - Consistencia externa: Requerimientos y el diseño.
      - Consistencia interna: Interna entre los requerimientos.

- *Integración del software*

  - *Desarrollo Plan de Integración*

    Realizar un plan para la integración del software mediante la unión de los elementos que lo componen. Dicho plan deberá incluir requerimientos de prueba, procedimientos, responsabilidades y plazos en los que deberán ser ejecutados.

  - *Integrar las unidades*

    Integrar los componentes de software y probarlos de acuerdo al plan de integración, asegurándose de que efectivamente los requerimientos del software sean satisfechos. Se deberá documentar los resultados tanto de la integración como de los resultados de las pruebas.

    Las pruebas realizadas deberán comprender casos de pruebas, procedimientos que serán calificados según si estas pruebas cumplen o no con los requerimientos, el desarrollador deberá asegurar que el sistema se encuentra listo para llevar a cabo esta tarea.

  - *Evaluación*

    Evaluar el plan de integración, el diseño, el código y las pruebas, los resultados obtenidos considerando los siguientes criterios:

    - Consistencia externa con los requerimientos.
    - Consistencia interna con los componentes del software.
    - Conformidad con los resultados esperados.
    - Viabilidad de las pruebas.

- *Pruebas de calificación de software*

  - *Calificación de Pruebas*

    El software será calificado dependiendo si este cumple o no a conformidad con los requerimientos, es necesario documentar los resultados obtenidos.

- *Actualización*

  Si es el caso se deberá actualizar la documentación entregable al usuario.

- *Evaluación*

  Evaluar el plan de integración, el diseño, el código y las pruebas, los resultados obtenidos considerando los siguientes criterios:

  - Alcance de las pruebas de los requerimientos.
  - Aprobación con los resultados esperados.
  - Viabilidad de la integración del sistema y de las pruebas.

- *Instalación del software*

  - *Plan de Instalación*

    Es necesario preparar un plan para llevar a cabo el proceso de instalación del software y documentarlo, en la planificación se determinarán los recursos e información para realizar esta actividad.

    Posterior a la creación al plan de instalación se deberá ponerlo en marcha, es necesario también documentar incidencias y los resultados obtenidos en la instalación.

  - *Apoyo a la aceptación del software*

    Es importante que el o los desarrolladores brinden ayuda y realicen un seguimiento a las revisiones y pruebas de aceptación llevadas a cabo por el usuario. Dichas revisiones se apoyaran también en otros procesos como revisiones conjuntas, auditorias, pruebas de calificación de software.

    Se deberá proporcionar formación inicial y apoyo al usuario según términos acordados al inicio del proyecto.

2.7.2.2 Procesos de apoyo

Proceso	Definición	Usado	Razón
Documentación	Proceso que consiste en el registro de la documentación existente de un proceso o actividad del ciclo de vida. Incluyendo actividades para planificar, diseñar, desarrollar, producir, modificar, siendo esta documentación necesaria para los involucrados como gerentes, proveedores, usuarios.	SI	Al ser necesario evidenciar y describir cada uno de los procesos llevados a cabo.
Gestión de la Configuración	Aplicación de técnicas y procedimientos para definir la línea base desde donde será el punto de partida para el control de modificaciones como la manipulación y entrega del producto de software.	SI	Este proceso será descrito con mayor detalle si es el caso para la realización de modificaciones.
Aseguramiento de la Calidad	Este proceso asegura que efectivamente el producto de software cumpla con los requerimientos descritos por el usuario. El aseguramiento de la calidad puede ser interno o externo y puede tomar	SI	Este proceso será descrito detallado y llevado a cabo puesto que es muy importante la calidad del software desarrollado por lo que serán realizadas todas las tareas y

Proceso	Definición	Usado	Razón
	resultados de otros procesos como verificación, validación, revisión conjunta, auditoria y solución de problemas.	SI	actividades que lo conforman.
Verificación	Este proceso consiste como su nombre lo indica verificaciones o revisiones tanto del contrato, procesos, requerimientos, diseño, código, integración, documentación teniendo en cuenta criterios definidos para cada actividad.	SI	Será realizado este proceso puesto que parte fundamental del desarrollo de la actual disertación es la verificación de su contenido.
Validación	Este proceso determina si el producto de software cumple con el uso para el cual fue desarrollado.	SI	Posterior al proceso de verificación será necesario describir de manera detallada actividades a realizarse para posteriormente ejecutar dicho proceso.
Revisión Conjunta	Este proceso puede ser llevado por cualquiera de las dos partes donde la una revisa a la otra, y se revisará el estado y los productos de una actividad del proyecto.	NO	El usuario no dispone de tiempo suficiente por lo que no será posible ejecutar este proceso
Auditoria	Puede ser ejecutado por cualquiera de las dos partes, donde se verifica el cumplimiento con requerimientos planes y contrato.	NO	La actual disertación no está enfocada en desarrollar ningún tipo de auditoría.

Proceso	Definición	Usado	Razón
Solución de Problemas	El propósito de este proceso es analizar y resolver problemas detectados durante la ejecución de los procesos de desarrollo, operación mantenimiento, documentando y asegurándose que el problema será resuelto.	NO	Al ser un proceso que afecta al proceso de desarrollo por lo que será aplicado si es el caso.

Tabla 2-4: Procesos de apoyo utilizados de la norma ISO 12207 [A]

- Detalle de las actividades dentro del proceso de documentación

  - *Planificación de la documentación*

  Es necesario implementar un plan en el que se incluirán todos los documentos que se van a realizar en el ciclo de vida del producto de Software.

  - *Diseño y desarrollo*

  Establecer normas de Documentación. Cada documento deberá ser diseñado aplicando normas como pies de páginas, figuras, tablas y demás elementos de presentación. Y deberán ser revisados por personal calificado antes de ser presentados, para asegurar que fue aplicado el formato, estilo de presentación y normas de documentación.

  - *Producción*

  La producción y entrega de los documentos se realizará de acuerdo al plan desarrollado anteriormente. Adicionalmente toda la documentación deberá ser almacenada en un lugar seguro.

- *Mantenimiento*

Serán documentadas las tareas llevadas a cabo al realizarla el proceso de Mantenimiento. Los documentos que se realicen en la gestión de la configuración se administrarán en el proceso de gestión de la configuración.

- Detalle de las actividades dentro del proceso de gestión de la configuración

- *Implementación del proceso*

  - *Planificación de Gestión de Configuración*

    Se deberá desarrollar un plan que contenga las actividades de la gestión de configuración, procedimientos a llevar a cabo y plazos en los que deberán ser cumplidos, los responsables de las actividades y la relación existente entre los responsables de realizar actividades como desarrollo y mantenimiento

  - *Identificación de la configuración*

    Para identificar los elementos de software y sus versiones de documentación se establecerá la línea de referencia.

  - *Control de configuración*

    Es importante el registro de las peticiones de cambio, el análisis y evaluación de dichos cambios, la aprobación o rechazo si es el caso y su posterior desarrollo, implementación verificación y release.

    Dicha información podrá ser auditada de tal manera que se pueda conocer todo el proceso que se llevo a cabo para realizar la modificación, las razones para haber modificado el sistema y la autorización para realizar dicha modificación.
    Es necesario controlar y auditar todos los accesos a los elementos de software que manejan funciones críticas.

  - *Determinación del estado de la configuración*

    Deberán generarse registros en los que consten informes del estado de los elementos, el número de cambios realizados, las versiones de cada elemento y

también las líneas de referencia, adicionalmente identificadores de releases, número y comparación entre releases.

- *Evaluación de la configuración*

  Se determina y asegura que el software cumpla su funcionalidad de acuerdo a los requerimientos y también se verifica si el diseño y código cumplen con los estándares planteados en procesos anteriores.

- *Gestión de releases y entrega*

  El release y entrega tanto del software como de la documentación tendrá que ser controlada y entregada formalmente, deberá existir copias maestras del código y de la documentación del ciclo de vida del producto.

❖ Detalle de las actividades dentro del proceso de aseguramiento de la calidad

➢ *Implementación del proceso*

Asegurar que los productos de software y los procesos ejecutados para la elaboración de dichos productos cumplen con los requerimientos establecidos y se encuentran dentro de la planificación

Es recomendable que el proceso de aseguramiento de la calidad sea coordinado con los procesos de verificación, validación y si es el caso también con revisión conjunta y auditoria.

- *Planificación de Tareas del Proceso de aseguramiento de la calidad*

  El plan de aseguramiento de calidad que deberá ser documentado, implementado y ser mantenido durante todo el proyecto. Deberá incluir:

  - Estándares de calidad, metodología, actividades y herramientas necesarias para realizar las actividades de aseguramiento de la calidad.
  - Procedimientos para identificar, recopilar y hacer posible el registro del cumplimiento de los estándares de calidad.

- Apoyarse en actividades y tareas de los procesos de soporte como verificación, validación y si es el caso revisión conjunta, auditoria y solución de problemas.
- Ejecución.

Se deberán ejecutar las actividades y tareas de aseguramiento planificadas. Si se detectan problemas o requerimientos que no satisfacen al usuario estos serán direccionados como entradas al proceso de solución de problemas.

- *Aseguramiento del producto*

En esta tarea se debe asegurar que el producto de software y la documentación cumplan con lo convenido inicialmente.

A la entrega del producto de software se deberá asegurar que se hayan cumplido a cabalidad con los requerimientos detallados en el convenio.

- *Aseguramiento del proceso*

Es importe que todos los procesos utilizados en el desarrollo del ciclo de vida del software como son: procesos principales (desarrollo) y procesos de apoyo (documentación, aseguramiento de la calidad) cumplan con lo convenido tanto las mediciones como el proceso de software se realicen de acuerdo a las normas y procedimientos establecidos.

- Detalle de las actividades dentro del proceso de verificación

- *Implementación del proceso*

  - *Análisis de los Aspectos Críticos de los Requerimientos*

Dichos aspectos se evaluarán en aspectos de:

- Probabilidad de que un requerimiento no detectado cause fracaso del sistema, o causen algún tipo de pérdida como financiera, o aplicando a la disertación insumos para la preparación de comidas.
- Riesgos asociados con la tecnología a usarse.

- Disponibilidad de fondos y recursos.

Se deberá establecer un proceso de verificación para el producto de software, en el que se determinarán el alcance, magnitud y complejidad y aspectos críticos considerando los enumerados anteriormente.

Es necesario determinar qué actividades del ciclo de vida y que productos de software serán verificados.

- *Documentación e Implementación de un Plan de Verificación*

Se deberá definir un plan de verificación que incluya las actividades que serán llevadas a cabo, las actividades del ciclo de vida y productos del software que serán verificados, las tareas que se realizarán para dicha verificación, los responsables y los plazos a ser cumplidos.

Los problemas y requerimientos no satisfechos en la verificación serán enviados a los procesos de solución de problemas.
Se deberá poner a disposición del usuario final y de las organizaciones o autoridades involucradas los resultados de las actividades de la verificación.

- *Verificación*

  - *Verificación del Contrato o Convenio:*

Los criterios con los que se realizan dicha verificación se enumeran a continuación:

- El proveedor está en la capacidad de satisfacer los requerimientos.
- Los requerimientos cubren las necesidades del usuario.
- Se ha considerado los procedimientos y el alcance e interacción entre las partes.
- Se han estipulado criterios y procedimientos de aceptación de los requerimientos.

- *Verificación del Proceso*

  Los criterios con los que se realizan dicha verificación se enumeran a continuación:

  - Los requerimientos para la planificación del proyecto son adecuados y están a su debido tiempo.
  - Los procesos utilizados para el desarrollo del proyecto son los adecuados y se realizan como fueron planificados.
  - El personal que efectúa el proyecto se encuentra capacitado.

- *Verificación de los Requerimientos*

  Los criterios con los que se realizan dicha verificación se enumeran a continuación:

  - Los requerimientos son viables y pueden ser probados.
  - Los requerimientos de software relacionados con seguridades físicas y lógicas son correctos.

- *Verificación del Diseño*

  Los criterios con los que se realizan la verificación del diseño se enumeran a continuación:

  - El diseño es correcto y coherente con los requerimientos.
  - El diseño implementado posee la secuencia correcta de eventos, entradas salidas, interfaces, definición e incluso recuperación ante posibles errores.
  - El diseño implementado cumple correctamente con los requerimientos de seguridad física y de accesos.

- *Verificación del Código*

  Los criterios con los que se realizan la verificación del código se enumeran a continuación:

- El código es trazable hacia el diseño y los requerimientos, se puede verificar que es correcto y cumple con normas establecidas de codificación.
- El código implementado posee una secuencia correcta de eventos, interfaces consistentes, flujos correctos de datos, adecuada asignación de sincronizaciones, tamaños y definiciones.
- El código se puede derivar del diseño o de los requerimientos.
- El código implementado cumple correctamente con los requerimientos de seguridad física y de accesos.

▪ *Verificación de la Integración*

A continuación se enumeran los criterios por los que se verificará la correcta integración

- Los componentes y unidades de software de cada elemento han sido integradas correcta y completamente.
- Los elementos tanto de hardware como de software han sido integradas completa y correctamente en el sistema.
- Las tareas de integración se llevaron a cabo de acuerdo un plan de integración.

▪ *Verificación de la Documentación*

La documentación será verificada teniendo en cuenta los siguientes criterios:

- La documentación se encuentra completa, es adecuada y consistente.
- La preparación de la documentación se realizó a su debido tiempo.
- Se siguen procedimientos especificados para la gestión de la configuración.

➢ *Detalle de las actividades dentro del proceso de validación*

▪ *Implementación del proceso*

Se deberá establecer si el proyecto merece la realización de la validación. Si este es el caso se establecerá el proceso de validación con el que se validará el

producto de software. Adicionalmente se seleccionaran las tareas de validación que incluyen los métodos técnicas y herramientas asociadas.

Se analiza si el proyecto necesita una validación realizada por terceros.

- *Documentación del Plan de Validación*

Se realizará un plan de validación que incluirá:

- Elementos que serán validados.
- Tareas de validación a realizarse.
- Recursos, responsables y plazos para la validación.

Luego de implementar el plan de validación si existen problemas o no conformidades estas serán tratados en el proceso de solución de problemas.

- *Validación*

Se deberá preparar los requerimientos de prueba, casos y especificaciones de prueba para analizar los resultados, es importante tomar en cuenta que los requerimientos de prueba reflejen los requerimientos especificados por el usuario estas deberán incluir.

- Pruebas con sobrecarga.
- Pruebas del software y su capacidad de aislar o minimizar efectos de errores, peticiones de asistencia del operador ante sobrecargas y situaciones límite.
- Pruebas de usuarios representativos que logren de manera efectiva las tareas previstas con el uso de software.
- Validar que el software satisface las necesidades y se usará para lo que fue creado.

2.7.2.3 Procesos de la organización

Proceso	Definición	Usado	Razón
Gestión	Administra actividades y tareas que pueden ser empleadas por cualquier otro proceso.	NO	Al ser necesaria la administración de cualquier proceso.
Infraestructura	Este proceso define las actividades para establecer y mantener las infraestructura (hardware, software, estándar, herramientas, etc.) necesaria por otros procesos.	NO	Es importante tomar en cuenta este proceso para el análisis y establecimiento de Hardware y Software. Sin embargo, este proceso no podrá ser realizado puesto que la infraestructura del cliente ya ha sido establecida.
Mejora del Proceso	Provee de actividades para establecer, evaluar, medir, controlar y mejorar un proceso de ciclo de vida del software.	NO	Este proceso no será analizado a mayor detalle y por ende no será aplicado a la disertación al enfocarse únicamente en el desarrollo y entregada del producto de software y no en su mantenimiento.
Recursos Humanos	Es un proceso para preparar y mantener capacitado al personal, puesto que es necesario que los desarrolladores de la aplicación conozcan del tema, para una correcta realización de la aplicación y posterior	NO	No será realizado este proceso puesto que el grupo de trabajo cuenta con el conocimiento para el desarrollo del producto de software

Proceso	Definición	Usado	Razón
Recursos Humanos	implementación, suministro, operación y mantenimiento.	NO	No será realizado este proceso puesto que el grupo de trabajo cuenta con el conocimiento para el desarrollo del producto de software

Tabla 2-5: Procesos de organización utilizados de la norma ISO 12207. [A]

2.7.3 Comparación entre modelos de ciclos de vida

2.7.3.1 Aspectos importantes para la definición de un ciclo de vida

Para definir un ciclo de vida se debe considerar el alcance del proyecto, es decir, definir un límite hasta que etapa se lo desarrollará. También se analizará la factibilidad de realizarlo, si se lo diseñará, desarrollará o si se le dará mantenimiento.

Otro aspecto a tomar en cuenta es el tiempo con el que los desarrolladores cuentan, y las etapas en las que se piensa dividir al proyecto que se llevará a cabo.

El presupuesto que la empresa o desarrolladores cuenten es un aspecto muy importante a tomarse en cuenta.

Un punto a considerar a la hora de elegir un modelo de ciclo de vida es cuanto se conoce del proyecto que se va a realizar, puesto que el desarrollo de una aplicación conlleva consigo un riesgo, y este variará con el modelo de ciclo de vida seleccionado, puesto que el riesgo existirá en mayor o menor grado, ya que la probabilidad de retomar etapas anteriores dependiendo del ciclo de vida seleccionado puede resultar una actividad muy costosa en términos de tiempo, dinero y esfuerzo.

El ambiente en el que será desarrollada la aplicación es un aspecto muy importante al momento de escoger un modelo de ciclo de vida, ya que se debe considerar el lugar (oficina, cliente, universidad) como también la intervención del cliente en las diferentes etapas de construcción de la aplicación.

En cada una de las etapas de un modelo de ciclo de vida, se pueden establecer una serie de objetivos, tareas y actividades que lo caracterizan. Existen distintos modelos de ciclo de vida, y la elección de un modelo para un determinado tipo de proyecto es realmente importante.

Ciclo de Vida	Ventajas	Desventajas
Cascada Puro	Puede ser apropiado, para proyectos estables con requisitos no cambiantes y donde es posible y probable que los diseñadores predigan totalmente áreas de problema del sistema y produzcan un diseño correcto antes de que empiece la implementación.	Es bastante inusual que un proyecto siga una secuencia lineal.
	Funciona bien para proyectos pequeños donde los requisitos están bien entendidos.	Este modelo es muy restrictivo y no permite movilizarse entre fases.
	Es simple y fácil de usar, siempre y cuando se siga un proceso disciplinado y planificado.	Los resultados y/o mejoras no son visibles progresivamente, el producto se ve cuando ya está finalizado.
	Al ser un modelo rígido es fácil de gestionar ya que cada fase tiene entregables específicos y un proceso de revisión.	Si los clientes cambiaran sus requisitos después de que el diseño está terminado, este diseño deberá ser modificado para acomodarse a los nuevos requisitos, invalidando una buena parte del esfuerzo.
Iterativo	No hace falta que los requisitos estén totalmente definidos al inicio del desarrollo, sino que se pueden ir refinando en cada una de las iteraciones.	Al no ser necesario tener los requisitos definidos desde el principio, puede verse también como un inconveniente ya que pueden surgir problemas relacionados con la arquitectura.

Ciclo de Vida	Ventajas	Desventajas
	Realiza el desarrollo en pequeños ciclos, lo que permite gestionar mejor los riesgos	
	Gestionar mejor las entregas del producto.	
Incremental	Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.	Para el uso de este modelo se requiere una experiencia para definir los incrementos y distribuir en ellos las tareas de forma proporcionada.
	Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos.	Cada fase de una iteración es rígida y no se superponen con otras.
	Este modelo facilita probar y depurar en una iteración más pequeña.	Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido.
	Facilita gestionar los riesgos.	
Espiral	Incorpora objetivos de calidad integra el desarrollo con el mantenimiento	Es un modelo que genera mucho trabajo adicional. Al ser el análisis de riesgos una de las tareas principales exige un alto nivel de experiencia y cierta habilidad en los analistas de riesgos.
	Es posible tener en cuenta mejoras y nuevos requerimientos sin romper con el modelo, ya que el ciclo de vida no es rígido ni estático.	No es un modelo recomendable para proyectos pequeños.
	Se produce software en etapas tempranas del ciclo de vida y suele ser adecuado para proyectos largos.	

Ciclo de Vida	Ventajas	Desventajas
Prototipos	Ofrece visibilidad del producto desde el inicio del ciclo de vida con el primer prototipo.	Este modelo está el hecho de que puede ser un desarrollo lento.
	Permite introducir cambios en las iteraciones siguientes del ciclo.	Se hacen fuertes inversiones en un producto desechable ya que los prototipos se descartan. Esto puede hacer que aumente el coste de desarrollo del producto.
	El prototipo es la representación del buen funcionamiento del producto y ayuda a que el cliente, experimente ayudando a reducir el riesgo de construir productos que no satisfagan las necesidades de los usuarios.	

Tabla 2-6: Ventajas y desventajas de los modelos de ciclo de vida. [A]

2.8 Justificación del proceso de desarrollo de software de la presente disertación

En base a lo expuesto se han seleccionado los siguientes elementos para la presente disertación:

Principio:	ISO 12207
Modelo de ciclo de Vida:	Lineal
Clasificación de metodología:	Tradicional
Enfoque:	Orientado a Objetos
Modelamiento:	UML
	Diagramas de: casos de uso, actividades, clases, secuencia, paquetes.

Tabla 2-7: Selección de principio, metodología, enfoque y modelo de ciclo de vida. [A]

En primera instancia se ha tomado como principio a la norma ISO 12207 la cual es un conjunto de consideraciones generales aplicables al proceso de construcción y manejo de software.

Se ha visto útil el modelo de ciclo de vida lineal ya que es el que mejor se adapta a los requerimientos del usuario especificados en el capítulo cuatro.

Por otro lado se seleccionó la metodología tradicional la cual menciona que durante el proceso de desarrollo se deben seguir normas provenientes de estándares seguidos por el entorno de desarrollo. Finalmente se tomó un enfoque orientado a objetos el cual describe muchas mejoras y ventajas en comparación al enfoque tradicional. Éste enfoque con ayuda del lenguaje unificado de modelado UML permitirá alcanzar el éxito en la producción de la aplicación informática ya que con la ayuda de UML se podrá especificar y modelar el diseño del proceso recreando un simulacro del sistema.

CAPITULO 3
RECURSOS TECNOLÓGICOS

3. RECURSOS TECNOLÓGICOS

Una vez que el ingeniero de software ha identificado los pasos a seguir dentro del proceso de desarrollo de software es hora de seleccionar las herramientas tecnológicas que le permitirá generar el sistema informático.

Los recursos tecnológicos son un conjunto de elementos que están disponibles y facilitan al desarrollador a alcanzar sus objetivos. Entre las herramientas útiles que se deben considerar están: la herramienta de modelado, el gestor de la base de datos, el lenguaje de programación y el IDE de desarrollo.

Una herramienta de modelado es un recurso muy importante que permite generar el modelo del sistema. Las herramientas de modelado, permiten crear un "simulacro" del sistema, a bajo costo y riesgo mínimo. A bajo costo puesto que es un conjunto de gráficos y textos que representan el sistema, pero no son el sistema físico real (el cual es más costoso).

Un gestor de base de datos permite la administración de los datos, es decir, el manejo de la información dentro de una organización. El gestor debe mantener los datos claros, ordenados además de cumplir con objetivos básicos en cuanto al nivel de seguridad, independencia, consistencia, tiempo de respuesta y manejo de transacciones.

Por otro lado, tenemos los IDEs de desarrollo que contienen gran cantidad de herramientas que permiten la edición, compilación y depuración de código escrito en determinados lenguajes de programación.

Cabe recalcar que en el mercado existen gran cantidad de herramientas, unas propietarias y otras “libres”, la selección será determinada luego de un análisis de las necesidades del sistema y el presupuesto para el desarrollo del mismo.

3.1 Definición

Los recursos tecnológicos son un medio que se valen de la tecnología para permitir y ayudar a optimizar y agilizar procesos disminuyendo tiempos de respuesta.

3.2 Clasificación

Los recursos tecnológicos se clasifican como específicos y transversales.

3.2.1 Recursos Específicos o Tangibles

Son los recursos que incluyen herramientas, equipos, instrumentos, materiales, máquinas, dispositivos y software específicos necesarios para lograr el propósito técnico establecido.

3.2.2 Recursos transversales

Son de tipo intangible, y pueden ser identificados como capital intelectual (estructural y humano) como información necesaria para procesos técnicos de la organización y conocimiento.

Los recursos transversales comprenden la cadena de valor, unidad estratégica de negocios, empresa y sus componentes por lo que son necesarios para el desarrollo de los procesos que se aplican sobre un sistema.

3.3 Herramienta de diseño CASE (Computer Aided Software Engineering)

El desarrollo de un nuevo software requiere que las actividades y tareas sean planificadas, organizadas, y completas en forma correcta y eficiente. Las herramientas CASE fueron desarrolladas para facilitar las tareas de coordinación, automatizando los eventos que necesitan ser mejorados en el ciclo de desarrollo de software incrementando así la velocidad en el desarrollo de los sistemas, permitiendo a los desarrolladores enfocarse en el análisis y diseño para minimizar el tiempo para codificar y probar.

3.3.1 Definición

CASE es el acrónimo de sus siglas en inglés Computer Aided Software Engineering o Ingeniería de Software Asistida por Ordenador.

Las herramientas CASE son aplicaciones informáticas que ayudan en el desarrollo de software proporcionando un conjunto de herramientas que son empleadas en las actividades, técnicas y metodologías, logrando ahorrar tiempo y dinero ya que facilita el trabajo de quienes desarrollarán la aplicación apoyando a una o más fases del ciclo de vida del desarrollo de software.

Los objetivos principales de las herramientas CASE son:

- Mejorar la productividad en el desarrollo del software reduciendo el tiempo y costo de desarrollo y mantenimiento de los sistemas informáticos.
- Automatizar el desarrollo del software, la documentación, la generación de código, las pruebas de errores.
- Automatizar y apoyar una o más fases del ciclo de vida del desarrollo de sistemas.
- Facilitar el uso y la aplicación de una metodología de desarrollo.
- Mejorar la planificación de un proyecto.
- Ayudar a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestionar en todas las fases de desarrollo de software al usarse con una misma herramienta.

3.3.2 Clasificación

Aunque no es fácil y no existe una forma única de clasificarlas, las herramientas CASE se pueden clasificar teniendo en cuenta los siguientes parámetros:

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.

La clasificación tomando como parámetro las fases del ciclo de vida de desarrollo son:

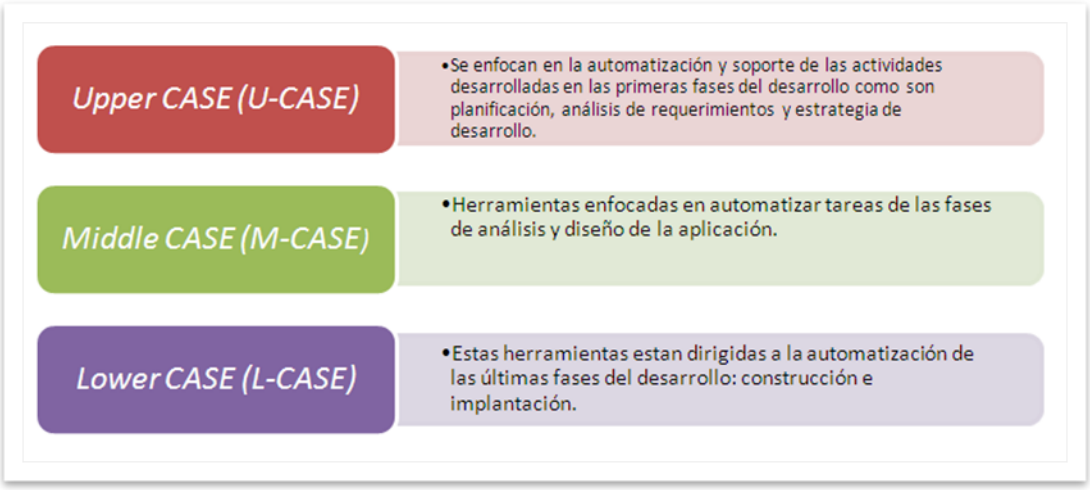


Ilustración 3-1: Clasificación de las herramientas CASE tomando como referencia el ciclo de vida [A]

Existen otros nombres que se le dan a este tipo de herramientas, y que no es una clasificación excluyente entre sí, ni con la clasificación antes descrita.

Ilustración 3-2: Otras herramientas CASE tomando como referencia el ciclo de vida. [A]

Por funcionalidad se diferencian algunas como:

- Herramientas de generación semiautomática de código.
- Editores UML.
- Herramientas de Refactorización de código.

- Herramientas de mantenimiento como los sistemas de control de versiones.

3.3.3 Componentes

Una herramienta CASE se compone de:

- Repositorio o también llamado diccionario donde se almacenan los elementos generados por la herramienta, como por ejemplo esquemas, grafos, matrices, información relativa a la gestión de proyectos, gestión de configuraciones, etc. y cuya gestión se realiza mediante el apoyo de un Sistema de Gestión de Base de Datos (SGBD) o de un sistema de gestión de ficheros.
- Meta modelo (no siempre visible), que constituye el marco para la definición de las técnicas y metodologías soportadas por la herramienta.
- Las herramientas Case dan la posibilidad de cargar el repertorio de la herramienta CASE con datos provenientes de otros sistemas, o generar a partir de la propia herramienta esquemas de base de datos, programas, que pueden, alimentar otros sistemas.
- Comprobación de errores, brindan la facilidad de permitir llevar a cabo un análisis de la exactitud, integridad y consistencia de los esquemas generados por la herramienta.
- Interfaz de usuario, que estará compuesta de editores de texto y herramientas de diseño gráfico que permitan, definir los diagramas, matrices que incluyen las distintas metodologías.

3.3.4 Ejemplos

3.3.4.1 Er/Estudio

Es una herramienta CASE que pertenece a la empresa Embarcadero ER/Studio y sirve para efectuar el modelado lógico y físico de bases de datos Relacionales y Multidimensionales.

ER/Studio incluye características como:

- Hacer una separación entre los modelos lógicos y físicos, pero manteniendo una total integración entre ellos, así como con la base de datos. Es posible generar

múltiples modelos físicos asociados a un mismo modelo lógico, para diferentes plataformas.

- Efectuar ingenierías de reversa de esas bases de datos para documentar en formato .rtf, .xls o bien, HTML, las estructuras de tablas, vistas, índices, objetos físicos, definiciones de seguridad, entre otros.
- Exportar e importar metadatos desde y hacia los principales productos del mercado.

3.3.4.2 Sybase PowerDesigner

PowerDesigner fue creado por Powersoft y posteriormente comprado por Sybase (compañía de informática localizada en California, Estados Unidos).

Power Designer es un conjunto de herramientas de modelamiento que combina distintas técnicas estándar de modelamiento en las que se pueden mencionar a:

- Modelamiento de aplicación a través de UML,
- Técnicas de Modelamiento de Procesos Empresariales y
- Técnicas tradicionales de modelamiento de base de datos.

PowerDesigner ofrece modelamiento de procesos empresariales, incluyendo soporte de simulación y procesamiento ejecutable de procesos de negocio, mayor integración con los ciclos de desarrollo de lenguaje como JAVA, C# y VB.Net y técnicas poderosas de generación inter-modelos, encadenamiento y sincronización.

PowerDesigner incluye características como:

Editor de asociaciones que brinda una vista global de todas las asociaciones definidas en un modelo y le permite crear asociaciones usando una interfaz simple drag and drop.
Permite la creación de DFDs que representen archivos físicos y transacciones y DFDs de negocio, que tienen que ver con flujos lógicos o conceptuales.

3.3.4.3 IBM Rational Data Architect

Es una herramienta de diseño de modelado e integración de datos, construida en la plataforma de código abierto Eclipse, que ayuda y facilita al modelamiento diseño, análisis de bases de datos, como también la comprensión de los elementos de datos y sus relaciones,

de esta forma optimiza los proyectos de base de datos y automatiza la integración de la información.

Rational Data Architect tiene las siguientes características:

- Crea modelos de datos lógicos y físicos.
- Ayuda al descubrimiento exploración visualización y definición de la estructura de las fuentes de datos y sus relaciones.
- Compara la estructura de modelos con modelos o con bases de datos y actualizar posteriormente bases de datos existentes para asegurar la consistencia.
- Analiza modelos y fuentes de datos para verificar la conformidad con los estándares, para lo que se ejecuta contra modelos o bases de datos existentes y verifica la conformidad a los patrones, normas y reglas de la empresa.
- Analiza el impacto ayudando a identificar el impacto de un cambio antes de que este sea implementado. El análisis de impacto consiste en listar las dependencias sobre un elemento específico. Los resultados son presentados de manera visual y desplegada en formato sencillo.

La integración del ciclo de vida brinda ayuda al usuario para integrar las otras fases del ciclo de vida.

3.3.4.4 TOAD Data Modeler

Es una aplicación que permite diseñar esquemas de base de datos y también generar el código SQL necesario para producirlas. Puede soportar cualquier sistema gestor de base de datos como MySQL, Oracle, Access, Firebird, Paradox, Postgre, Sybase y muchos otros.

Toad Data Modeler permite crear diagramas de entidad-relación, definir reglas de integridad referencial, generar scripts SQL que construyan la base de datos o detallados informes en HTML y RTF. Posee una herramienta denominada 'Model Explorer' que permite navegar por todos los atributos del modelo que estemos creando.

Características de Toad Data Modeler:

- Ayuda a la creación de estructuras de alta calidad, generarlas automáticamente, y de manera visual.

- Permite a la generación de reportes detallados para documentar fácilmente las bases de datos existentes.
- Facilita el rediseño puesto que es posible tomar una base de datos, rediseñar el modelo y generar el SQL del nuevo diseño.
- Ayuda a la migración de las estructuras de una base de datos a otra versión o plataforma de manera simple.
- Compara un modelo con una base de datos existente, y genera automáticamente los scripts de "ALTER" necesarios para lograr la sincronización de dos bases de datos (estas características se puede realizar únicamente en el gestor de bases de datos Oracle).
- Genera y hace ingeniería inversa con facilidad.
- Edita modelos, triggers, llaves, índices, tablas, atributos directamente en pantalla en forma interactiva.
- Permite realizar el seguimiento de cambios con un administrador de versiones interno.

3.3.5 Criterios de selección

Seleccionar una herramienta CASE en la actualidad no es una tarea simple considerando el número de aplicaciones existentes ofreciendo siempre mejores prestaciones a los usuarios, por lo que es importante considerar los siguientes puntos:

- Plataforma que soporta la herramienta CASE.
- Lenguaje/s y/o sistema gestor de base de datos al que va orientada.
- Metodología y/o técnicas soportadas.
- Posibilidades de integración con otras plataformas (presentes y futuras).
- Criterios habituales en la selección de software: precio, asistencia técnica, mantenimiento, etc.

3.4 Gestor de bases de datos

Antes de definir que es un Sistema Gestor de Base de Datos (SGDB) será necesario conocer que es una base de datos.

3.4.1 Definición de base de datos

Colección o depósito de datos ordenados relacionados entre sí, capaces de almacenar grandes cantidades de información y que permiten el manejo de forma rápida y eficiente.

3.4.2 Definición de gestor de base de datos

Conjunto de programas que administran y gestionan la información contenida en una base de datos. Permiten realizar la definición, mantenimiento de la integridad de los datos dentro de la base de datos, tener un control de la seguridad y privacidad de los datos como también la manipulación de los datos proporcionando una interfaz entre ellos, los programas que los manejan y los usuarios finales.

Un SGBD debe proporcionar a los usuarios:

- La capacidad de almacenar datos, acceder a ellos y actualizarlos.
- Un catálogo en el que se almacenan las descripciones de los datos y que sea accesible por los usuarios. Este catálogo es denominado diccionario de datos.
- Mecanismos que aseguren que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente.
- La capacidad de recuperar la base de datos en caso de que ocurra algún imprevisto que la dañe llevándola a un estado consistente.
- Protección contra accesos no autorizados, tanto intencionados como accidentales.
- Garantizar la integridad de la base de datos mediante la validez y consistencia de los datos almacenados. Normalmente se expresa mediante restricciones, que son una serie de reglas que la base de datos no puede violar.
- Una serie de herramientas que permitan administrar la base de datos de modo efectivo y facilite el trabajo de los administradores de la base de datos.

3.4.2.1 Diccionario de datos

Es una base de datos donde se guardan todas las propiedades de la base de datos, descripción de la estructura, relaciones entre los datos, etc. El diccionario debe contener:

- La descripción externa, conceptual e interna de la base de datos.
- Las restricciones sobre los datos.

- El acceso a los datos.
- Las descripciones de las cuentas de usuario.
- Los permisos de los usuarios.
- Los esquemas externos de cada programa.
- La información que describe los datos de la base de datos (meta datos).

Normalmente, un diccionario de datos almacena:

- Nombre, tipo y tamaño de los datos.
- Nombre de las relaciones entre los datos.
- Restricciones de integridad sobre los datos.
- Nombre de los usuarios autorizados a acceder a la base de datos.
- Esquemas externos, conceptuales e internos, y correspondencia entre los esquemas.
- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Algunos de los beneficios del diccionario de datos son los siguientes:

- La información sobre los datos se puede almacenar de un modo centralizado. Esto ayuda a mantener el control sobre los datos, como un recurso que son.
- El significado de los datos se puede definir, lo que ayudará a los usuarios a entender el propósito de los mismos.
- La comunicación se simplifica ya que se almacena el significado exacto. El diccionario de datos también puede identificar al usuario o usuarios que poseen los datos o que los acceden.
- Las redundancias y las inconsistencias se pueden identificar más fácilmente ya que los datos están centralizados.
- Se puede tener un historial de los cambios realizados sobre la base de datos.
- El impacto que puede producir un cambio se puede determinar antes de que sea implementado, ya que el diccionario de datos mantiene información sobre cada tipo de dato, todas sus relaciones y todos sus usuarios.
- Permite garantizar y proporcionar la integridad de la información para auditorías.

3.4.2.2 El administrador de la base de datos

Es una persona o grupo de personas responsables del control del sistema gestor de base de datos. Las principales tareas de un administrador son:

- La definición del esquema lógico y físico de la base de datos.
- La definición de las vistas de usuario.
- La asignación y edición de permisos para los usuarios.
- Mantenimiento y seguimiento de la seguridad en la base de datos.
- Mantenimiento general del sistema gestor de base de datos.

3.4.2.3 Los lenguajes

Un sistema gestor de base de datos debe proporcionar una serie de lenguajes para la definición y manipulación de la base de datos. Estos lenguajes son los siguientes:

- Lenguaje de definición de datos (DDL). Para definir los esquemas de la base de datos.
- Lenguaje de manipulación de datos (DML). Para manipular los datos de la base de datos.
- Lenguaje de control de datos (DCL). Para la administración de usuarios y seguridad en la base de datos.

3.4.2.4 Gestores de bases de datos libres y propietarios

Actualmente una las necesidades en las soluciones informáticas incluyendo los Sistemas Gestores de Bases de Datos son la rapidez, efectividad en los procesos y los grandes flujos de información. Ante esta demanda surgieron varios gestores de bases de datos capaces de manejar la información de modo sencillo facilitando servicios para el desarrollo y el manejo de la información que se resguarda.

Con la salida al mercado de múltiples entornos de desarrollo el enfoque está en conocer las características, ventajas y desventajas de cada herramienta, que ofrece el mercado, y para el caso especifico del desarrollo de este trabajo damos a conocer características generales de los productos que más se destacan como: Oracle, Microsoft SQL Server; que

comercialmente son los más fuertes, sin embargo en el mundo del software libre, se aprecian opciones tan completas como MySQL, y PostgreSQL.

Sistemas gestores de base de datos “libres”

Se denominan libres puesto que permiten el acceso a su código fuente, podrá ser modificado y es permitido distribuirlo con los cambios realizados. A demás pueden ser redistribuidos puesto que la copia de éste no constituye un delito.

Entre algunos gestores de bases de datos libres podemos citar:

3.4.2.5 PostgreSQL

Es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.

Su desarrollo comenzó hace más de 15 años, se ha caracterizado por contar con características como estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de oracle), PL/Perl, PL/Python y PL/Tcl.
Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID, XML, matrices, etc.
Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido)
APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt y muchos otros.

3.4.2.6 MySQL

MySQL es uno de los gestores de base de datos de código fuente abierto más usado. Fue desarrollado y proporcionado por MySQL LAB, es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de bases de datos MySQL.

Su origen se dio a la búsqueda por parte de los fundadores de crear un manejador de bases de datos que fuera "rápida".

Se encuentra en desarrollo constante, el servidor MySQL ofrece un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de MySQL un servidor bastante apropiado para acceder a bases de datos en Internet.

3.4.2.7 Sistemas gestores de bases de datos propietarios.

Los sistemas gestores de bases de datos propietarios se caracterizan por ser vendidos por la empresa productora de este, quien distribuye y entrega al comprador una copia del programa ejecutable, junto con la autorización de ejecutar dicho programa en un número determinado de computadoras.

El cliente adquiere únicamente la "licencia" del producto donde se expresa que el cliente simplemente obtiene la facultad de utilizar dicho programa en determinada cantidad de computadoras (dependiendo del monto que haya abonado). La licencia deja en claro que el programa sigue siendo propiedad de la empresa productora del mismo y que el usuario no está facultado a realizar ningún cambio en él.

A continuación mencionamos algunos de los sistemas gestores de base de datos propietarios más conocidos:

3.4.2.8 Oracle

Oracle es una herramienta cliente/servidor para la gestión de base de datos, este SGBD se caracteriza por su gran potencia y su elevado precio, por lo que hace que sea utilizado principalmente por empresas muy grandes y multinacionales.

Oracle se basa en la tecnología cliente / servidor. Necesita para su utilización primero la herramienta servidor y posteriormente se pueden conectar los clientes a la base de datos

desde otros equipos con herramientas de desarrollo como Oracle Designer y Oracle Developer, que son las herramientas de programación sobre Oracle.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

- Soporte de transacciones,
- Estabilidad,
- Escalabilidad y
- Soporte multiplataforma.

La única edición gratuita es la Express Edition, que es compatible con las demás ediciones de Oracle Database 10gR2 y Oracle Database 11g.

3.4.2.9 Sql Server 2000

Sql Server es el sistema de gestión de base de datos representativa de Microsoft. SQL Server 2000 proporciona agilidad a sus operaciones de análisis y administración de datos al permitir a su organización adaptarse rápida y fácilmente para obtener ventaja competitiva en un entorno de cambios constantes.

SQL Server 2000 es un paquete completo de base de datos y análisis de datos que abre las puertas al rápido desarrollo de una nueva generación de aplicaciones comerciales de nivel empresarial.

SQL Server 2000 se ha caracterizado por tener exitosos resultados en pruebas de referencia por su escalabilidad y velocidad. Es un producto de base de datos totalmente habilitado para Web que proporciona una compatibilidad fundamental con el Lenguaje de marcado extensible (XML, Extensible Markup Language) y la capacidad para realizar consultas en Internet por encima del servidor de seguridad.

3.4.2.10 DB2

Es un gestor de base de datos relacional que integra XML de manera nativa, que permite almacenar documentos completos dentro del tipo de datos xml para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.

La automatización es una de sus características más importantes, ya que permite eliminar tareas rutinarias y permitiendo que el almacenamiento de datos sea más ligero, utilizando menos hardware y reduciendo las necesidades de consumo de alimentación y servidores.

La memoria se ajusta y se optimiza el rendimiento del sistema, con un interesante sistema que permite resolver problemas de forma automática e incluso adelantarse a su aparición, configurando automáticamente el sistema y gestión de los valores.

Permite la automatización de tareas, reducción de las necesidades de consumo de alimentación, un alto rendimiento que reduce los servidores necesarios para ejecutar la base de datos, escalabilidad sencilla y alta disponibilidad en su arquitectura de discos de datos y otras soluciones que facilitan la colaboración entre profesionales.

3.4.3 Criterios de selección

La elección de un Sistema de Gestión de Bases de Datos (SGBD) es un punto clave debido a que se debe considerar la importancia de resguardar la información, el presupuesto con que cuenta la organización.

Actualmente las empresas quieren incrementar o mantener su productividad para lo que se deberá gestionar eficientemente todos los datos que son manejados apoyándose en SGBD que se adecuan a las necesidades. Al disponer de varias opciones resulta imprescindible contar con criterios que ayudarán a inclinarse por una u otra solución.

El principal objetivo es encontrar un SGBD que sea capaz de responder adecuadamente al conjunto de aplicaciones y exigencias.

Característica	Justificación
Modelo de datos: relacional, jerárquico o en red.	Es muy importante seleccionar un SGBD con el que se realice modelamientos relacionales
Compatibilidad con las herramientas de ayuda para el desarrollo: lenguajes de cuarta generación, generadores de aplicaciones, asistentes, generadores de informes, CASE, etc.	Es una característica muy importante a tomar en cuenta que la herramienta de modelamiento sea compatible con el sistema gestor de base de datos.

Característica	Justificación
Capacidad de almacenamiento y recuperación de datos: velocidad de ejecución de consultas, creación de índices, importación de datos, etc.	Es un aspecto que para esta disertación no se lo toma como con alta prioridad al tratarse de una aplicación que se encontrará en una pc.
Protección de datos: acceso simultáneo de varios usuarios, etc.	Es un aspecto que para esta disertación no se lo toma como con alta prioridad al tratarse de una aplicación que se encontrará en una pc.
Control de accesos de los usuarios: intentos de acceso de usuarios no autorizados, etc.	Aspecto con prioridad media puesto que al ser usada la aplicación en un solo ordenador se conocerá al responsable de administrarlo.
Consumo de recursos: memoria RAM, etc. (algunos SGBD incorporan monitores que permiten realizar un seguimiento de los recursos consumidos).	Característica de alta prioridad, es necesario tener en cuenta conocer los recursos con los que cuenta el Departamento de Nutrición
Precio de adquisición del producto.	Al tratarse de una disertación el precio de adquisición es un aspecto a considerarse para la elección de SGBD

Tabla 3-1: Criterios para seleccionar un gestor de base de datos. [A]

3.4.3.1 Selección del Sistema gestor de Base de datos

Actualmente las bases de datos libres han alcanzado un nivel de madurez que garantiza su estabilidad y su continuidad a largo plazo, convirtiéndolas en alternativas viables. En la mayoría de los usos tradicionales que se dan a una base de datos relacional se encuentran:

- El diseño del Modelo de datos relacional, jerárquico o en red.
- Compatibilidad con las herramientas de ayuda para el desarrollo.
- Protección de datos y control de accesos de los usuarios.
- Consumo de recursos.
- Precio de adquisición del producto.

3.5 Lenguajes de programación

3.5.1 Programación

La programación orientada a objetos es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. [Ver Capítulo 2, Referencia 2.7]

A continuación se mencionan algunos de los lenguajes de programación orientado a objetos:

Nombre	Descripción	Licencia
Phyton	Es un lenguaje de programación de alto nivel cuya filosofía hace énfasis en una sintaxis muy limpia y que favorezca un código legible. Creado por Python Software Foundation.	Libre
	Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.	
Visual Basic .NET	Es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. Debido a cambios significativos en el lenguaje VB.NET, este no es compatible hacia atrás con Visual Basic.	Propietaria
C#	Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.	Propietaria
	Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET.	

Nombre	Descripción	Licencia
JAVA	Poderoso y seguro lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems. El lenguaje toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores.	Libre

Tabla 3-2: Lenguajes de programación orientada a objetos. [A]

3.6 IDE de desarrollo

3.6.1 Definición

Un entorno de desarrollo integrado IDE (Integrated Development Environment) es un programa informático compuesto por un conjunto de herramientas de programación.
Un IDE es un entorno de programación que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Algunos IDEs soportan múltiples lenguajes de programación como Eclipse o Netbeans.

3.6.2 Componentes

3.6.2.1 Un editor de texto.

Los editores de texto son incluidos en el sistema operativo o en algún paquete de software instalado y se usan cuando se deben crear o modificar archivos de texto como archivos de configuración, scripts o el código fuente de algún programa.

3.6.2.2 Un compilador.

Es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser simplemente texto.

3.6.2.3 Un intérprete.

Es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel. Los intérpretes se diferencian de los compiladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

3.6.2.4 Un depurador.

Es un programa usado para probar y depurar (eliminar los errores) de otros programas.
Los depuradores también ofrecen funciones más sofisticadas tales como correr un programa paso a paso, parar el programa; es decir, pausar el programa para examinar el estado actual en cierto evento o instrucción especificada por medio de un breakpoint, y el seguimiento de valores de algunas variables.

- Posibilidad de ofrecer un sistema de control de versiones.

Una versión es el estado en el que se encuentra un sistema informático en un momento dado de su desarrollo o modificación. Se llama control de versión al manejo de los diversos cambios que se realizan sobre los elementos de la aplicación. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada aplicación desarrolladas.

- Factibilidad para ayuda en la construcción de interfaces gráficas de usuario.

Las interfaces de usuario también conocidas como GUI es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

3.6.3 IDEs de desarrollo “libres”

Entre los IDEs de desarrollo para la programación orientada a objetos más nombrados están: Eclipse y NetBeans.

3.6.3.1 Eclipse

Eclipse es una plataforma, usada para crear entornos integrados de desarrollo de código abierto multiplataforma. Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge.

Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin fines de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Sobre eclipse se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados. La arquitectura de plugins de Eclipse permite, además de integrar diversos lenguajes, introducir otras aplicaciones o accesorios que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces.

Los siguientes componentes constituyen la plataforma de cliente enriquecido de Eclipse:

- Plataforma principal - inicio de Eclipse, ejecución de plugins.
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - Un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto.
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes.

A continuación presentamos los diversos lenguajes de programación utilizados en Eclipse:

Lenguaje	Líneas de código	%
Java	1.911.693	92,66%
ANSI C	133.263	6,46%
C++	10.082	0,49%
JSP	3.613	0,18%
sh	2.066	0,10%
perl	1.468	0,07%
php	896	0,04%
sed	2	0,00%

Tabla 3-3: Lenguajes de programación usados en el IDE Eclipse. [10]

3.6.3.2 Netbeans

NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

El IDE es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas.

Esta escrito completamente en Java usando la plataforma NetBeans. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

Lenguaje	Líneas de código	%
JAVA	1.974.732	99,19%
JSP	7.917	0.40%
Haskell	3.138	0,16%
CPP	1.761	0.09%
Yacc	1.123	0,06%
Sh	1.080	0,05%
Lex	506	0,03%
Perl	350	0,02%
Objc	288	0,01%
Ansic	20	0,00%

Tabla 3-4: Lenguajes de programación usados en el IDE NetBeans. [11]

3.7 Selección de los recursos tecnológicos

A continuación se muestra una tabla con los recursos tecnológicos seleccionados:

Herramienta CASE	Power Designer
Lenguaje de programación	JAVA
Gestor de base de datos	MySQL
IDE	NetBeans

Tabla 3-5: Selección de las herramientas de desarrollo. [A]

La tabla anterior describe una inclinación por lo "libre", ya que debido a la naturaleza del proyecto es mucho más factible el crear la aplicación de software con herramientas no propietarias.

En cuanto al lenguaje de programación, Java es la mejor opción para desarrollar una aplicación robusta y eficaz que se adapte a cualquier sistema operativo. Java tiene gran nivel de aceptación en la mayoría de entornos por lo que nos permitirá alcanzar el desarrollo de un sistema exitoso.

Con ayuda del gestor de base de datos MySQL será posible la elaboración una base de datos robusta que integre toda la información. Además posee gran cantidad de tutoriales, ayudas, blogs ya que al ser "libre" se encuentra disponible en la web.

Finalmente con ayuda de la Herramienta CASE seleccionada podremos generar el modelo del sistema. Se modelará diagramas UML simulando situaciones ideales del manejo de la aplicación. Además podremos generar el modelo lógico y físico de la base de datos el cual contendrá a las entidades y sus relaciones que serán definidas en el Capítulo 4.

CAPITULO IV

DISEÑO DE LA SOLUCIÓN INFORMÁTICA

4. DISEÑO DE LA SOLUCIÓN INFORMÁTICA

Una vez definido el proceso; en el cual intervendría el software, dentro del Departamento de Nutrición del Hospital de Niños “Baca Ortiz” se siguió el proceso de desarrollo de software haciendo uso de las herramientas tecnológicas para finalmente implementar e instalar el sistema informático.

Dentro del proceso de desarrollo de software se estableció un marco de referencia que permitió generar un conjunto de planes necesarios para la correcta elaboración y ejecución de actividades para llevar a cabo la producción del sistema. Con el establecimiento de un ciclo de vida de desarrollo de software se determinó las etapas por las que debía atravesar el diseño del sistema computacional.

En las primeras fases fue necesario analizar los requerimientos del usuario y plasmarlos en diagramas de casos de uso y diagramas de actividades para entenderlos de mejor manera. Posterior a ello, se creó la arquitectura del sistema el cual incluyó: requerimientos técnicos para instalar el sistema, prototipos de interfaz gráfica, diagramas de clases, diagramas de secuencia, el modelo entidad relación de la base de datos, estándares de diseño. Una vez que el “plano del sistema” fue entendido y minuciosamente detallado se procedió con la implementación del mismo, siguiendo una estrategia de desarrollo que cumple con estándares de calidad basándose en: el plan de integración, el plan de verificación y el plan de validación para obtener como resultado un sistema de calidad que fue finalmente instalado y probado en un ambiente real. Además, se elaboró un plan para capacitar a los usuarios finales del sistema quienes a su vez expresaron su conformidad en un documento de aceptación de software.

4.1 Planificación de Tareas del Proceso de Calidad

El aseguramiento de calidad de la presente disertación se realizó siguiendo el formato de presentación definido por la Pontificia Universidad Católica del Ecuador.
Para el desarrollo del producto de software se definieron estándares de implementación y diseño que se detallan en el presente capítulo.

4.2 Implementación del proceso

4.2.1 Modelo de Ciclo de vida

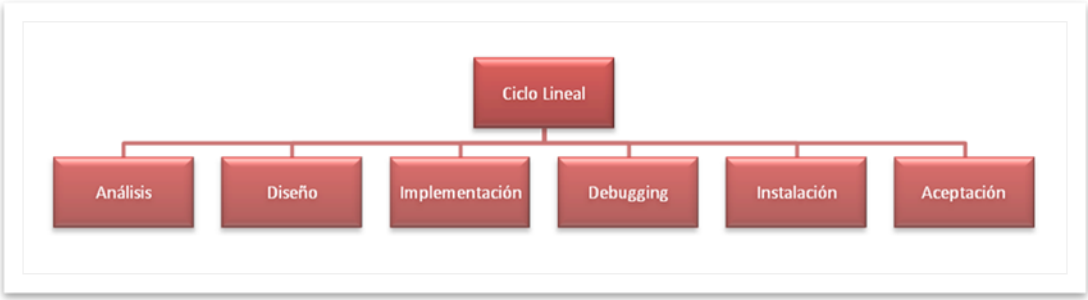


Ilustración 4-1: Fases del modelo de ciclo de vida lineal [D]

4.2.2 Plan de configuración

El cambio de la aplicación se puede dar en cualquier momento, por lo que es necesario tener un orden de las acciones a tomar en el plan de configuración que servirá para:

- Identificar el cambio del software.
- Controlar dicho cambio.
- Garantizar que el cambio quede bien implantado.
- Informar el cambio.

4.2.2.1 Línea Base:

Se puede definir como la especificación que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo, y que servirá como base para un desarrollo posterior pudiendo ser cambiada solamente a través de procedimientos formales de control de cambios.

Elementos que serán Línea Base:

1) Especificación del sistema.
2) Especificación de requisitos.
3) Especificación de diseños.
4) Descripción del diseño de datos.
5) Descripción del diseño arquitectónico.
6) Descripciones del diseño de interfaces.
7) Descripciones de los objetos.
8) Listados del código fuente.
9) Plan y procedimiento de pruebas.
10) Casos de prueba.
11) Programas ejecutables.
12) Descripción de la base de datos.
13) Manual del usuario final.
14) Informes de problemas del software.
15) Estándares.

4.2.2.2 Los Procedimientos de Control de Configuración

En esta sección se detallan las actividades de solicitud, evaluación, aprobación e implementación de cambios a los elementos de la línea base.

Los cambios apuntan tanto a la corrección como al mejoramiento.
El procedimiento que se describe a continuación es el que se utilizará cada vez que se precise introducir un cambio al sistema.

Se entiende por cambio al sistema, las modificaciones que afecten a la línea base del sistema, como pueden ser:

- Cambios en los Requerimientos.
- Cambios en el Diseño.
- Cambios en la Arquitectura.
- Cambios en las herramientas de desarrollo.
- Cambios en la documentación del proyecto (agregar nuevos documentos o modificar la estructura de los existentes).

4.2.2.3 Procedimiento para solicitud de cambios

1) Crear un documento con la solicitud de cambio.
2) Evaluar el impacto del cambio.
3) Revisar la solicitud de cambio para entender su alcance.
4) Determinar las personas del proyecto que deben realizar el análisis de evaluación del cambio e involucrarlas.
5) Desarrollar un plan para la evaluación del cambio.
6) Aprobar o desaprobar el cambio.
7) La "Mesa de Control de Configuración" es el comité encargado en la aprobación o desaprobación de cambios.
8) Implementación de cambios.
9) Una vez realizada la evaluación del cambio, se decide en qué momento implementarlo. Esta etapa involucra los procesos necesarios para implementar la solicitud y monitorear el progreso del trabajo.
10) Además se especificará el momento de liberación del cambio; así como también los responsables de las actividades que involucran el cambio.

4.2.2.4 Cambios de versión

Los cambios de versión serán realizados dependiendo de las necesidades del cliente.
Las versiones seguirán esta secuencia de numeración 1.0, 1.1, 1.2, etc.

4.2.2.5 La Mesa de Control de Configuración

Miembros encargados de aprobar los cambios en la línea base.

- Administrador de Desarrollo.
- Administrador de Configuración.

4.2.3 Herramientas para el desarrollo

A continuación se muestra una tabla con las herramientas de desarrollo seleccionadas en el capítulo tres.

Recursos tecnológicos	Seleccionado
CASE	Power Designer
Lenguaje de programación	JAVA
Gestor de base de datos	MySQL
IDE	NetBeans

Tabla 4-1: Selección de recursos tecnológicos. [A]

4.2.4 Planificación de Actividades

A continuación se detalla la estrategia de desarrollo para la implementación de las clases que conforman el sistema.

Clases	**Responsable**
Tipo de Persona	Cristina Matheu
Ingredientes	Cristina Matheu
Unidad de medida	Cristina Matheu
Ración	Cristina Matheu
Dieta	Carlos Cargua
Tipo de Dieta	Carlos Cargua
Etiqueta de la Ración	Carlos Cargua
Régimen	Carlos Cargua
Menú	Carlos Cargua
Dimensionamiento	Carlos Cargua, Cristina Matheu

Tabla 4-2: Estrategia de desarrollo. [A]

4.3 Análisis de los requerimientos del sistema

Antes de analizar los requerimientos que el sistema será capaz de cumplir, es necesario definir un conjunto de términos que permiten una mejor compresión del ambiente en el cual será instalado el sistema.

4.3.1 Glosario de términos

4.3.1.1 Tipo de persona

Representa el tipo de persona presentes en el hospital: pacientes y personal.

4.3.1.2 Dieta

Etimológicamente la palabra «dieta» significa 'régimen de vida'. Se acepta como sinónimo de régimen alimenticio, que alude al 'conjunto y cantidades de los alimentos o mezclas de alimentos que se consumen habitualmente'. Las dietas definidas en el departamento se mencionan a continuación y se detallan:

- Dieta normal
- Dieta líquida
- Dieta blanda
- Dieta restringida

4.3.1.3 Tipo de dieta

Los tipos de dietas son una clasificación de las dietas:

Dieta líquida

- Sonda
- Amplia
- Estricta

Dieta blanda

- Hiperproteínica sin residuos
- Hiperproteínica
- Sin lactosa

Restringida

- Régimen individual

- Hiposódica normoproteínica
- Hiposódica hipoproteica

4.3.1.4 Ingrediente

Es una sustancia que puede ser líquida o sólida que es apta para el consumo humano. Por ejemplo: leche, pan, huevo, sal, azúcar, arroz, carne etc.

4.3.1.5 Unidad de Medida

Es una cantidad estandarizada de una determinada magnitud física. Si es para líquidos están: litros, mililitros, decímetros cúbicos, galón, etc. Si es para sólidos están: libras, kilos, gramos, tonelada etc.

4.3.1.6 Ración

Se considera una ración alimenticia a la cantidad habitual que una persona consume de un alimento. Por ejemplo:

- Lácteos: una ración una taza de leche (250 cc), dos yogures o 40 gramos de queso.
- Alimentos proteicos: un plato de carne o pescado de unos 80-100 gramos, 2 huevos o un plato de legumbres cocidas.
- Pan y féculas: 60 gr. de pan (es decir, un bollito), un plato de arroz o pasta cocida (unos 60-80 gr. en crudo) o una patata mediana de 120 gr.
- Verduras y ensaladas: un plato.
- Fruta: una pieza de tamaño mediano (unos 200 gramos).

❖ Detalle de ración

Contiene el conjunto de ingredientes con su respectiva cantidad y unidad de medida.

4.3.1.7 Etiqueta de la ración

Permite la identificación rápida de una ración:

Etiqueta	Ración
Sopa	Menestrón de lenteja
	Crema de zapallo
Acompañado	Ensalada de tomate y lechuga
	Salsa de champiñones
Pan	Blanco
	Integral
	Dulce

Tabla 4-3: Ejemplificación de etiqueta. [A]

4.3.1.8 Régimen

Es el conjunto de sustancias alimentarias que se ingieren formando hábitos o comportamientos nutricionales que forma parte del estilo de vida del ser humano.
Agrupa el conjunto de Raciones que se deben ingerir en un Momento del día (desayuno, colación, almuerzo, cena, merienda).

- Momento del día

Representa el momento en que debe ingerirse las raciones. Por ejemplo un desayuno a las 7:30, una colación a las 10:15 de la mañana, etc.

- Detalle del régimen

Comprende todas las raciones que serán consumidas por las personas.

4.3.1.9 Menú

Representa el menú del día. En éste constan todas las raciones o comidas que las personas ingerirán en un momento del día (régimen) siguiendo un tipo de dieta. Un ejemplo de menú se encuentra en el ANEXO B.

- Detalle de menú

Comprende la agrupación de los régimenes que forman el menú de un solo día.

4.3.1.10 Proceso de Dimensionamiento

Es el proceso de dimensionar la ración para un mayor número de personas. Por ejemplo, una sandía se puede compartir a unas diez personas aproximadamente, este detalle depende de la cantidad necesaria que un ser humano deba ingerir para generar energía o adquirir vitaminas esenciales para el correcto funcionamiento de su organismo. (Esta tarea la lleva a cabo un nutricionista experto en el tema). A continuación nos vemos en la necesidad de conocer ¿Cuántas sandias serán necesarias para compartirlas entre cien personas? Para ello utilizamos un cálculo sencillo; una regla de tres, lo que nos da como resultado cien sandías. La precisión del dimensionamiento dependerá de la precisión con que son planificadas las raciones.

4.3.2 Requerimientos del Sistema

4.3.2.1 Requerimientos funcionales

El sistema permitirá:

F0. Ingresar al Sistema

F1. Administrar la clase Tipo de persona

- **F1.1** Ingresar
- **F1.2** Editar
- **F1.3** Eliminar

F2. Administrar la clase Dieta

- **F2.1** Ingresar
- **F2.2** Editar
- **F2.3** Eliminar

F3. Administrar la clase Tipo de Dieta

- **F3.1** Ingresar
- **F3.2** Editar
- **F3.3** Eliminar

F4. Administrar la clase Ingrediente

- **F4.1** Ingresar
- **F4.2** Editar
- **F4.3** Eliminar

F5. Administrar la clase unidad de medida

- **F5.1** Ingresar

F5.2 Editar

F5.3 Eliminar

F6. Administrar la clase etiqueta de la ración

F6.1 Ingresar

F6.2 Editar

F6.3 Eliminar

F7. Administrar la clase ración

F7.1 Ingresar

F7.2 Editar

F7.3 Eliminar

F8. Administrar la clase Régimen

F8.1 Ingresar

F8.2 Editar

F8.3 Eliminar

F9. Administrar la clase Menú

F9.1 Ingresar

F9.2 Editar

F9.3 Eliminar

F10. Administrar la clase Usuario

F10.1 Ingresar

F10.2 Editar

F10.3 Eliminar

F11. Administrar el Proceso de Dimensionamiento de ración

F11.1 Ingresar

F11.2 Editar

F11.3 Eliminar

F12. Consultas

F12.1. Consulta de tipo de persona

F12.1.1 Consulta general

F12.1.2 Consulta por parámetro

F12.2. Consulta de Dieta

F12.2.1 Consulta general

F12.2.2 Consulta por parámetro

F12.3. Consulta de Tipo de Dieta

F12.3.1 Consulta general

F12.3.2 Consulta por parámetro

F12.4. Consulta de Ingrediente

F12.4.1 Consulta general

F12.4.2 Consulta por parámetro

F12.5. Consulta de Unidad de medida

F12.5.1 Consulta general

F12.5.2 Consulta por parámetro

F12.6. Consulta de etiqueta

F12.6.1 Consulta general

F12.6.2 Consulta por parámetro

F12.7. Consulta de ración

F12.7.1 Consulta general

F12.7.2 Consulta por parámetro

F12.8. Consulta de régimen

F12.8.1 Consulta general

F12.8.2 Consulta por parámetro

F12.9. Consulta de menú

F12.9.1 Consulta general

F12.9.2 Consulta por parámetro

F12.10. Consulta de Usuarios

F12.10.1 Consulta general

F12.10.2 Consulta por parámetro

F12.11. Consulta del Proceso de Dimensionamiento

F12.11.1 Consulta general

F12.11.2 Consulta por parámetro

F13. Salida del Sistema

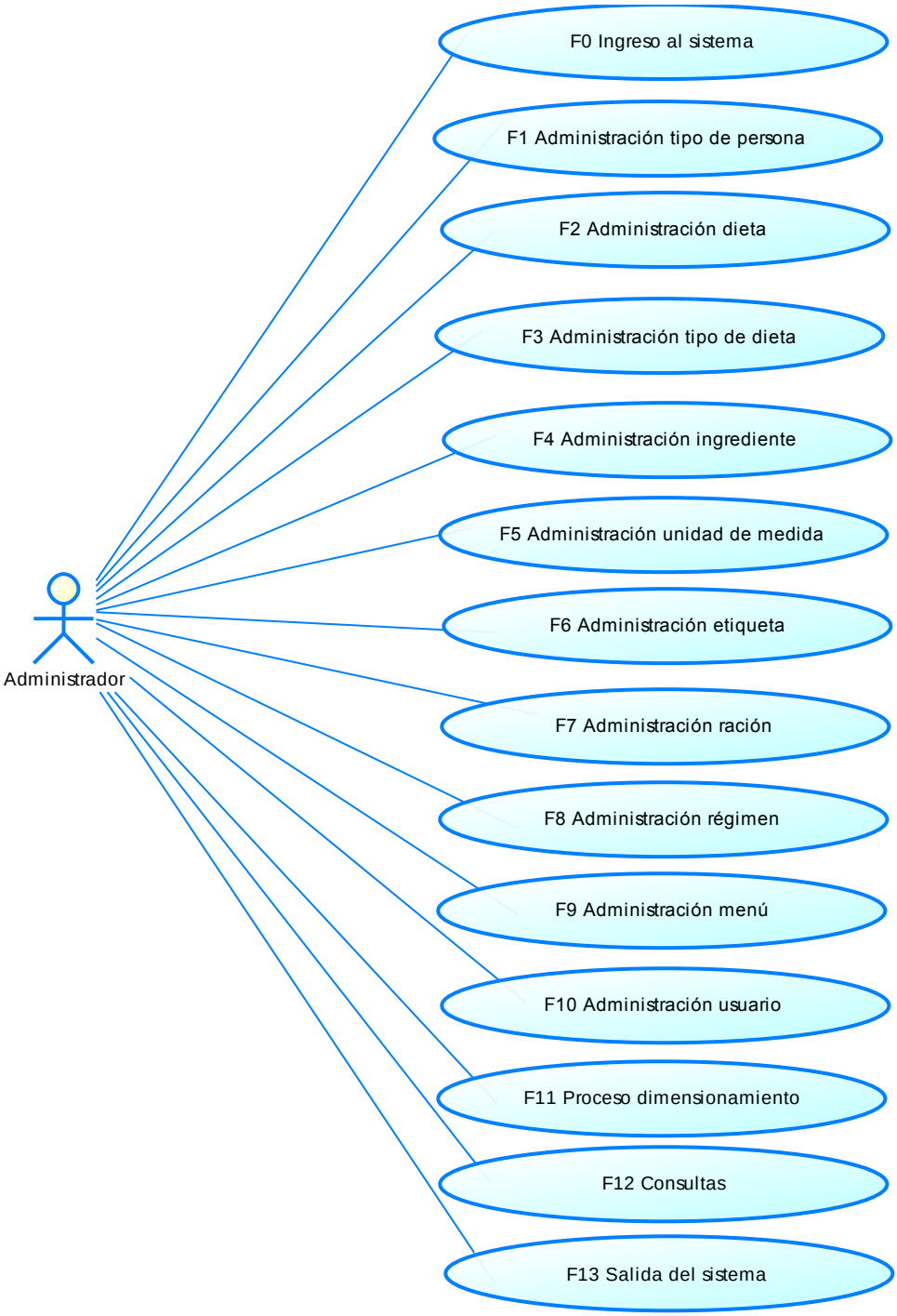
F13.1 Cerrar Sesión

F13.2 Salir del sistema

4.3.2.2 Casos de uso

Desde este punto hasta la sección 4.5.2.4 los diagramas que se ilustran son de autoría propia.

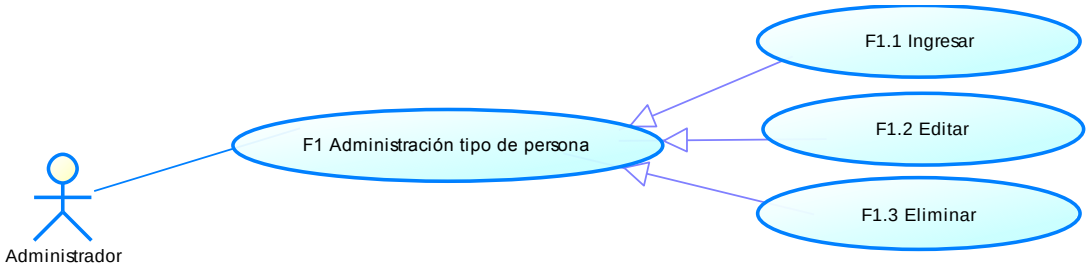
- Diagrama General



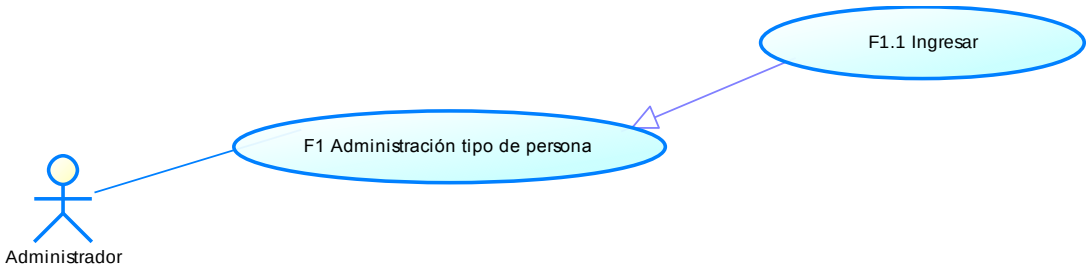
❖ Diagramas a detalle

Para ilustrar los diagramas de casos de uso a detalle se tomó como ejemplo la administración y la consulta general y por parámetro de la clase tipo de persona. El resto de diagramas se adjuntan en el CD que se anexan al presente trabajo.

F1 Administración tipo de persona



F1.1 Ingresar



Actor: Administrador.
Definición: Permitirá el ingreso de un tipo de persona.

Descripción:

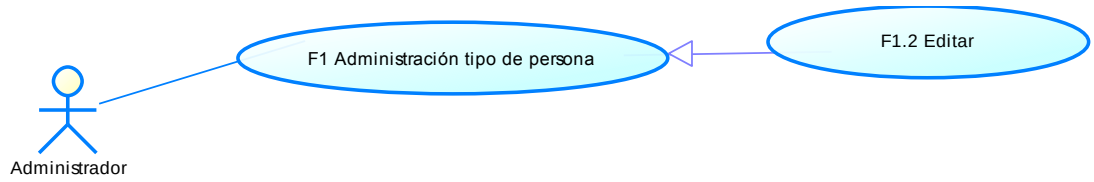
Paso	Actor	Paso	Sistema	Excepción
1	Selecciona la pestaña Administración de Ingredientes.	2	Presenta la Ventana de Ingredientes.	
		3	Carga un nuevo código automáticamente.	E1
4	Ingresa la información del nuevo Ingrediente			

5	Presiona el botón “Guardar”.	6	Verifica que esté lleno el formulario.	E2
		7	Guarda los datos en la base de datos.	E1,E2

Excepción:

Código	Descripción	Alternativa
E1	No hay conexión a la base de datos	Presentar mensaje de error.
E2	No se graban los datos.	Presentar mensaje.

F1.2 Editar



Actores: Administrador

Definición: Permitirá modificar un tipo de persona.

Descripción:

Paso	Actor	Paso	Sistema	Excepción
1	Selecciona la opción Consultas > Administración > Tipo de persona > General del menú principal.	2	Muestra la ventana Consulta General de tipo de persona.	E1
		3	Carga lista de tipo de personas.	

4	Selecciona el tipo de persona y escoge la opción editar.	5	Muestra la ventana Tipo de persona con los datos de ese registro.	
6	Actualiza los datos.			
7	Presiona el botón grabar.	8	Actualiza la información en la base de datos.	E1
		9	Muestra un mensaje de que la información fue grabada con éxito.	

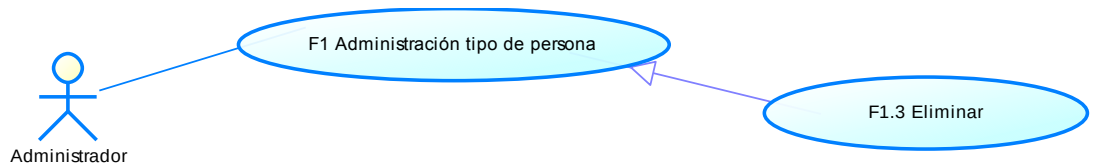
Excepción:

Código	Descripción	Alternativa
E1	No hay conexión a la base de datos.	Presentar mensaje de error.

Excepción:

Paso	Descripción
3	Si no hay información, ver F1.1 [Ingreso de tipo de persona]

F1.3 Eliminar



Actores: Administrador

Definición: Permitirá eliminar de forma lógica los tipos de persona que ya no se utilizan.

Descripción:

Paso	Actor	Paso	Sistema	Excepción
1	Selecciona la opción Consultas > Administración > Tipo de persona > General del menú principal.	2	Muestra la ventana Consulta General de tipo de persona.	E1
		3	Carga lista de tipo de personas.	
4	Selecciona el tipo de persona y escoge la opción eliminar.	5	Muestra un diálogo de confirmación.	
6	Confirma.	7	Actualiza la información en la base de datos.	E1
		8	Muestra un mensaje de que el registro fue eliminado con éxito.	

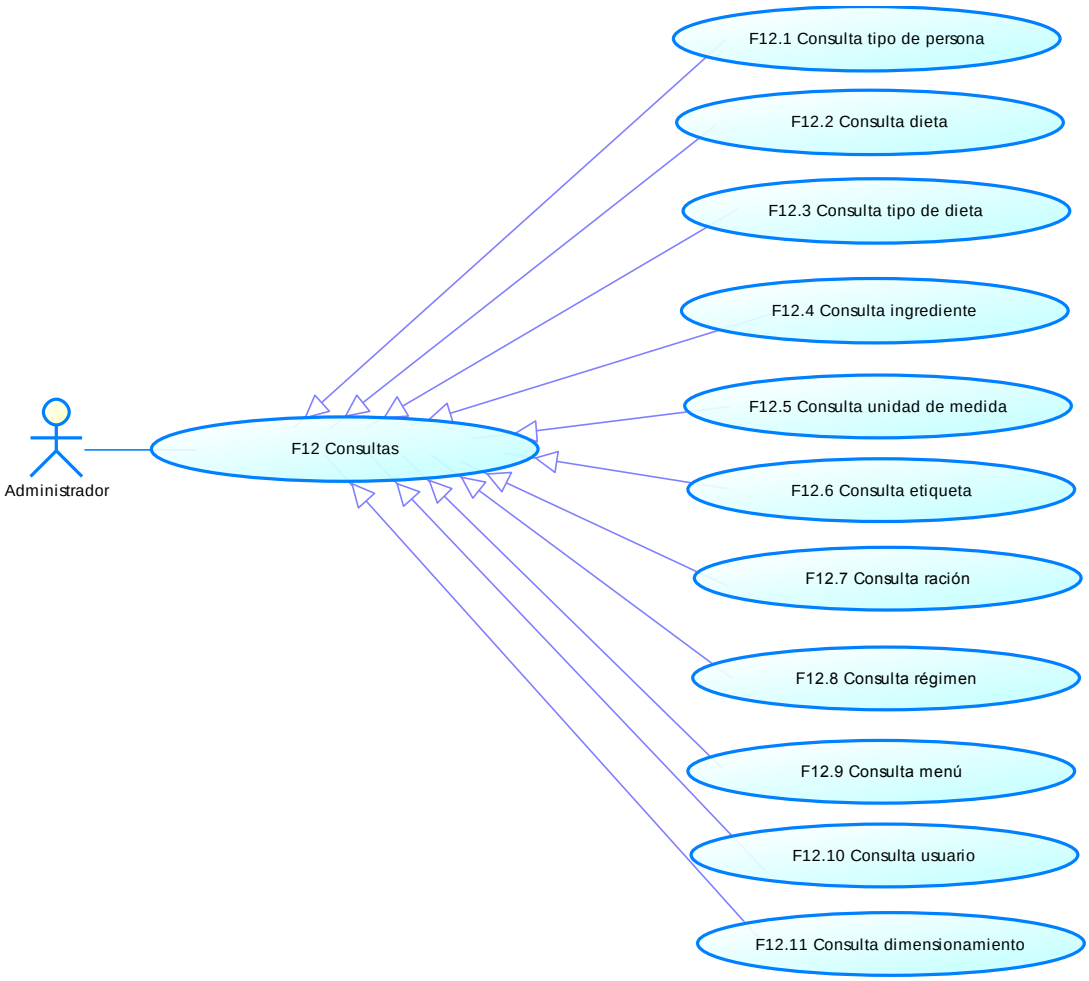
Excepción:

Código	Descripción	Alternativa
E1	No hay conexión a la base de datos.	Presentar mensaje de error.

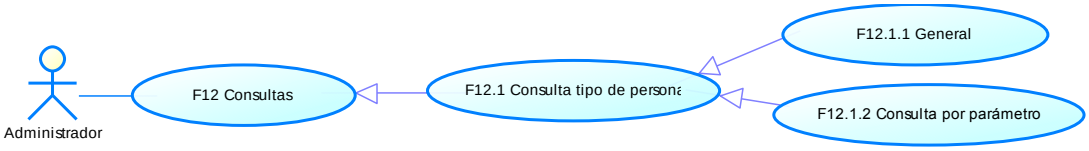
Extensión:

Paso	Descripción
6	Si no confirma, ver F1.1, F1.2 [Ingreso y edición de tipo de persona]

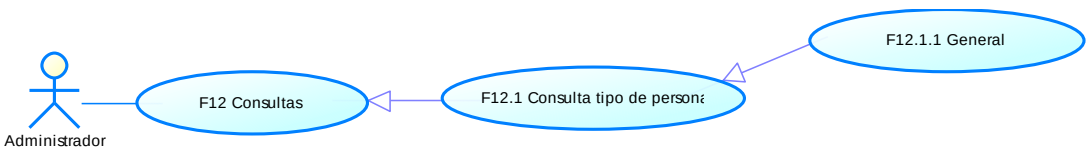
F12 Consultas



F12.1 Consulta tipo de persona



F12.1.1 General



Actores: Administrador.

Definición: Permite al administrador consultar de forma genérica todos los datos del tipo de persona.

Descripción:

Paso	Actor	Paso	Sistema	Excepción
1	Selecciona la opción Consultas > Administración > Tipo de persona > General del menú principal.	2	Muestra la ventana Consulta General de Tipo de persona	E1
		3	Recupera la información.	E1
		4	Presenta la información.	

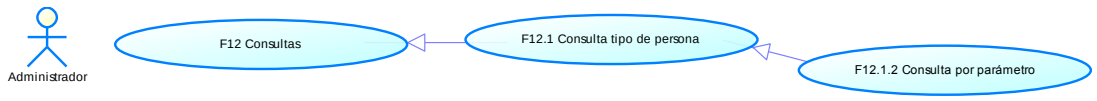
Excepción:

Código	Descripción	Alternativa
E1	No hay conexión a la base de datos.	Presentar mensaje de error.

Extensión:

Paso	Descripción
3	Si no hay información ver F1.1 [Ingreso de tipo de persona]

F12.1.2 Por parámetro



Paso	Actor	Paso	Sistema	Excepción
1	Selecciona la opción Consultas > Administración > Tipo de persona > Por parámetro del menú principal.	2	Muestra la ventana Consulta Por parámetro de Tipo de persona.	E1
3	Escoge el parámetro de búsqueda.			
4	Ingresa la información del parámetro escogido.			
5	Presiona el botón buscar.	6	Busca en la base de datos.	E1,E2
		7	Presenta la información solicitada.	

Excepción:

Código	Descripción	Alternativa
E1	No hay conexión a la base de datos.	Presentar mensaje de error.
E2	Registro no encontrado.	Presentar mensaje de registro no encontrado.

Extensión:

Paso	Descripción
7	Si no hay registros ver F1.1 [Ingreso de tipo de persona]

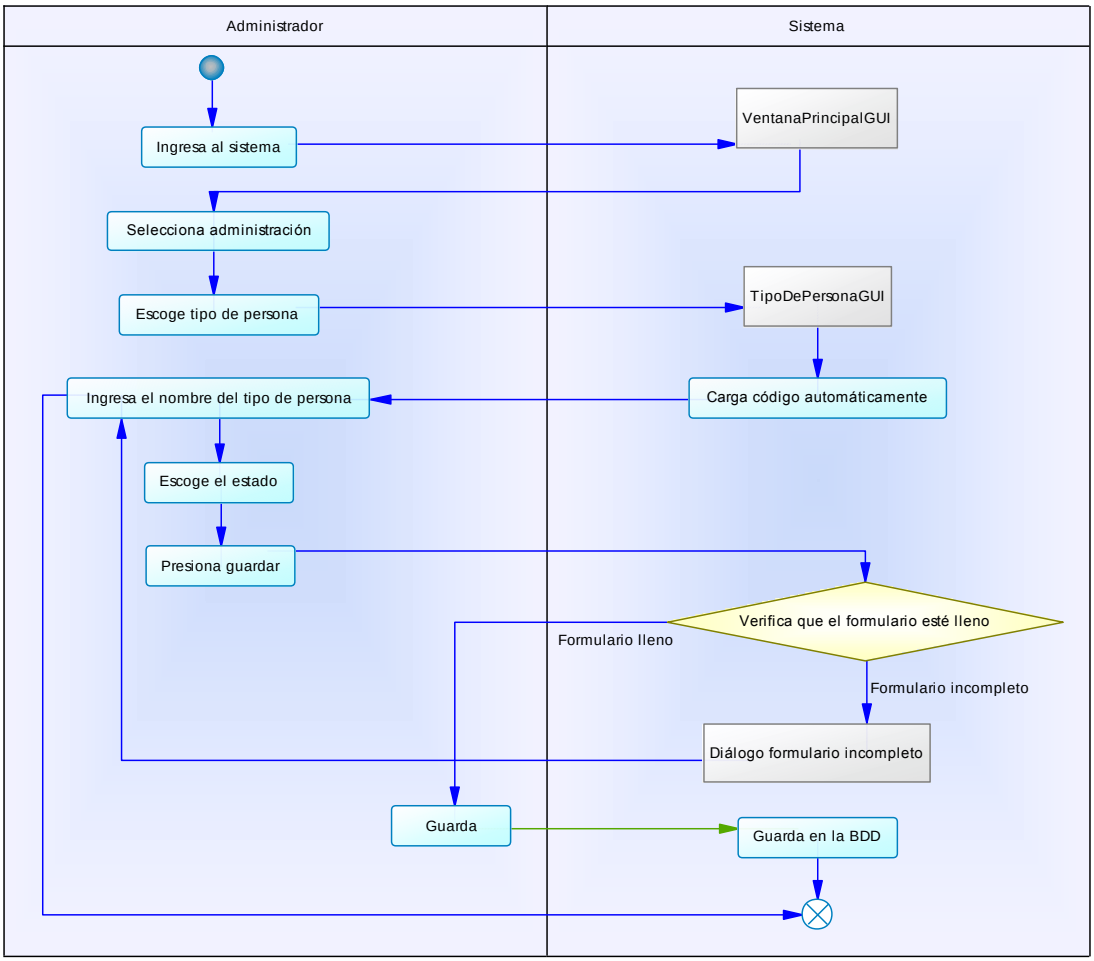
4.3.3 Evaluación de los Requerimientos

Luego de enumerar el listado de los requerimientos funcionales, se concluyó que estos podrán ser desarrollados.

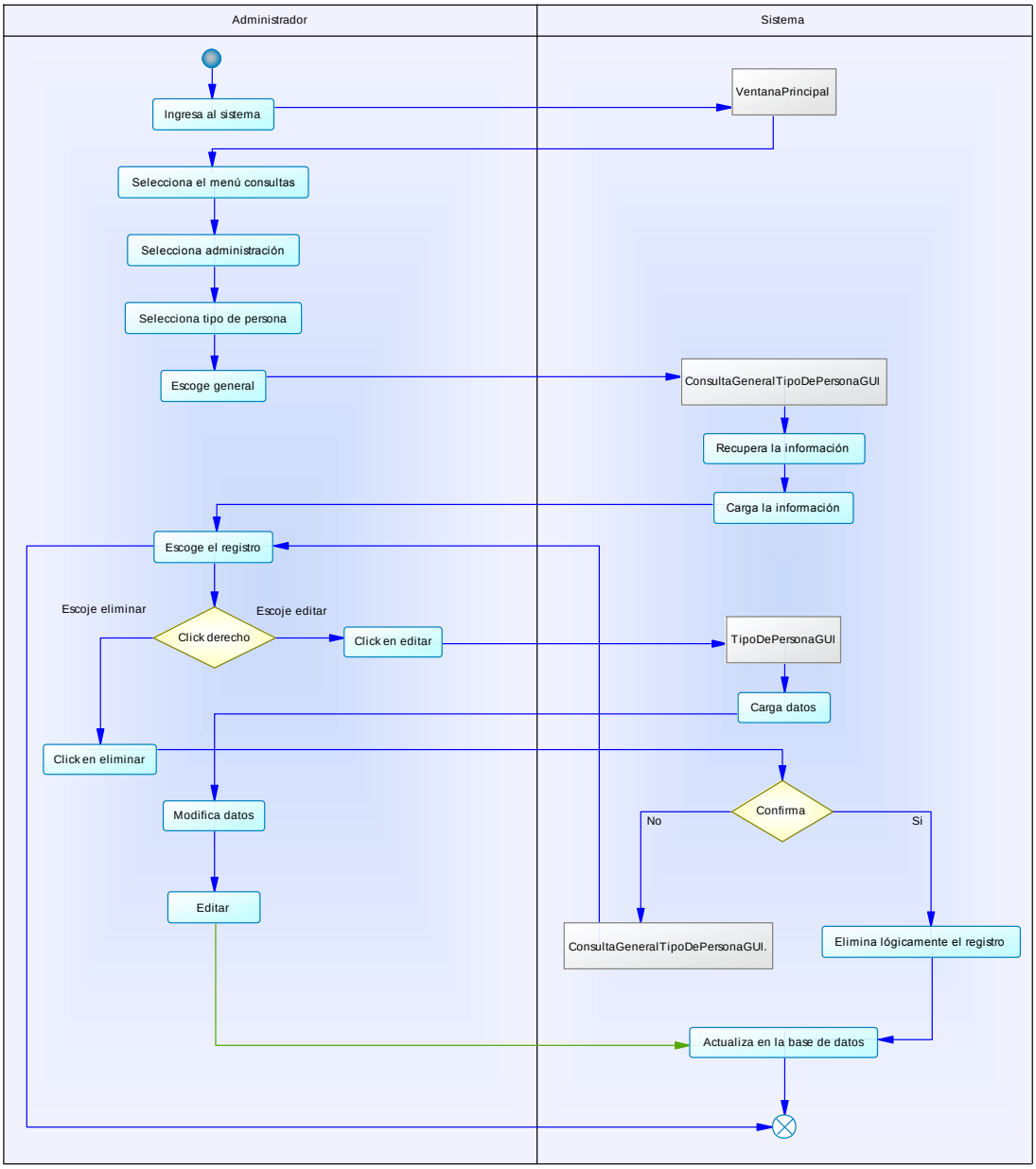
4.3.3.1 Diagramas de actividad

F1 Administración tipo de persona

F1.1 Ingresar

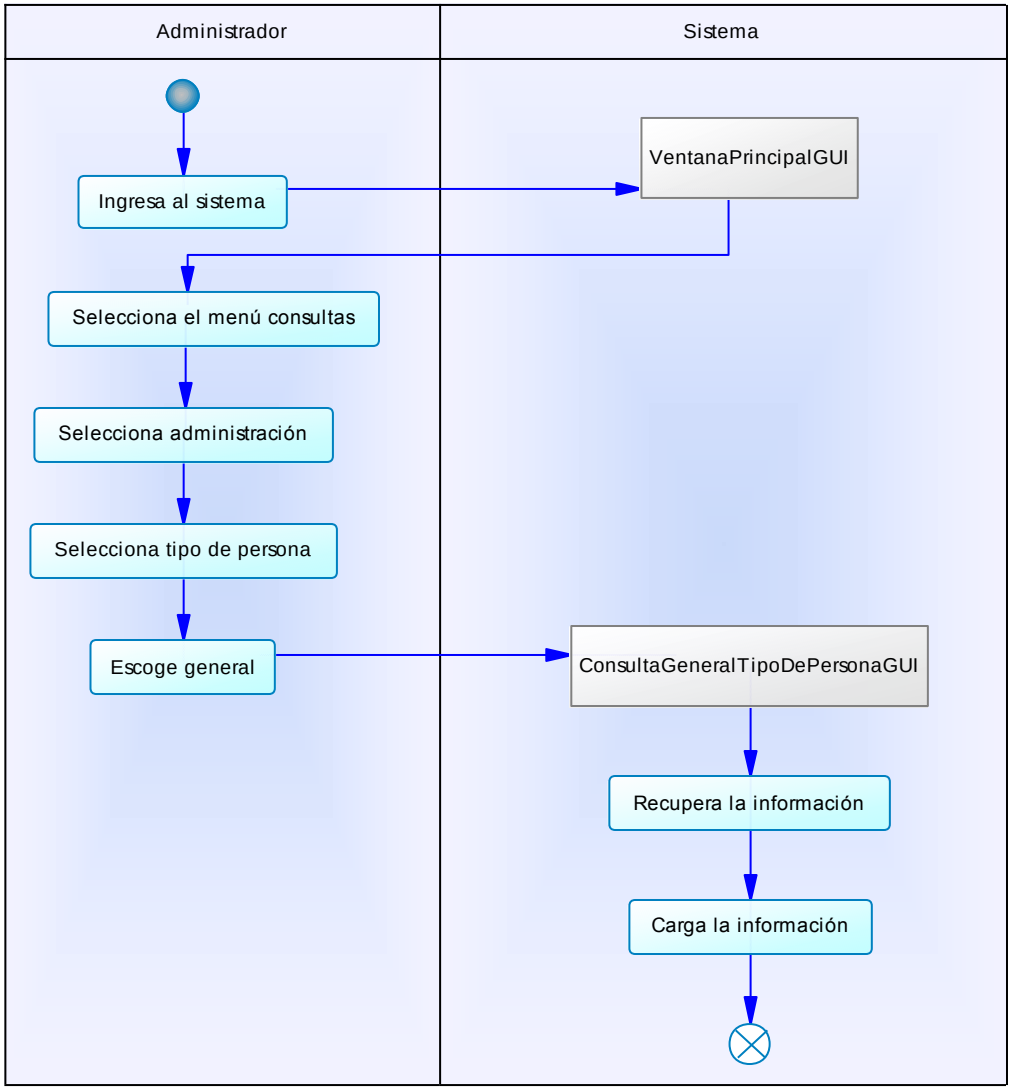


F1.2 Editar y F1.3 eliminar

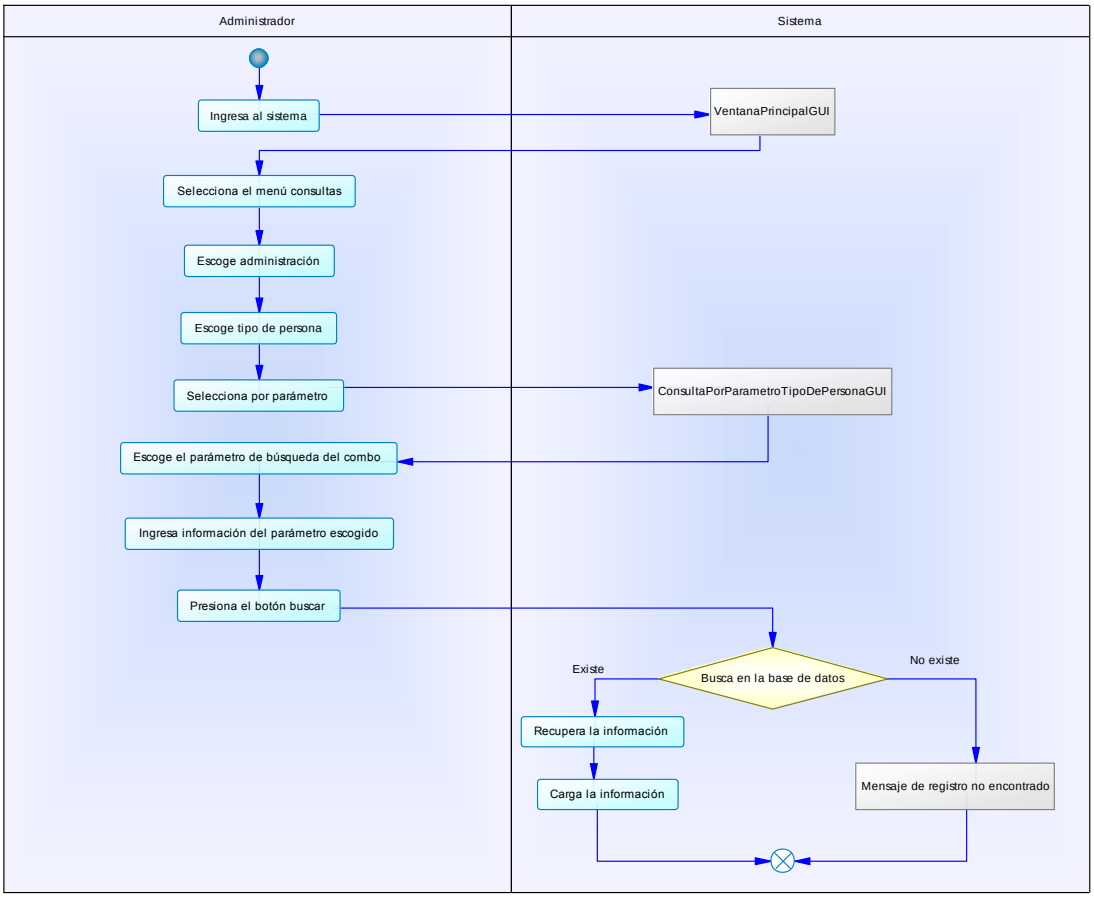


F12. Consultas

F12.1.1 Consulta general Tipo de Persona



12.1.2 Consulta por parámetro Tipo de Persona



4.4 Diseño de la arquitectura

El diseño de la arquitectura del sistema comprende la definición de los requerimientos mínimos a nivel de hardware y software y de la definición de los estándares de desarrollo de interfaces gráficas y de codificación de los módulos del sistema.

4.4.1 Arquitectura del Sistema

4.4.1.1 A nivel de Hardware

Requerimientos mínimos:

Requerimiento	**Cliente**
Memoria RAM	128 Mb
Disco duro	10 Gigabytes
Procesador	Pentium 4

Tabla 4-4: Estrategia de desarrollo. [A]

4.4.1.2 A nivel de Software

Requerimientos mínimos:

Requerimiento	Cliente
Sistema Operativo	Windows XP o Windows 7
Gestor de BDD	MySQL
JAVA	JRE 1.6.0

Tabla 4-5 Estrategia de desarrollo. [A]

❖ Diseño por capas

Esta solución informática será desarrollada en tres capas, es decir se dividirá la aplicación en capas distribuidas de la siguiente manera:

Capa 1: Contendrá la interfaz gráfica que facilite al usuario la comunicación con el sistema. → GUI.
Capa 2: Servirá para centralizar la lógica del negocio. → DP.
Capa 3: Servirá para establecer la conexión a la base de datos. → MD.

❖ Estándares de programación

➢ *Estándares generales*

- El sistema implementará tres capas: la interfaz gráfica (GUI), el dominio del problema (DP) y la conexión a la base de datos (MD).
- Documentar clases, métodos e interfaces con el fin de comprender el propósito por el cual es implementado.
- Crear un utilitario de interfaz grafica (GUI) con el objetivo de reusar código
- Crear un utilitario a nivel de base de datos (MD) con el objetivo de agilizar la conexión a la información guardada.

- *Estándar de diseño de interfaz gráfica (GUI)*

- Dentro de la clase GUI existe un objeto llamado clase DP el cual hace referencia a la clase básica definida.
- Dentro de cada clase GUI se incluirá el Utilitario GUI que permitirá reusar código.
- Existe un manejador de ventanas denominado "manager". Este permite recuperar y cambiar las ventanas con get y set. También guarda el usuario que ingresa al sistema, el cual se puede cambiar o recuperar con get y set.
- Todas las ventanas tendrán 2 constructores, 1 en blanco y el otro recibe como parámetro a "manager".
- Todas las ventanas tendrán una variable entera denominada "desde" la cual permite identificar desde que ventana fue construida.
- Todas las ventanas son inicializadas en manager con el constructor en blanco.
- Todas las ventanas de consultas que tenga la variable "desde" inicializada en cero permitirán la creación de un nuevo registro. Es decir habrá un botón visible llamado nuevo.
- Todas las tablas de las consultas tendrán un pop up que permitirá: editar, eliminar o ejecutar cualquier otra función necesaria.
- Todas las ventanas tendrán un objeto tipo usuario.
- Todas las ventanas de administración y procesos estarán predispuestas para guardar un nuevo registro. Es decir tendrán valores por defecto en campos de texto para códigos, combos, spinners, radio buttons, no en tablas ni en otros campos de texto que no sean para códigos. Además estarán habilitados los botones guardar, nuevo y cerrar.
- Las ventanas de administración y procesos también contendrán funciones como cambiar atributos de la clase DP, set clase DP, cargar datos.
- Todas las ventanas de administración tendrán una función denominada preparar nuevo código el cual carga un nuevo código en el campo de texto respectivo, también tendrá otra función llamada preparar nuevo registro el cual limpia el formulario, reinicia variables crea un nuevo objeto de la clase DP a la cual pertenece y solo habilita el botón guardar. Adicionalmente una función llamada preparar para editar o eliminar, el cual cambia de nombre al botón guardar por grabar y habilita el botón eliminar. Además una función llamada iniciar; en caso de ser necesario, la cual podría traer tree map, inicializar elementos de tablas, etc.

- Las funciones estarán agrupadas por comentarios convencionales de java y serán fijos: manejo de formulario, clase DP, existiendo la posibilidad de agregar nuevos comentarios.
- Mostrar mensaje de éxito cuando una acción ha sido llevada a cabo en su totalidad.
- Mostrar mensaje de advertencia cuando un formulario este incompleto o cuando los datos ingresados sean inconsistentes.
- Mostrar mensaje de error cuando falló la conexión con la base de datos.
- Centrar ventanas internas antes de ser presentadas.
- Agregar herramienta de consejo (tool tip) a los ítems del menú principal.
- No agregar herramienta de consejo (tool tip) cuando es clara la etiqueta de un botón, tabla, spinner o cualquier otro objeto.
- Los combos que cargan información de la base de datos tienen dos columnas una es invisible al usuario y contiene la clave primaria del registro la otra columna es visible al usuario y muestra algún tipo de información que permita la rápida identificación del registro.
- En cada grilla (tabla) se añade una etiqueta y un campo de texto que permita buscar información en cada una de las columnas de dicha tabla. Esto es que conforme se ingrese información en el campo de texto de búsqueda se irá filtrando la información de la tabla, es decir se mostrará solo las coincidencias; el resto de información se almacenará en memoria con el objetivo de ser recuperada.
- Al pie de cada grilla existirá un campo de texto con el número de registros recuperados de la base de datos. Además de un botón "actualizar" que permita recuperar y presentar nueva información en caso de haberla.
- Cuando es presionado un botón, un ítem de un menú, desplegado un pop up, el puntero del mouse cambiará por un puntero de espera hasta que termine la acción.
- Controlar el número de caracteres en un campo de texto.
- Controlar que se ingrese solo números en un campo de texto que es solo para dígitos.
- Controlar que campos de texto obligatorios estén llenos.
- Controlar que se escoja un ítem diferente a "escoja uno" en combos.

➢ *Estándar de desarrollo de clases de dominio del negocio (DP)*

- Todas las clases DP tienen una clase GUI. Salvo las de detalle o históricos.

- Existe una variable entera denominada sistema gestor de base de datos sgdb la cual permite identificar al sistema gestor de base de datos como: oracle, mysql, puente odbc, postgresql, sql server etc.
- Existe una clase denominada usuario la cual contiene el sgdb.
- Todas las clases DP tendrán una clase MD, excepto las clases detalle o histórico.
- Todas las clases DP tendrán un objeto tipo usuario.
- Todas las clases DP tendrán dos constructores: uno en blanco y otro que recibe como parámetro al objeto usuario.

➢ *Estándar de conexión a la base de datos (MD)*

- Todas las clases MD tendrán dos constructores: uno en blanco y otro que recibe como parámetro al objeto de la clase DP a la que corresponde.
- Dentro de cada clase MD se incluirá el Utilitario MD que permitirá agilizar la conexión.
- Se definirá un nombre que hará referencia a la tabla en base de datos.
- Existirá dentro de cada clase tres tipos de objetos: un objeto tipo conexión (connection), un estatuto (statement) y un cursor (result set).

4.4.2 Evaluación de la Arquitectura

Las características descritas anteriormente se deberán tomar en cuenta y son de suma importancia, puesto que dependerá de este factor el uso de la aplicación.

4.4.2.1 A nivel de Hardware

La computadora donde el sistema será instalado cuenta con las siguientes características:

Requerimiento	**Cliente**
Memoria RAM	2 GB
Disco duro	100 GB
Procesador	AMD Athlon (tm) II X2 B24

Tabla 4-6: Características Hardware Computadora Hospital. [A]

4.4.2.2 A nivel de Software

La computadora donde el sistema será instalado cuenta con las siguientes características:

Requerimiento	Cliente
Sistema Operativo	Windows XP
Gestor de BDD	MySQL (Será instalado)
JAVA	JRE 1.6.0 (Será instalado)

Tabla 4-7: Características Hardware Computadora Hospital. [A]

Como se puede observar las características detalladas de la computadora presente en el Departamento de Nutrición de Hospital de Niños "Baca Ortiz" cumple con los requerimientos mínimos, por lo que es factible la instalación de la solución informática.

4.5 Diseño detallado del software

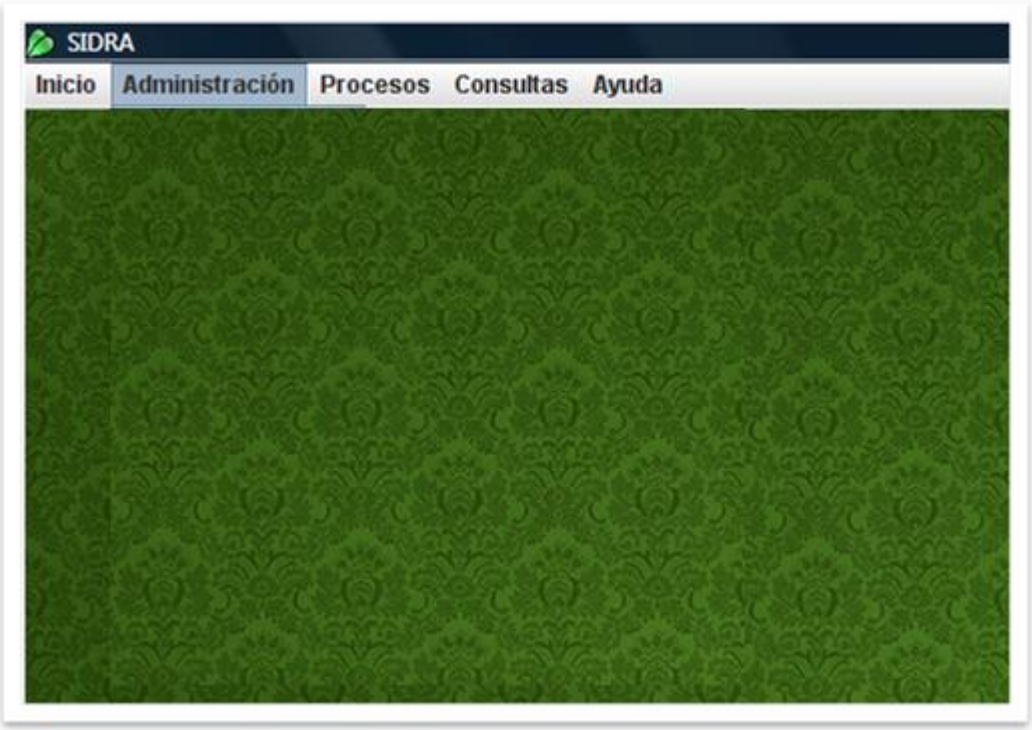
4.5.1 Diseño General

4.5.1.1 Prototipos de la interfaz

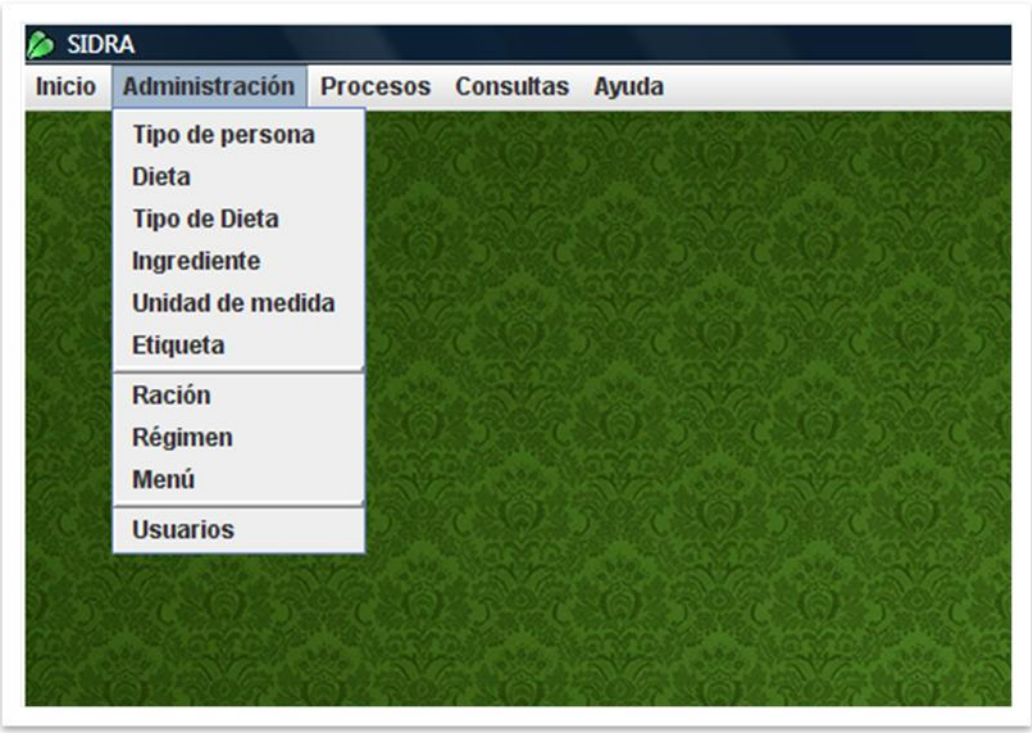
A continuación se muestran las interfaces que permiten al usuario la comunicación con el sistema.

- Ingreso al sistema

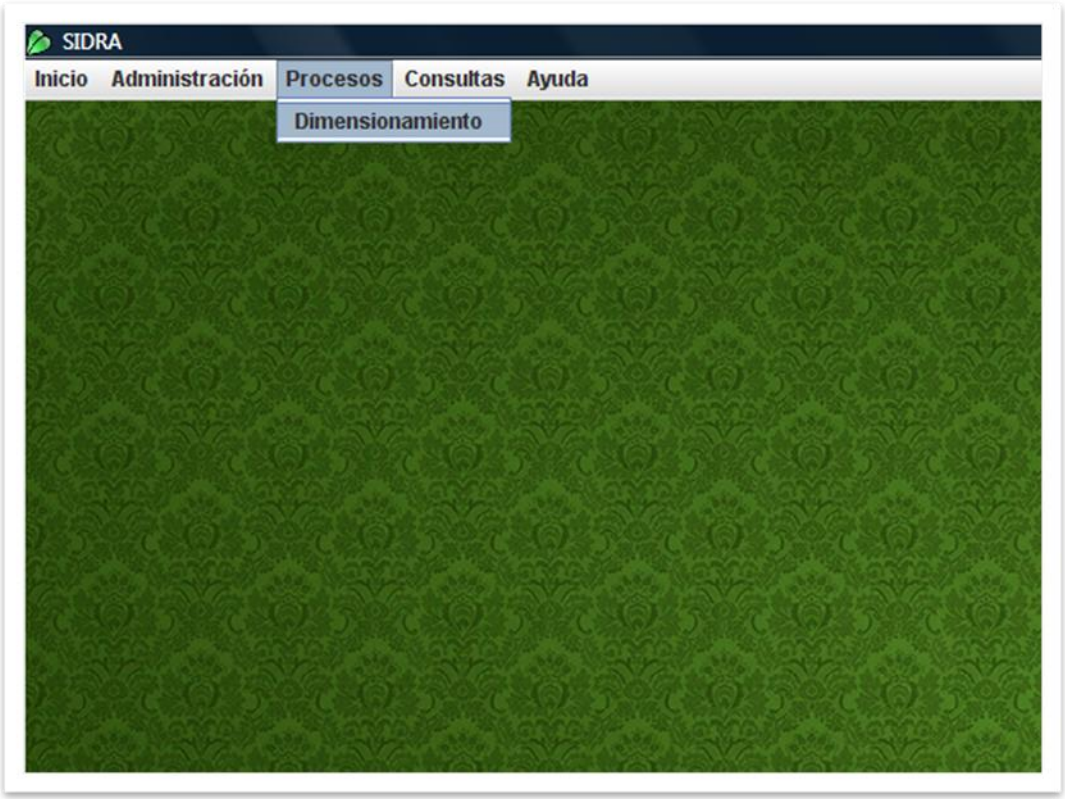
- Menú Principal



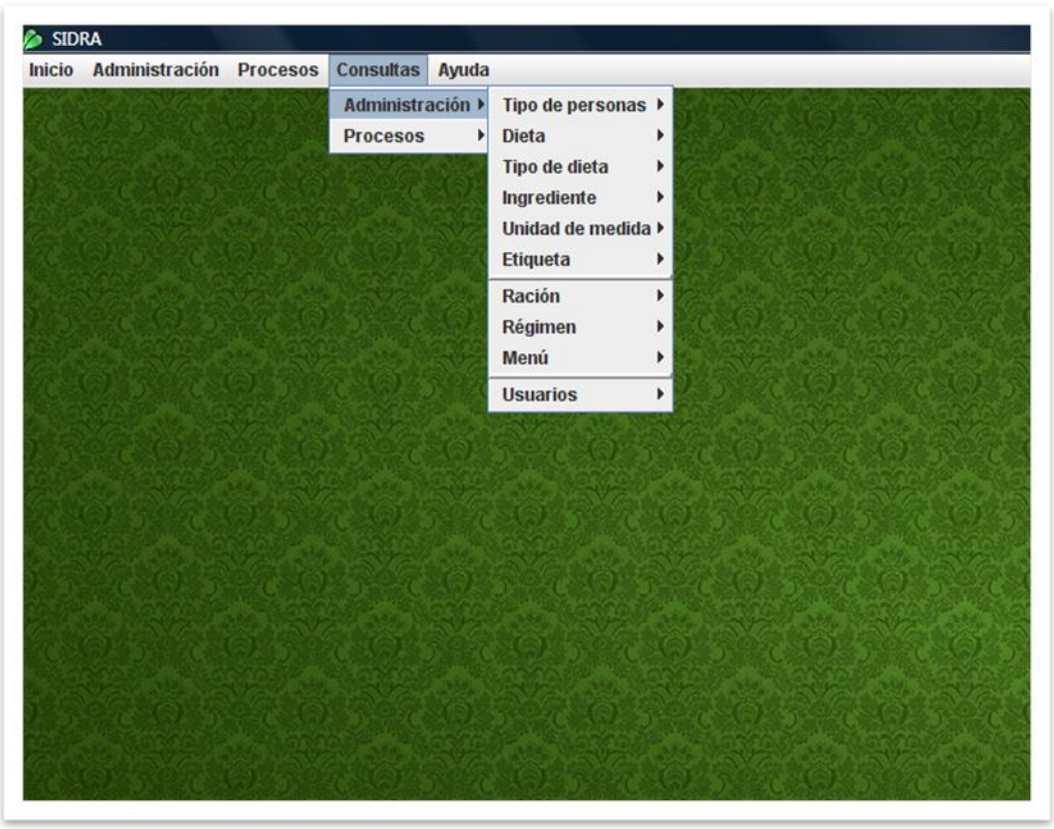
- Administración



- Proceso



- Consultas

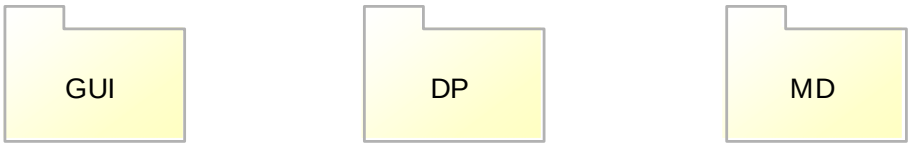


- Acerca de

4.5.2 Diseño Arquitectónico

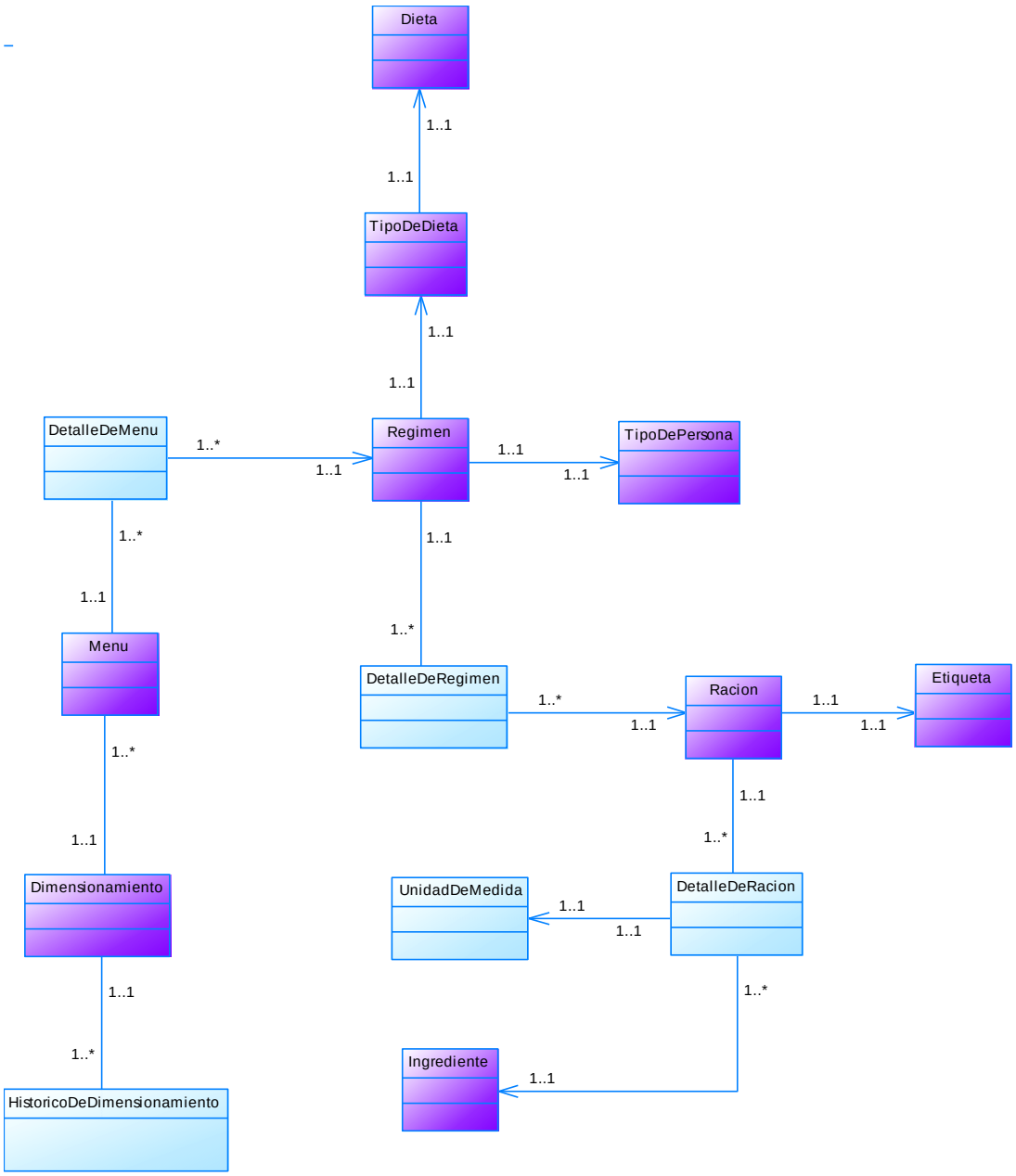
El diseño arquitectónico comprende la elaboración detallada de componentes internos del sistema; así como, también la elaboración de diagramas que permiten entender su funcionamiento en un ambiente de prueba.

4.5.2.1 Diagrama de paquetes



4.5.2.2 Diagrama de clases

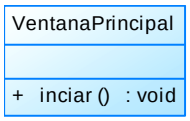
❖ Diagrama conceptual

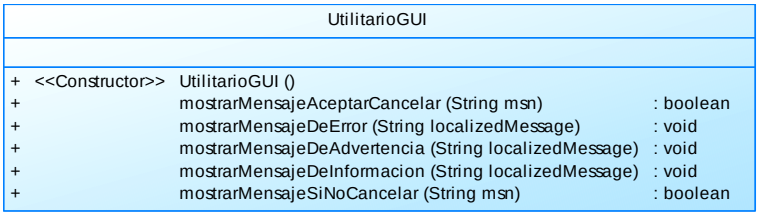


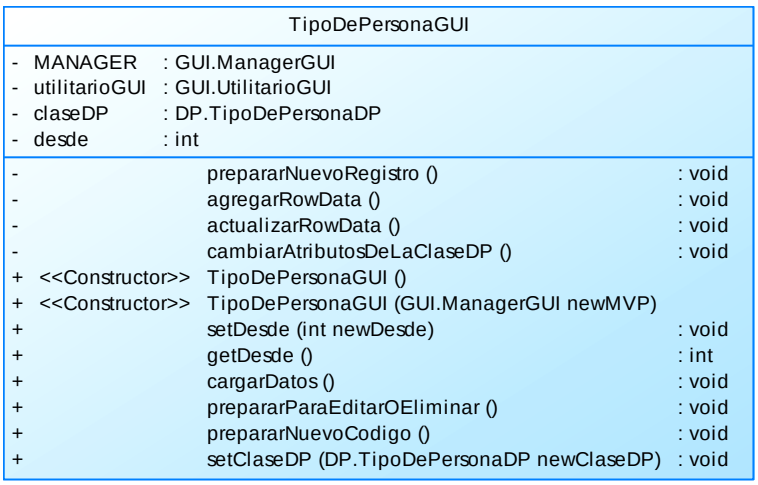
❖ Diagramas a detalle

Para ilustrar los diagramas a detalle de clases se tomó como ejemplo la administración y la consulta general y por parámetro de la clase tipo de persona. El resto de diagramas se adjuntan en el CD que se anexa al presente trabajo.

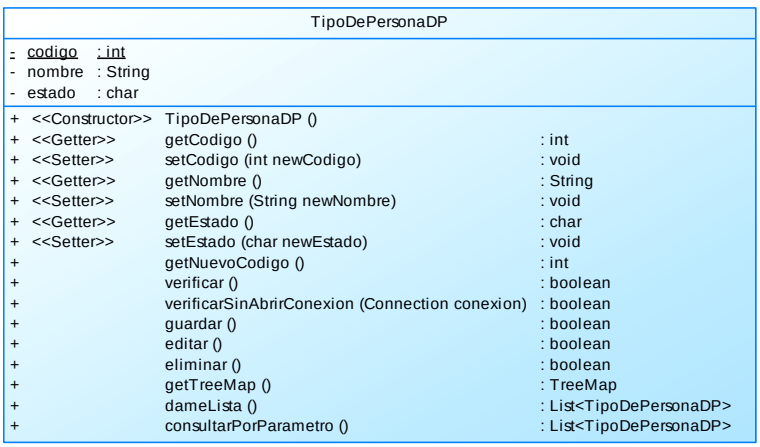
- *Interfaz gráfica (GUI)*



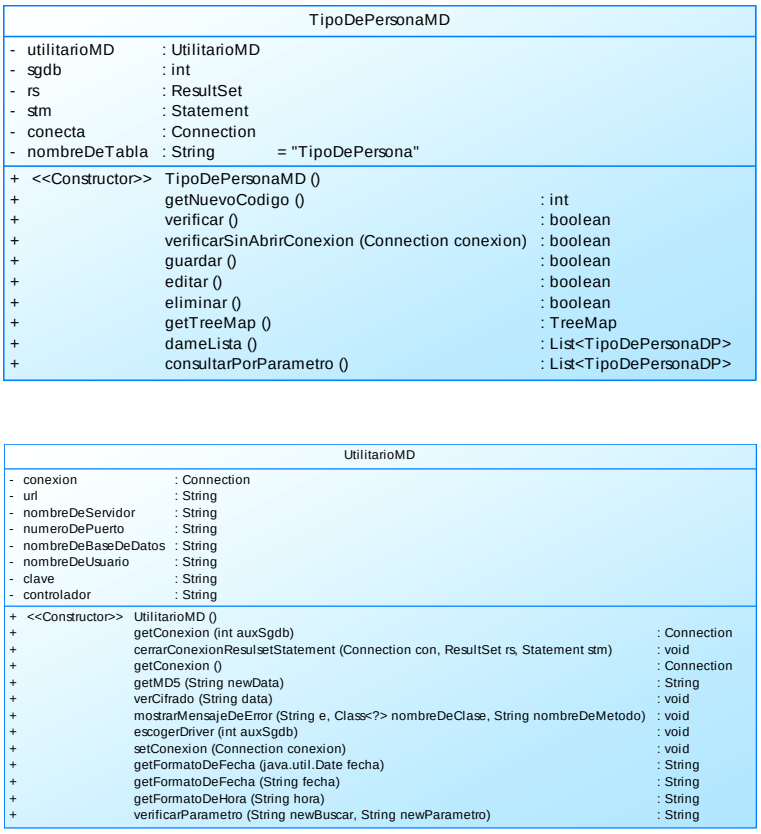




- *Dominio del negocio (DP)*



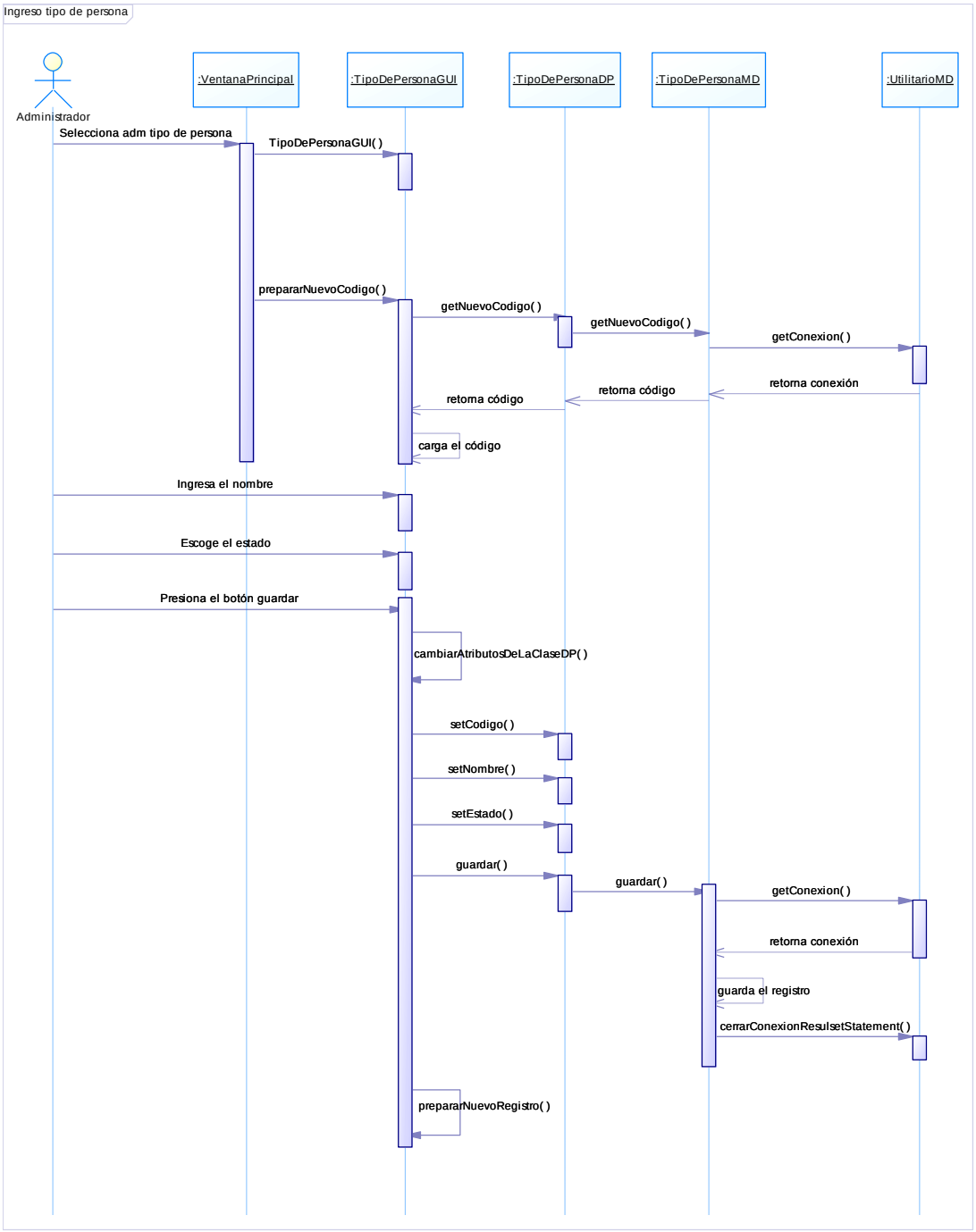
- *Conexión a la base de datos (MD)*



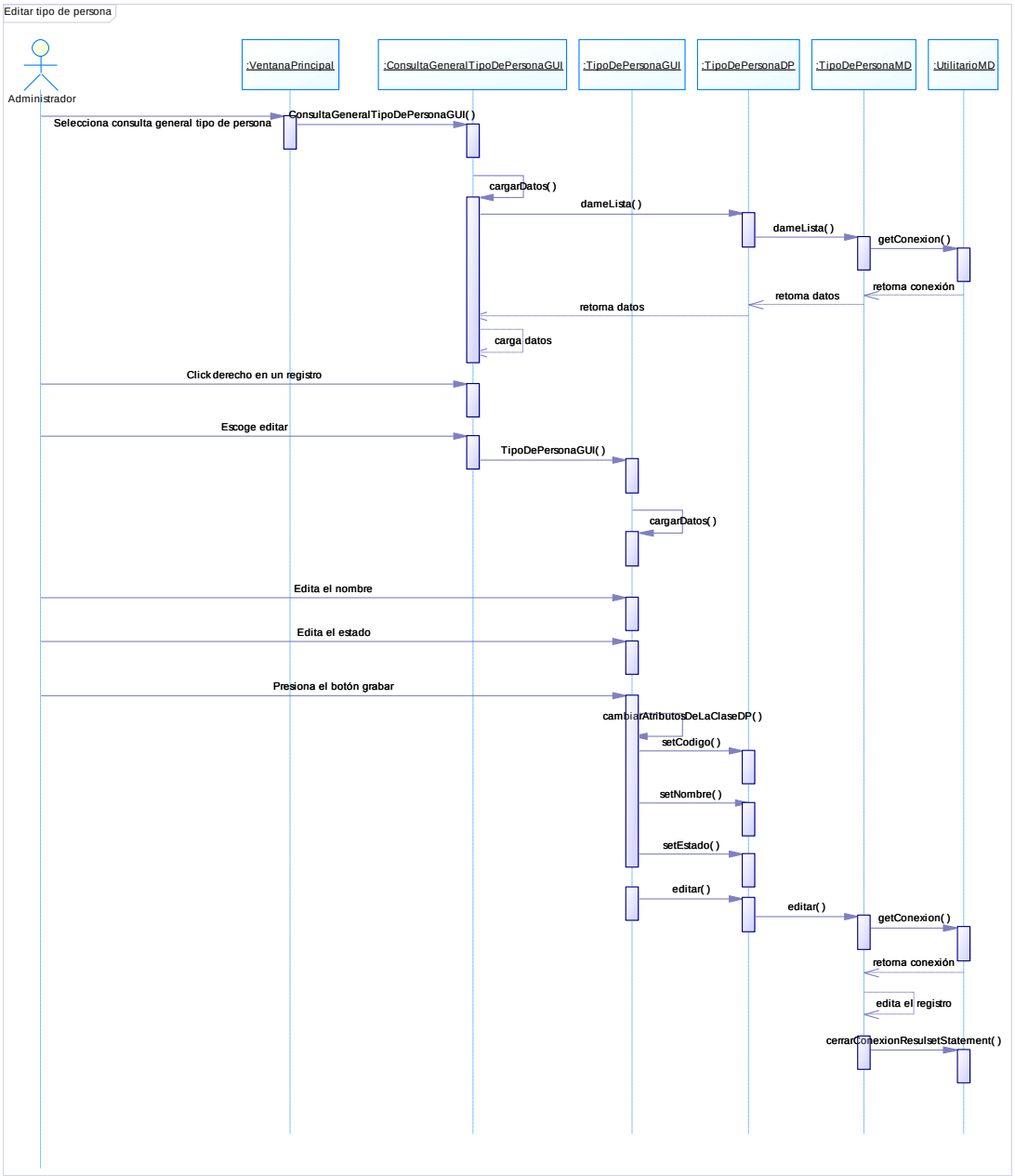
4.5.2.3 Diagramas de secuencia

Para ilustrar los diagramas de secuencia se tomó como ejemplo la administración y la consulta general y por parámetro del módulo tipo de persona. El resto de diagramas se adjuntan en el CD que se anexan al presente trabajo.

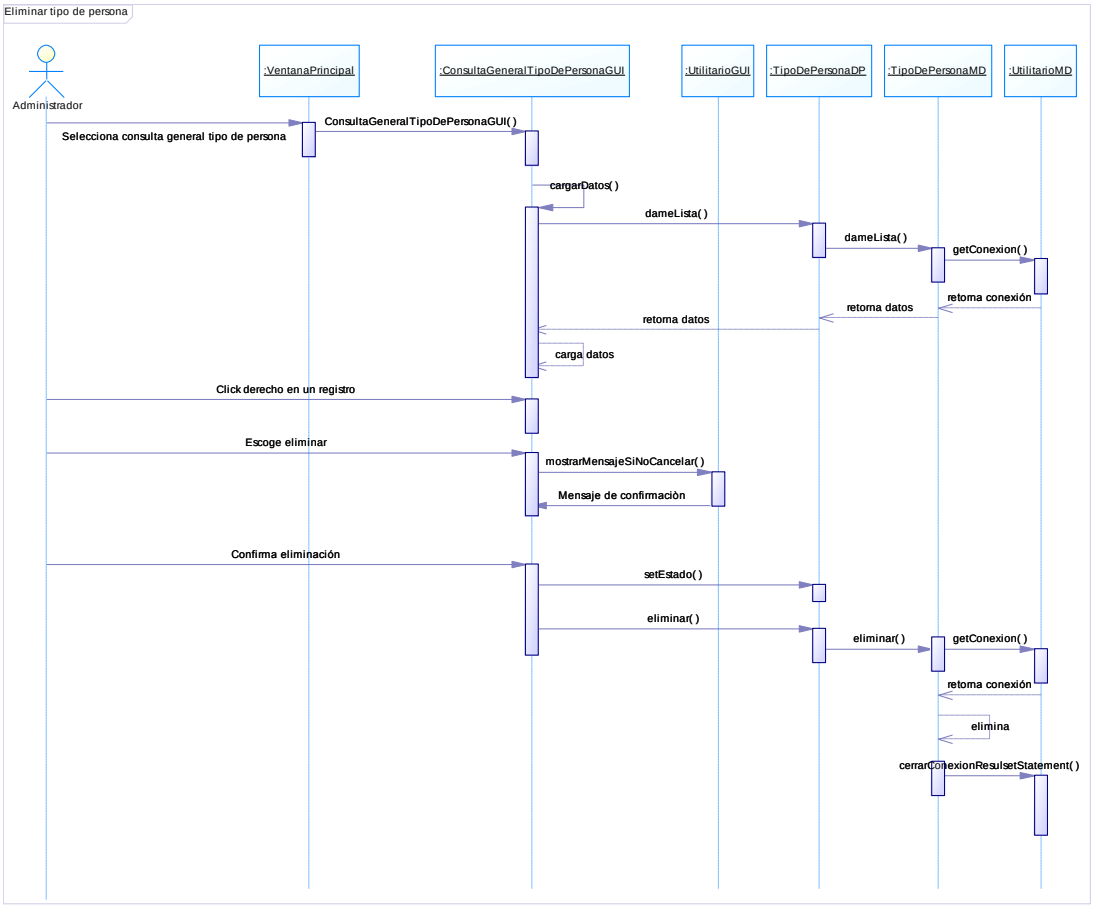
- Ingreso de tipo de persona



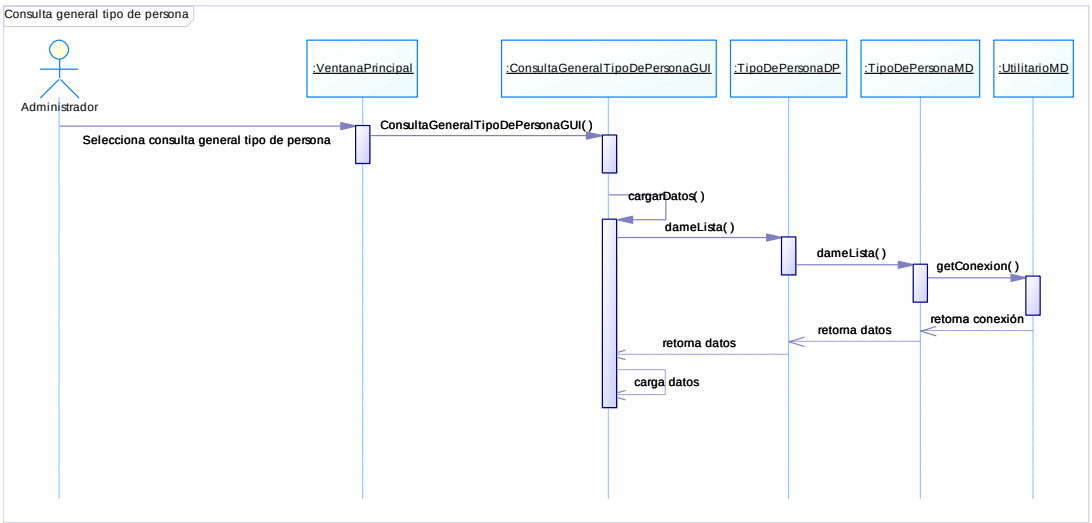
- Editar tipo de persona



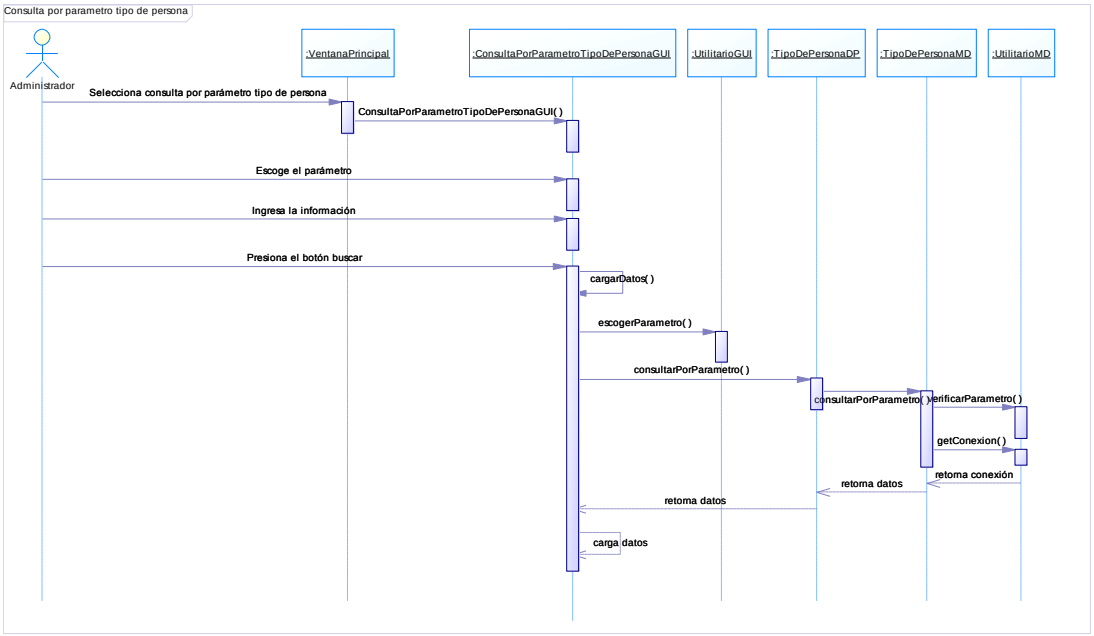
❖ Eliminar tipo de persona



❖ Consulta general de tipo de persona

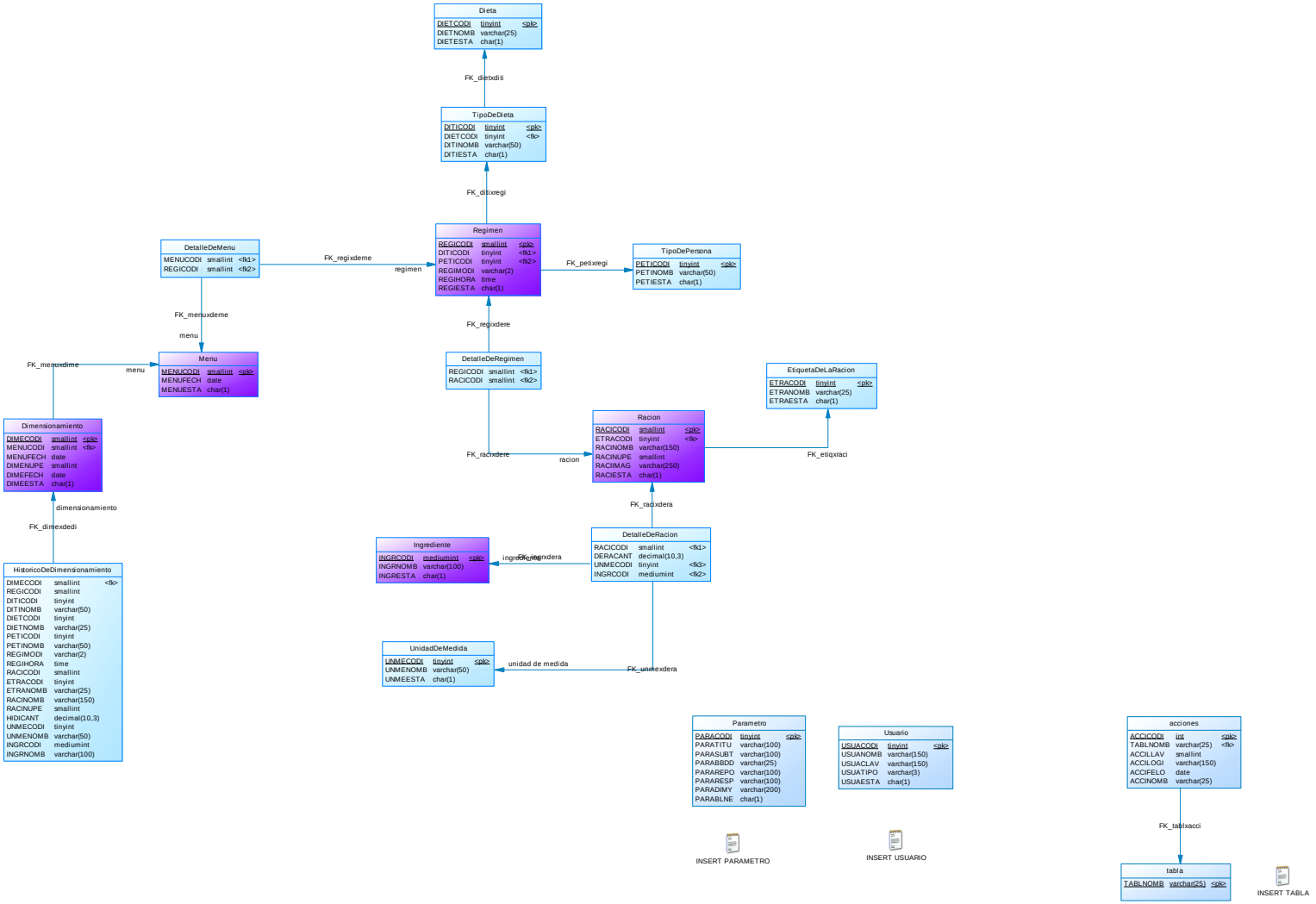


❖ Consulta por parámetro tipo de persona



4.5.2.4 Diagrama entidad relación (base de datos)

A continuación se muestra el diagrama de entidades con sus respectivas relaciones, tipos de datos definidos para el gestor MySQL.


Dieta
DIETCODI tinyint <pk>
DIETNOMB varchar(25)
DIETESTA char(1)
FK_dietxditi
TipoDeDieta
DITICODI tinyint <pk>
DIETCODI tinyint <fk>
DITINOMB varchar(50)
DITIESTA char(1)
FK_ditixregi
Regimen
REGICODI smallint <pk>
DITICODI tinyint <fk1>
PETICODI tinyint <fk2>
REGIMODI varchar(2)
REGIHORA time
REGIESTA char(1)
FK_petixregi
TipoDePersona
PETICODI tinyint <pk>
PETINOMB varchar(50)
PETIESTA char(1)
DetalleDeMenu
MENUCODI smallint <fk1>
REGICODI smallint <fk2>
FK_regixdeme
regimen
FK_menuxdeme
menu
Menu
MENUCODI smallint <pk>
MENUFECH date
MENUESTA char(1)
FK_menuxdime
menu
FK_regixdere
DetalleDeRegimen
REGICODI smallint <fk1>
RACICODI smallint <fk2>
EtiquetaDeLaRacion
ETRACODI tinyint <pk>
ETRANOMB varchar(25)
ETRAESTA char(1)
Racion
RACICODI smallint <pk>
ETRACODI tinyint <fk>
RACINOMB varchar(150)
RACINUPE smallint
RACIIMAG varchar(250)
RACIESTA char(1)
FK_racixdere
racion
FK_etiqxraci
Dimensionamiento
DIMECODI smallint <pk>
MENUCODI smallint <fk>
MENUFECH date
DIMENUPE smallint
DIMEFECH date
DIMEESTA char(1)
dimensionamiento
FK_dimexdedi
FK_racixdera
Ingrediente
INGRCODI mediumint <pk>
INGRNOMB varchar(100)
INGRESTA char(1)
ingrediente
FK_ingrxdera
DetalleDeRacion
RACICODI smallint <fk1>
DERACANT decimal(10,3)
UNMECODI tinyint <fk3>
INGRCODI mediumint <fk2>
HistoricoDeDimensionamiento
DIMECODI smallint <fk>
REGICODI smallint
DITICODI tinyint
DITINOMB varchar(50)
DIETCODI tinyint
DIETNOMB varchar(25)
PETICODI tinyint
PETINOMB varchar(50)
REGIMODI varchar(2)
REGIHORA time
RACICODI smallint
ETRACODI tinyint
ETRANOMB varchar(25)
RACINOMB varchar(150)
RACINUPE smallint
HIDICANT decimal(10,3)
UNMECODI tinyint
UNMENOMB varchar(50)
INGRCODI mediumint
INGRNOMB varchar(100)
UnidadDeMedida
UNMECODI tinyint <pk>
UNMENOMB varchar(50)
UNMEESTA char(1)
unidad de medida
FK_unmexdera
Parametro
PARACODI tinyint <pk>
PARATITU varchar(100)
PARASUBT varchar(100)
PARABBDD varchar(25)
PARAREPO varchar(100)
PARARESP varchar(100)
PARADIMY varchar(200)
PARABLNE char(1)
INSERT PARAMETRO
Usuario
USUACODI tinyint <pk>
USUANOMB varchar(150)
USUACLAV varchar(150)
USUATIPO varchar(3)
USUAESTA char(1)
INSERT USUARIO
acciones
ACCICODI int <pk>
TABLNOMB varchar(25) <fk>
ACCILLAV smallint
ACCILOGI varchar(150)
ACCIFELO date
ACCINOMB varchar(25)
FK_tablxacci
tabla
TABLNOMB varchar(25) <pk>
INSERT TABLA

4.6 Codificación y pruebas del software

Se tomó como ejemplo el código fuente de la administración de tipo de persona. El resto del código se adjunta en un CD.

4.6.1.1 Código fuente de la administración tipo de persona

El código fuente de la administración se encuentra en el ANEXO C.

4.6.1.2 Diccionario de datos:

A continuación se muestra el detalle de cada una de las tablas definidas en el diagrama entidad-relación de la base de datos.

Entidad	Alias	Campo	Tipo	Tamaño (bytes)	Llave primaria	Descripción
Tipo de persona	**PETI**	PETICODI	Tinyint	1	SI	Clave única del tipo de persona.
		PETINOMB	varchar(50)	50	NO	Nombre del tipo de persona.
		PETIESTA	char(1)	1	NO	Estado del tipo de persona.
Dieta	**DIET**	DIETCODI	Tinyint	1	SI	Clave única de la dieta.
		DIETNOMB	varchar(25)	25	NO	Nombre de la dieta.
		DIETESTA	char(1)	1	NO	Estado de la dieta.
Tipo de dieta	**TIDI**	DITICODI	Tinyint	1	SI	Clave única del tipo de dieta.
		DIETCODI	Tinyint	1	NO	Clave foránea de la dieta.
		DITINOMB	varchar(50)	50	NO	Nombre del tipo de dieta.
		DITIESTA	char(1)	1	NO	Estado del tipo de dieta.
Ingrediente	**INGR**	INGRCODI	Mediumint	3	SI	Clave única del ingrediente.
		INGRNOMB	varchar(100)	100	NO	Nombre del ingrediente.
		INGRESTA	char(1)	1	NO	Estado del ingrediente.
Unidad de medida	**UNME**	UNMECODI	Tinyint	1	SI	Clave única de la unidad de medida.
		UNMENOMB	varchar(50)	50	NO	Nombre de la unidad de medida.
		UNMEESTA	char(1)	1	NO	Estado de la unidad de medida.
Etiqueta de la ración	**ETRA**	ETRACODI	Tinyint	1	SI	Clave única de la etiqueta
		ETRANOMB	varchar(25)	25	NO	Nombre de la etiqueta.
		ETRAESTA	char(1)	1	NO	Estado de la etiqueta.
Ración	**RACI**	ETRACODI	Tinyint	1	SI	Clave única de la ración
		ETRANOMB	varchar(25)	25	NO	Nombre de la ración.

		ETRAESTA	char(1)	1	NO	Estado de la ración.
Detalle de la ración	**DERA**	RACICODI	Smallint	2	NO	Clave foránea de la ración.
		DERACANT	decimal(10,3)	10	NO	Cantidad de ingrediente.
		UNMECODI	Tinyint	1	NO	Clave foránea de la unidad de medida.
		INGRCODI	Mediumint	3	NO	Clave foránea del ingrediente.
Régimen	**REGI**	REGICODI	Smallint	2	SI	Clave única del régimen.
		DITICODI	Tinyint	1	NO	Clave foránea del tipo de dieta.
		PETICODI	Tinyint	1	NO	Clave foránea del tipo de persona.
		REGIMODI	varchar(2)	2	NO	Momento del día: desayuno, almuerzo etc.
		REGIHORA	Time	3	NO	Hora del régimen.
		REGIESTA	char(1)	1	NO	Estado del régimen.
Detalle de régimen	**DERE**	REGICODI	Smallint	2	NO	Clave foránea de régimen.
		RACICODI	Smallint	2	NO	Clave foránea de la ración.
Menú	**MENU**	MENUCODI	Smallint	2	SI	Clave única del menú.
		MENUFECH	Date	3	NO	Fecha del menú.
		MENUESTA	char(1)	1	NO	Estado del menú.
Detalle de menú	**DEME**	MENUCODI	Smallint	2	NO	Clave foránea del menú.
		REGICODI	Smallint	2	NO	Clave foránea del régimen.
Dimensionamiento	**DIME**	DIMECODI	Smallint	2	SI	Clave única del dimensionamiento.
		MENUCODI	Smallint	2	NO	Clave foránea del menú.
		MENUFECH	Date	3	NO	Fecha del menú.
		DIMENUPE	Smallint	2	NO	Número de personas.
		DIMEFECH	Date	3	NO	Fecha del dimensionamiento.
		DIMEESTA	char(1)	1	NO	Estado del dimensionamiento.
Histórico de dimensionamiento	**HIDI**	DIMECODI	Smallint	2	NO	Clave foránea de dimensionamiento.
		REGICODI	Smallint	2	NO	Clave de régimen.
		DITICODI	Tinyint	1	NO	Clave del tipo de dieta.
		DITINOMB	varchar(50)	50	NO	Nombre del tipo de dieta.
		DIETCODI	Tinyint	1	NO	Clave de la dieta.
		DIETNOMB	varchar(25)	25	NO	Nombre de la dieta.
		PETICODI	Tinyint	1	NO	Clave del tipo de persona.
		PETINOMB	varchar(50)	50	NO	Nombre del tipo de persona.
		REGIMODI	varchar(2)	2	NO	Momento del día.
		REGIHORA	Time	3	NO	Hora del régimen.
		RACICODI	Smallint	2	NO	Clave de la ración.
		ETRACODI	Tinyint	1	NO	Clave de la etiqueta de la ración.
		ETRANOMB	varchar(25)	25	NO	Nombre de la etiqueta de la ración.
		RACINOMB	varchar(150)	150	NO	Nombre de la ración.
		RACINUPE	Smallint	2	NO	Número de personas de la ración.
		HISTCANT	decimal(10,3)	10	NO	Cantidad dimensionada.
		UNMECODI	Tinyint	1	NO	Clave de la unidad de medida.
		UNMENOMB	varchar(50)	50	NO	Nombre de la unidad de medida.
		INGRCODI	Mediumint	3	NO	Clave del ingrediente.

		INGRNOMB	varchar(100)	100	NO	Nombre del ingrediente.
Usuario	**USUA**	USUACODI	Tinyint	1	SI	Clave única del tipo del usuario.
		USUANOMB	varchar(150)	150	NO	Nombre del usuario.
		USUACLAV	varchar(150)	150	NO	Clave del usuario.
		USUATIPO	varchar(3)	3	NO	Tipo de usuario.
		USUAESTA	char(1)	1	NO	Estado del usuario.
Parámetro	**PARA**	PARACODI	Tinyint	1	SI	Clave única del tipo del parámetro.
		PARATITU	varchar(100)	100	NO	Título del reporte.
		PARASUBT	varchar(100)	100	NO	Subtítulo del reporte.
		PARABBDD	varchar(25)	25	NO	Nombre de la base de datos.
		PARAREPO	varchar(100)	100	NO	Dirección física de los reportes.
		PARARESP	varchar(100)	100	NO	Dirección física del respaldo.
		PARADIMY	varchar(200)	200	NO	Dirección física de MySQL.
		PARABLNE	char(1)	1	NO	Impresión en blanco negro o a color.

Tabla 4-8**:** Diccionario de datos. [A]

4.6.1.3 Plan de pruebas del software.

La siguiente tabla muestra el plan de pruebas aplicado al módulo tipo de persona. El resto de diagramas se adjuntan en el CD que se anexan al presente trabajo.

Entradas	Resultados Esperados	Casos de Uso
1. Seleccionar del Menú Principal la opción de Administración de Tipos de Personas.	Presentar la ventana de Administración de Tipo de personas.	F1
2. Ingreso de Tipos de Personas.	Almacenamiento de datos.	F1.1
Correcto		
Incorrecto	Mensaje de error apropiado.	
3. Selecciona consulta general de tipo de persona.	Presentar información de consulta.	F12.1.1
4. Actualizar datos de tipo de personas.	Almacenamiento de datos.	F1.2
Correcto		

Incorrecto	Mensaje de error apropiado.	
5. Selecciona consulta general de tipo de persona.	Presentar información de consulta.	F12.1.1
6. Eliminar datos de tipo de persona.		F1.3
Correcto	Eliminación de datos.	
Incorrecto	Mensaje de error apropiado.	
7. Consulta de datos		F12.1
General	Presentar información de consulta.	F12.1.1
Por parámetro.	Presentar información de consulta.	F12.1.2

Tabla 4-9: Plan de pruebas aplicado a la administración tipo de persona. [A]

4.6.1.4 Satisfacer requerimientos

Los procedimientos de prueba fueron realizados para verificar que se satisfacen los requerimientos permitiendo determinar que han sido implementados correctamente.

La siguiente tabla muestra el procedimiento realizado para verificar que se cumple el requerimiento del módulo tipo de persona planteado al inicio. El resto de tablas se adjuntan en el CD que se anexan al presente trabajo.

Entradas	Resultados Esperados	Casos de Uso	Cumple
1. Seleccionar del Menú Principal la opción de Administración de Tipos de Personas.	Presentar la ventana de Administración de Tipo de personas.	F1	✔
2. Ingreso de Tipos de Personas.	Almacenamiento de datos.	F1.1	✔

Correcto			
Incorrecto	Mensaje de error apropiado.		
3. Selecciona consulta general de tipo de persona.	Presentar información de consulta.	F12.1.1	✔
4. Actualizar datos de tipo de personas.	Almacenamiento de datos.	F1.2	✔
Correcto			
Incorrecto	Mensaje de error apropiado.		
5. Selecciona consulta general de tipo de persona.	Presentar información de consulta.	F12.1.1	✔
6. Eliminar datos de tipo de persona.		F1.3	✔
Correcto	Eliminación de datos.		
Incorrecto	Mensaje de error apropiado.		
7. Consulta de datos		F12.1	✔
General	Presentar información de consulta.	F12.1.1	✔
Por parámetro.	Presentar información de consulta.	F12.1.2	✔

Tabla 4-10: Procedimiento para verificar el cumplimiento de la administración tipo de persona. [A]

4.6.1.5 Evaluación:

- Viabilidad de la Integración

Para garantizar la viabilidad de la integración de las administraciones se diseñó una estrategia para la apropiada distribución de las clases implementadas. Además de la definición de convenciones de programación.

4.6.1.6 Consistencia

A continuación se muestra en una tabla la consistencia externa; con los requerimientos, y la interna; entre los requerimientos de la clase tipo de persona tomada como ejemplo.

Entradas	Resultados Esperados	Casos de Uso	Viabilidad de Integración	Viabilidad de Mantenimiento	Consistencia
1. Seleccionar del Menú Principal la opción de Administración de Tipos de Personas.	Presentar la ventana de Administración de Tipo de personas.	F1	✔	✔	✔
2. Ingreso de Tipos de Personas.	Almacenamiento de datos.	F1.1	✔	✔	✔
Correcto					
Incorrecto	Mensaje de error apropiado.				
3. Consulta de datos general y por parámetro.	Presentar información de consulta.	F12.1.1	✔	✔	✔
		F12.1.2			
4. Actualizar datos de tipo de personas.	Almacenamiento de datos.	F1.2	✔	✔	✔
Correcto					
Incorrecto	Mensaje de error apropiado.				

5. Consulta de datos general y por parámetro.	Presentar información de consulta.	F12.1.1	✔	✔	✔
	Mensaje de error apropiado.	F12.1.2	✔	✔	✔
Entradas	Resultados Esperados	Casos de Uso	Viabilidad de Integración	Viabilidad de Mantenimiento	Consistencia
6. Eliminar datos de tipo de clientes.	Eliminación de datos.	F1.3	✔	✔	✔
Correcto					
Incorrecto	Mensaje de error apropiado.				
7. Consulta de datos general y por parámetro.	Presentar información de consulta.	F12.1.1	✔	✔	✔
		F12.1.2	✔	✔	✔

Tabla 4-11: Viabilidad de integración y consistencia externa e interna del módulo tipo de persona. [A]

4.7 Integración del software:

4.7.1 Plan de integración

Para la realización del plan de integración se tomo en cuenta la estrategia donde se distribuyó las administraciones para su desarrollo, adicionalmente se consideró que los módulos de Ración, Régimen, Menú necesitan de las clases anteriormente desarrolladas.
Posterior al desarrollo del resto de administraciones se procedió a la integración de las administraciones para finalmente realizar la implementación del proceso de Dimensionamiento de Ración.

A continuación se muestra en una tabla la integración del módulo tipo de persona y sus componentes necesarios distribuidos en las tres capas.

Administración	Método	Componentes
Tipo de Persona	Ingresar	Clases GUI, MD, DP. Utilitario GUI, Utilitario MD.
	Actualizar	Clases GUI, MD, DP. Utilitario GUI, Utilitario MD.
	Eliminar	Clases GUI, MD, DP, Utilitario GUI, Utilitario MD.
	Consultar	Clases GUI, MD, DP, Utilitario GUI, Utilitario MD.

Tabla 4-12: Integración del módulo tipo de persona en las tres capas de desarrollo. [A]

	Semanas			
Actividad	**1ra**	**2da**	**3ra**	**4ta**
Desarrollo Administración Tipo de Personas				
Desarrollo Administración Ingredientes				
Desarrollo Administración Unidad de Medida				
Desarrollo Administración Dieta				
Desarrollo Administración Tipo de Dieta				
Desarrollo Administración Etiqueta de la Ración				
Desarrollo Administración de Ración				
Integración de las administraciones				
Desarrollo Administración de Régimen				
Desarrollo Administración de Menú				
Desarrollo Administración de Dimensionamiento				

Tabla 4-13: Estrategia de integración de los módulos del sistema. [A]

4.7.2 Integrar las unidades

Es de vital importancia la integración de todas las administraciones al igual que la realización de las correspondientes pruebas de integración en las que se certificará el

correcto funcionamiento, adicionalmente es necesario el desarrollo previo de los módulos cumpliendo con la estrategia puesto que como se indicó anteriormente existen administraciones que emplean las clases previamente desarrolladas.

A continuación se tomó como ejemplo el módulo de Ración para ilustrar la integración de las clases que componen dicho módulo. El resto se adjuntan en el CD que se anexa al presente trabajo.

Administración	Capa	Clases Usadas
Ingreso de Ración	GUI	Ración
	GUI	Consulta General Ingrediente
	GUI	Añadir Cantidad de Unidad de Medida
	DP	Ración
	DP	Detalle de Ración
	MD	Ración
	DP	Etiqueta
	MD	Etiqueta
	DP	Ingrediente
	MD	Ingrediente
	DP	Unidad de Medida
	MD	Unidad de Medida
	MD	Utilitario

Tabla 4-14: Integración del módulo ración. [A]

4.7.3 Evaluación

A continuación se detalla el cumplimiento de los estándares de desarrollo de los módulos de del sistema:

	Estándares			
	General	**Diseño de interfaz gráfica**	**Dominio del problema**	**Conexión a la base de datos**
Módulo	✔	✔	✔	✔
Tipo de persona	✔	✔	✔	✔
Ingredientes	✔	✔	✔	✔
Unidad de Medida	✔	✔	✔	✔
Dieta	✔	✔	✔	✔
Tipo de Dieta	✔	✔	✔	✔
Etiqueta la Ración	✔	✔	✔	✔
Ración	✔	✔	✔	✔
Régimen	✔	✔	✔	✔
Menú	✔	✔	✔	✔
Dimensionamiento	✔	✔	✔	✔

Tabla 4-15: Evaluación del cumplimiento de los estándares de desarrollo. [A]

4.8 Pruebas de calificación de software

4.8.1 Calificación de Pruebas

El software será calificado dependiendo si este cumple o no a conformidad con los requerimientos especificados por el usuario y características que el usuario considerará importantes a la hora de usar el sistema.

Aspectos Evaluados por el Usuario	Cumplimiento
Intuitivo.	✔
Interfaces amigables.	✔
Cumplimiento de Requerimientos.	✔
Se puede ingresar nueva información.	✔

Aspectos Evaluados por el Usuario	Cumplimiento
Información susceptible de modificar	✔
Los datos se pueden eliminar.	✔
Permite consultar la información guardada.	✔
Dimensiona las raciones.	✔
Genera un reporte a detalle.	✔

Tabla 4-16: Calificación del sistema por parte del usuario. [A]

4.9 Instalación del software

4.9.1 Plan de Instalación

El plan de instalación de software contiene:

1) Requerimientos técnicos del sistema
2) Características generales del sistema
3) Características específicas
4) Descripción de módulos
5) Descripción del sistema
6) Entrada de datos
7) Consultas y salidas
8) Capacitación de usuarios

Se detalla en el ANEXO D.

4.9.2 Apoyo a la aceptación del software

El apoyo de la aceptación del software conlleva a la creación de un plan de capacitación de usuarios el cual comprende:

1) El sistema
2) Beneficios esperados
3) Unidades relacionadas
4) Presentación de los módulos del sistema
5) Presentación del sistema
6) Procesamiento
7) Presentación de consultas y salidas

8) Actividades anexas

Y se detalla en el ANEXO E. Además de un documento de aceptación de software detallado en el ANEXO F.

4.10 Plan de Verificación

Para asegurar el cumplimiento de la calidad se realizó un ckeck list enumerando los estándares de calidad definidos anteriormente para su comprobación.

4.11 Plan de Validación

El plan de validación incluyó:

- Plan de integración

En este plan se detalló las actividades a realizarse para la correcta integración de los módulos y de esta manera asegurar su funcionamiento. En dicho plan se especificaron responsables y plazos.

- Plan de pruebas

En el plan de pruebas se especifica las entradas realizadas por el usuario, los resultados que el sistema deberá presentar en base a los requerimientos definidos en los casos de uso. Este plan de pruebas se realizó para cada administración implementada.

- Plan de aceptación

En el plan de aceptación se enumeraron los módulos que el sistema maneja. En este plan es el usuario quien verifica y certifica el correcto funcionamiento y la validez del producto mediante el cumplimiento de los requerimientos enumerados por el usuario.

Los planes descritos anteriormente ayudaron a validar que el sistema cumpla con las necesidades y requerimientos del usuario.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

El proyecto realizado ha contribuido de manera muy importante para identificar y resaltar los procesos que maneja el Departamento de Nutrición del Hospital de Niños "Baca Ortiz". En base a dicha definición se consideró en la presente tesis la intervención de la solución informática para la mejora del proceso Preparación y Elaboración de las recetas alimenticias. Al finalizar el trabajo se establecen las conclusiones que se explicarán a continuación:

- Dentro de los puntos que consideramos tienen más importancia dentro de un proyecto de esta naturaleza son el detectar cuáles son las necesidades reales de las personas que trabajarán día a día con el sistema y que los procesos operativos del departamento se apeguen a la realidad del trabajo diario.

- La automatización del proceso Preparación de las recetas alimenticias que incluyó los microprocesos: registro de alimentos utilizados, revisión del menú, calculo de cantidades de cada alimento, registro de cantidades totales por alimento dieron origen al proceso del Dimensionamiento de Raciones que genera un reporte con las cantidades de cada ingrediente necesarias para elaborar una receta para un determinado número de personas.

- El uso de la solución informática permite optimizar el tiempo que invierten los nutricionistas al planificar las dietas, ya que el sistema permite almacenar las recetas, permitiendo que puedan ser incluidas en nuevos menús de manera ágil y eficiente.

- El manejo del sistema SIDRA reduce el tiempo de trabajo del personal que administra los recursos y las cantidades necesarias para la preparación de recetas, haciendo el cálculo de ingredientes utilizados más preciso.

- La aplicación de software permite tener un mejor control de las cantidades de cada ingrediente a ser utilizadas para la preparación de las recetas que formarán cada régimen alimenticio (desayuno, colación, almuerzo, merienda, cena) evitando que exista sobrantes que se convertirán en desperdicios.

- Con la ayuda del sistema el personal del Departamento de Nutrición puede planificar a largo plazo permitiendo realizar presupuestos para la adquisición de insumos y el manejo de un stock mínimo en bodega anticipándose a las necesidades del departamento.

- El uso de un marco de referencia como la norma ISO 12207 permitió establecer una estrategia de desarrollo del proyecto lo que ayudó a definir actividades y tareas que se llevaron a cabo, incluyendo la determinación del ciclo de vida de desarrollo de software es un aspecto crucial, puesto que una selección errónea puede causar el fracaso del proyecto.

- Asimismo, el lenguaje de programación Java, permitió el desarrollo de un software con una interface amigable al usuario, dando como resultado final un programa computacional con características profesionales que permitieron el fácil entendimiento e interacción sistema - usuario.

- Una de las funcionalidades que brinda el sistema consiste en la modificación de las recetas que componen un menú facilitando de esta forma, el cambio oportuno de ingredientes en caso de que no existiesen en bodega.

- Una de las características fundamentales que el sistema posee es el procesamiento de datos automático de la cantidad de ingredientes por régimen alimenticio es decir calcula la cantidad de insumos que se necesitan para elaborar de forma individual o grupal al desayuno, colación, almuerzo, merienda o cena.

5.1.1 Oportunidades de Mejora

El software desarrollado en este proyecto tiene oportunidad de mejorar respecto a incluir otras facilidades como por ejemplo:

- Permitir la obtención de back up cada cierto tiempo como medida de contingencia en caso de que ocurra un fallo tecnológico.

- Registrar en el formulario enviado por el Ministerio de Salud el total de ingredientes utilizados en el menú elaborado, puesto que esta actividad se realiza de forma manual.

- Tener la capacidad de manejar el costo por ingrediente y así determinar el valor total del menú a elaborarse.

- Elaborar reportes y graficas estadísticas para conocer los ingredientes más usados, promedio de personas atendidas en los diferentes momentos del día.

- Integrar el registro de pacientes que ingresan diariamente a las salas del Hospital de Niños "Baca Ortiz" con el Sistema de Dimensionamiento de Raciones, que permitirá de manera automática conocer el número total de pacientes.

- Relacionar la estandarización de porciones por tipo de preparación con su respectivo ingrediente, agilitando el proceso de creación de una receta.

- Mostrar el manual de usuario desde uno de los ítems del Menú Principal, permitiendo al usuario consultar de manera rápida la información requerida cuando lo necesite.

- Permitir designar derechos de usuarios para el acceso al menú principal del sistema, que incluye administración de módulos, procesos y consultas, de esta manera se podrá controlar las actividades ejecutadas por cada usuario permitiendo realizar un seguimiento de la manipulación de la información.

5.2 Recomendaciones

- En el desarrollo de la presente disertación se identificaron cinco procesos que comprenden el funcionamiento del departamento de Nutrición del Hospital de Niños "Baca Ortiz". En este proyecto se automatizó uno de ellos por lo que se recomienda para futuras disertaciones tomar en cuenta el resto de procesos.

  Una vez automatizados los procesos de este departamento se recomienda integrarlos para manejar la información de una manera centralizada, evitando la duplicidad de la información y asegurándose que esta sea confiable.

- Esta solución informática podrá ser aplicada a otras entidades como restaurantes, cuarteles, asilos y demás establecimientos que elaboran menús para un gran número de consumidores.

- Se recomienda para la creación de una ración que el ingreso de la "cantidad de la unidad de medida del ingrediente" sea estandarizada para los insumos líquidos y sólidos por ejemplo litros y libras respectivamente lo que facilitará el cálculo en grandes cantidades.

- Se recomienda la impartición y enseñanza de un mayor número de herramientas CASE, e IDE´S, que permitirán a los estudiantes agilitar el proceso de desarrollo de software a través de la selección adecuada de las herramientas que son claves a la hora de desarrollar el sistema, hasta la implementación y entrega del producto al usuario.

6. BIBLIOGRAFÍA

- Tecnología Orientada a Objetos (POO)
  http://java.ciberaula.com/articulo/tecnologia_orientada_objetos/
  5 de Enero del 2011
- Herramientas UML
  http://www.ingenierosoftware.com/analisisydiseno/uml.php
  7 de Enero del 2011
- Modelado de Sistemas con UML
  http://lucas.hispalinux.es/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf
  8 de Enero del 2011
- Ciclo de Vida en Cascada
  http://www.ia.uned.es/ia/asignaturas/adms/GuiaDidADMS/node10.html
  9 de Enero del 2011
- Desarrollo en Cascada
  http://es.wikipedia.org/wiki/Desarrollo_en_cascada
  9 de Enero del 2011
- Software
  http://es.wikipedia.org/wiki/Software
  9 de Enero del 2011
- Diseño de una Metodología Ágil de Desarrollo de Software.
  http://materias.fi.uba.ar/7500/schenone-tesisdegradoingenieriainformatica.pdf
  9 de Enero del 2011
- Ciclo de vida de Software
  http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_3/lpcu097%20-%2001.pdf
  12 de Enero del 2011
- Metodologías de Desarrollo de Software
  http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html
  12 de Enero del 2011

- Modelo de prototipos
  http://es.wikipedia.org/wiki/Modelo_de_prototipos
  17 de Enero del 2011
- Tipos de Metodologías
  http://www.eumed.net/libros/2009c/584/Metodologias%20Estructuradas%20y%20Metodologias%20Orientadas%20a%20Objetos.htm
  17 de Enero del 2011
- Metodología Orientada a Objetos
  http://profesores.fi-b.unam.mx/carlos/aydoo/intro.html
  23 de Enero del 2011
- Metodología Orientada a Objetos
  http://translate.google.com.ec/translate?hl=es&langpair=en%7Ces&u=http://www.ogcio.gov.hk/eng/prodev/eoom.htm
  23 de Enero del 2011
- Mejores prácticas para el Desarrollo de Software
  ttp://www.taringa.net/posts/apuntes-y-monografias/5853842/Mejores-Practicas-para-el-Desarrollo-de-Software.html
  29 de Enero del 2011
- Metodología para el Desarrollo de Software
  http://ingsw.pbworks.com/Metodolog%C3%ADas-para-el-desarrollo-de-Software
  29 de Enero del 2011
- Organizadores gráficos diagramas de flujo
  http://www.eduteka.org/modulos.php?catx=4&idSubX=116
  9 de Febrero del 2011
- Necesidad de una Metodología
  http://www.ia.uned.es/ia/asignaturas/adms/GuiaDidADMS/node9.html
  9 de Febrero del 2011
- Diseño de bases de datos relacionales
  http://usuarios.multimania.es/cursosgbd/UD4.htm
  18 de Febrero del 2011
- Programación orientada a objetos
  http://algonzalezpoo.wordpress.com/encapsulamiento/
  18 de Febrero del 2011
- Introducción a la Programación Orientada a Objetos
  http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto.Oct98/1_Intro/1_IntPoo.htm#_Toc441217857
  18 de Febrero del 2011

- Fundamentos de Ingeniería de Software
  http://www.slideshare.net/ktyk/uml-casos-de-uso
  27 de Febrero del 2011
- Norma técnica peruana
  http://www.bvindecopi.gob.pe/normas/isoiec12207.pdf
  6 de Abril del 2011
- Relaciones entre casos de uso en UML
  http://www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r11_art4_c.pdf
  6 de Abril del 2011
- Diagramas de estructura estática
  http://www.clikear.com/manuales/uml/diagramasestructuraestatica.aspx
  6 de Abril del 2011
- Modelo de clases
  http://www.dcc.uchile.cl/~psalinas/uml/modelo.html
  7 de Abril del 2011
- Elementos de UML
  http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html
  9 de Abril del 2011
- Crea diagramas de secuencia UML
  http://www.teqi.org/es/node/875
  12 de Abril del 2011
- Diagrama de Casos de uso
  http://www.wikilearning.com/tutorial/desarrollo_orientado_a_objetos_con_uml-diagrama_de_casos_de_uso/6321-5
  13 de Abril del 2011
- Lenguaje UML
  http://jms32.eresmas.net/tacticos/UML/UML09/UML0901.html
  13 de Abril del 2011
- Diagramas de actividad
  http://www.osmosislatina.com/lenguajes/uml/actividad.html
  28 de Abril del 2011
- UML
  http://www.uml.org/
  1 de Mayo del 2011
- Diagrama de Componentes UML
  http://www.scribd.com/doc/2568098/UML-Diagramas-de-actividad
  1 de Mayo del 2011

- Diagrama de estado
  http://www.clikear.com/manuales/uml/diagramasestado.aspx
  1 de Mayo del 2011
- UML Interaction Overview Diagrams
  http://www.altova.com/umodel/interaction-diagrams.html
  1 de Mayo del 2011
- Enfoque Orientado a Objetos
  http://www.mitecnologico.com/Main/ElEnfoqueOrientadoAObjetos
  3 de Mayo del 2011
- Ingeniería de software
  http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE
  15 de Mayo del 2011
- Recursos Tecnológicos
  http://www.alegsa.com.ar/Diccionario/C/8152.php
  18 de Mayo del 2011
- Recursos y capacidades, sistemas y tecnologías de la información
  http://io.us.es/cio2006/docs/000089_final.pdf
  18 de Mayo del 2011
- Ingeniería de software y su relación con las herramientas Case
  http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rea_c_ji/capitulo2.pdf
  22 de Mayo del 2011
- Herramientas CASE
  http://paulchasiluisa.galeon.com/
  23 de Mayo del 2011
- Arquitectura de las herramientas CASE
  http://desarrollodeaplicacionesinformaticas.com/index.php/Analisis-y-diseno-detallado-de-aplic.-informaticas/Tema-12-Herramientas-CASE/4-arquitectura-de-las-herramientas-case.html
  27 de Mayo del 2011
- Embarcadero ER/ Studio
  http://blogs.embarcadero.com/andreanolanusse/modelado-de-bases-de-datos-con-embarcadero-er-studio-un-ejemplo-con-interbase/
  27 de Mayo del 2011
- IBM Rational Data Architect
  http://www.ibm.com/developerworks/ssa/data/library/techarticle/dm0801kokkat/index.htm
  28 de Mayo del 2011

- Sistemas gestores de bases de datos
  http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html
  29 de Mayo del 2011
- Toad Data Modeler
  http://es.scribd.com/doc/6495177/Toad-Data-Modeler
  29 de Mayo del 2011
- Comparación entre sistemas gestores de bases de datos
  htttp://www.monografias.com/trabajos29/comparacion-sistemas/comparacion-sistemas.shtml
  1 de Junio del 2011
- Gestores de bases de datos
  http://wwwdi.ujaen.es/~barranco/publico/ofimatica/tema7.pdf
  1 de Junio del 2011
- Sobre PostgreSQL
  http://www.postgresql.org.es/sobre_postgresql
  1 de Junio del 2011
- Oracle
  http://es.wikipedia.org/wiki/Oracle
  2 de Junio del 2011
- PostgresSQL
  http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/informe.pdf
  3 de Junio del 2011
- NetBeans
  http://es.wikipedia.org/wiki/NetBeans
  4 de Junio del 2011

7. ANEXOS

ANEXO A

Tipos de dietas y alimentos permitidos y prescripción para Nutrición Infantil.

MINISTERIO DE SALUD
HOSPITAL PEDIATRICO "BACA ORTIZ"

TIPOS DE DIETAS

TIPO DE DIETA	DIETA
Normal	Dieta normal
Sonda, Amplia, Estricta	Dieta líquida
Hiperproteínica sin residuos, Hiperproteínica, Sin lactosa	Dieta blanda
Régimen individual, Hiposódica normoproteínica, Hiposódica hipoproteica	Dieta restringida

ANEXO B

Ejemplo de un menú del día en el área de producción y nutrición del departamento.

MINISTERIO DE SALUD PUBLICA

HOSPITAL PEDIATRICO "BACA ORTIZ" — **PRIMERA SEMANA**

AREA DE PRODUCCIÓN Y NUTRICIÓN-GESTION HOTELERA

FECHA: Sábado

MENU DEL DIA

	DESAYUNO		ALMUERZO		MERIENDA
	PERSONAL		Menestrón de lenteja con:		Sancocho de yuca con:z.a,-
*	Café en leche	***SOPA:**	verde,col,fideo macarron,z.a,-	***SOPA:**	arroz,verde, hierbas, leche y-
*	Queso		hierbasa,leche,condimentos		condimentos suficiente.
PAN:	Semi tegral				
JUGO:	Taja de papaya	**ARROZ:**	Blanco	**ARROZ:**	Blanco
	PACIENTES	***CARNE:**		***CARNE:**	Asada
Cereal en leche:	Sémola				
	Pan y queso	**PERSONAL.**		**ACOMPAÑADO**	Guiso de achogchas con:papa
Hiposódicas.	Huevo				queso, leche,hierbas y condim-
COLACION					
*Fruta	guineo-sandía	***POSTRE:**	fruta Natural-granadilla	**CAFÉ:**	Agua aromàtica
*hiposódicas:	pera o manzana				
*Pan y otros:	Pan de dulce	***JUGO:**	Papaya con maracuyá		***PACIENTES***
*Papilla de fruta,	verdura:		***PACIENTES***		
*Cereal en agrio:	Avena	***JUGO:**	Papaya con maracuyá	***Cereal en leche**	Arroz de leche con pasas

DIETAS

	CENA	REGIMEN PARA LA EDAD		REGIMEN PARA LA EDAD	
*ARROZ:	Blanco/pimiento verde	*Sopa normal licuada hiperproteíca con pollo		*Sopa normal licuada hiperproteíca con carne	
*CARNE:	Estofada				
*ACOMPAÑADO:	Majado de verde con	*Arroz:	Amarillo	*Arroz:	Blanco
	cebolla blanca,queso.	*Pollo	Desmenuzado entomatado	*carne	Molida estofada:refrito de
			(cebolla,tomate,pimiento,hierbas)		(cebolla,tomate,pimiento,hierbas
*CAFÉ:	En agua	*Acompañado:	Puré de papa con z. Blanca	*Acompañado:	Enrollado de verde con queso
		*Jugo	Papaya con maracuyá	Cereal en leche:	Arroz de leche con pasas
HIPERPROTEICA SIN RESIDUOS		**HIPERPROTEICA SIN RESIDUOS**		**HIPERPROTEICA SIN RESIDUOS**	
*Cereal en agrio y/o en leche		*Sopa	Menudencias con: z.a, arroz,--	*Sopa	Crema de apio con: menudencia
*Pan tostado:			hierbas y condimentos.		papas,hierbas,condimentos.
*Huevo:		*Arroz	Blanco	*Arroz	Blanco
*Compota de furta:		*Pollo	Al jugo	*Carne:	Bisteck
*Leche de soyo:		*Acompañado:	Puré de papa con z. Blanca	*Acompañado:	Enrollado de verde con queso
		*Jugo.	Papaya con maracuyá	Cereal en leche:	Arroz de leche con pasas
	LIQUIDAS		**LIQUIDAS**		**LIQUIDAS**
*Cereal en leche:		*Sopa licuada hiperproteíca con: pollo		*Sopa hiperproteíca con: carne	
*Huevo :	tibio				
Jugo de:		* Jugo:		*Cereal en leche:	Quinuavena
	HIPOSODICAS		**HIPOSODICAS**		**HIPOSODICAS**
*Igual a la dieta normal: sin sal		*Igual a la dieta normal: sin sal		*Igual a la dieta normal: sin sal	
*compota:		*compota:		*Compota:	

Autorizado por: *……..*

JV/janeth — ***NUTRICIONISTA-SUPERVISIÓN***

ANEXO C

Código fuente de las tres capas de la administración de tipo de persona

Clase GUI

```
/*************************************************
*******************
 * Module:   TipoDePersonaGUI.java
 * Author:  SEP
  * Purpose: Defines the Class
TipoDePersonaGUI.java

*************************************************
******************/
package GUI;

import javax.swing.JOptionPane;

/**
 *
 * @author SEP
 */
public class TipoDePersonaGUI extends
javax.swing.JInternalFrame {

    private ManagerGUI MANAGER;
    private UtilitarioGUI utilitarioGUI;
    private DP.TipoDePersonaDP claseDP;
    private int desde;
    private DP.UsuarioDP usuarioDP;

    /**
     * Constructor en blanco
     */
    public TipoDePersonaGUI() {
    }

    /**
     * Constructor TipoDePersonaGUI
     * @param newMVP
     */
    public TipoDePersonaGUI(GUI.ManagerGUI newMVP)
{
        this.MANAGER = newMVP;
        this.utilitarioGUI = new
GUI.UtilitarioGUI();
        this.usuarioDP = MANAGER.getUsuarioDP();
        this.claseDP = new
DP.TipoDePersonaDP(this.usuarioDP);
        this.initComponents();
    }

    // <editor-fold defaultstate="collapsed"
desc="Manejo del formulario">
    /**
     * Cambia el valor de la bandera "desde" para
identificar desde que ventana fue
     * construida.
     * <p>Desde = 0 --> construido desde VP</p>
     * <p>Desde = 1 --> construido desde
ConsultaGeneralTipoDePersonaGUI</p>
     * @param newDesde
     */
    public void setDesde(int newDesde) {
        this.desde = newDesde;
    }

    /**
     * Devuelve la bandera "desde".
     * @return int
     */
    public int getDesde() {
        return this.desde;
    }

    /***
     * Cargar los datos de la clase DP.
     */
    public void cargarDatos() {
        this.txt_codigo.setText("" +
this.claseDP.getCodigo());

this.txt_nombre.setText(this.claseDP.getNombre());

this.utilitarioGUI.cargarCmbEstado(cmb_estado,
this.claseDP.getEstado());
    }

    /**
     * Prepara el formulario para editar o eliminar
el registro.
     */
    public void prepararParaActualizarOEliminar() {
        this.btn_guardar.setText("Grabar");
        this.btn_eliminar.setEnabled(true);
        this.btn_nuevo.setEnabled(true);
    }

    /**
     * Prepara un nuevo registro.
     */
    public void prepararNuevoRegistro() {
        this.txt_codigo.setText("" +
this.claseDP.getNuevoCodigo());
        this.txt_nombre.setText("");
        this.txt_nombre.requestFocus();
        this.cmb_estado.setSelectedIndex(1);
        this.btn_guardar.setText("Guardar");
        this.btn_eliminar.setEnabled(false);

        this.claseDP = new
DP.TipoDePersonaDP(this.usuarioDP);
    }

    /**
     * Prepara un nuevo registro.
     */
    public void prepararNuevoCodigo() {
        this.txt_codigo.setText("" +
this.claseDP.getNuevoCodigo());
    }

    /**Añade una nueva fila a la tabla.*/
    private void agregarRowData() {
        if (this.desde == 1) {

MANAGER.getConsultaGeneralTipoDePersonaGUI().agrega
rTipoDePersonaDP(claseDP);
        }
    }

    /**Actualiza datos de la fila seleccionada
     * En caso de "Eliminar" los datos de la fila
seleccionada se borra de forma
     * lógica no física.*/
    private void actualizarRowData() {
        if (this.desde == 1) {

MANAGER.getConsultaGeneralTipoDePersonaGUI().actual
izarTipoDePersonaDP(claseDP);
        }
    }// </editor-fold>

    // <editor-fold defaultstate="collapsed"
desc="Clase DP">
    /**
     * Cambia la ClaseDP.
     * @param newClaseDP
     */
    public void setClaseDP(DP.TipoDePersonaDP
newClaseDP) {
        this.claseDP = newClaseDP;
    }

    /**
     * Cambia los atributos de la claseDP.
     */
    private void cambiarAtributosDeLaClaseDP() {

this.claseDP.setCodigo(this.utilitarioGUI.getEntero
(this.txt_codigo));

this.claseDP.setEstado(this.utilitarioGUI.escogerEs
tado(cmb_estado));

this.claseDP.setNombre(this.txt_nombre.getText().tr
im());
    }// </editor-fold>
}
```

Clase DP

```
* Module:  TipoDePersonaDP.java
 * Author:  SEP
 * Purpose: Defines the Class TipoDePersonaDP

*************************************************
******************/
package DP;

/**TipoDePersonaDP */
public class TipoDePersonaDP {

    private int codigoPersona;
    private String nombre;
    private char estado;

/*************************************************
******************
    private MD.TipoDePersonaMD claseMD;
    private DP.UsuarioDP usuarioDP;

    /**Constructor
     * @param newUsuarioDP
     */
    public TipoDePersonaDP(DP.UsuarioDP
newUsuarioDP) {
        this.usuarioDP = newUsuarioDP;
        this.codigoPersona = 1;
        this.nombre = "TIPO";
        this.estado = 'A';
        this.claseMD = new
MD.TipoDePersonaMD(this);
```

```
    }

    // <editor-fold defaultstate="collapsed"
desc="Get - Set">
    /**
     * @return int
     * */
    public int getCodigo() {
        return codigoPersona;
    }

    /** @param newCodigo
     * */
    public void setCodigo(int newCodigo) {
        codigoPersona = newCodigo;
    }

    /**Devuelve el nombre
     * @return String
     */
    public String getNombre() {
        return nombre;
    }

    /** @param newNombre */
    public void setNombre(String newNombre) {
        nombre = newNombre;
    }

    /**
     * @return char
     * */
    public char getEstado() {
        return estado;
    }

    /** @param newEstado
     * */
    public void setEstado(char newEstado) {
        estado = newEstado;
    }// </editor-fold>

    // <editor-fold defaultstate="collapsed"
desc="DP">
    /**Devuelve el objeto UsuarioDP
     * @return DietaDP
     */
    public DP.UsuarioDP getUsuarioDP() {
        return this.usuarioDP;
    }

    /**Cambia el objeto UsuarioDP
     * @param newUsuarioDP
     */
    public void setUsuarioDP(DP.UsuarioDP
newUsuarioDP) {
        this.usuarioDP = newUsuarioDP;
    }// </editor-fold>

    // <editor-fold defaultstate="collapsed"
desc="CRUD">
    /**Retorna nuevo codigo.
     * @return int
     */
    public int getNuevoCodigo() {
        return this.claseMD.getNuevoCodigo();
    }

    /**Verificar
     * @return boolean
     * */
    public boolean verificar() {
        return this.claseMD.verificar();
    }

    /**
     * Verifica sin abrir la conexión.
     * @param conexion
     * @return boolean
     * */
    public boolean
verificarSinAbrirConexion(java.sql.Connection
conexion) {
        return this.claseMD.verificar();
    }

    /**Guardar
     * @return boolean
     * */
    public boolean guardar() {
        return this.claseMD.guardar();
    }

    /**Editar
     * @return boolean
     * */
    public boolean editar() {
        return this.claseMD.editar();
    }

    /**Eliminado lógico.
     * @return boolean
     **/
    public boolean eliminar() {
        return this.claseMD.eliminar();
    }// </editor-fold>

    // <editor-fold defaultstate="collapsed"
desc="Consultas">
    /**Devuelve TreeMap
     * @return
     */
    public java.util.TreeMap<Integer, String>
getTreeMap() {
        return this.claseMD.getTreeMap();
    }

    /**Consulta general
     * LLena los datos en una coleeccion.*/
    public void consultaGeneral() {
    }

    /**
     * @return List
     */
    public java.util.List<DP.TipoDePersonaDP>
dameLista() {
        return this.claseMD.dameLista();
    }

    /**
     * @param parametro
     * @param buscar
     * @return List
     */
    public java.util.List<DP.TipoDePersonaDP>
consultarPorParametro(String parametro, String
buscar) {
        return
this.claseMD.consultarPorParametro(parametro,
buscar);
    }
    // </editor-fold>
}
```

Clase MD

```
/**********************************************
*******************
 * Module:  TipoDePersonaMD.java
 * Author:
 * Purpose: Defines the Class TipoDePersonaMD

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

package MD;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JComponent.*;

/**
 * Define la clase TipoDePersonaMD
 */
public class TipoDePersonaMD {

    private DP.TipoDePersonaDP claseDP;
    private DP.AccionDP accionDP;
    private int sgdb;
    private UtilitarioMD utilitarioMD;
    private Connection conecta;
    private Statement stm;
    private ResultSet rs;
    private final String nombreDeTabla =
"TipoDePersona";

    /**
     * @param personaDP
     * @param usuarioDP
     * @pdOid c52ae2ac-3d66-4f65-a79c-a14b66aea2c4
*/
    public TipoDePersonaMD(DP.TipoDePersonaDP
personaDP, DP.UsuarioDP usuarioDP) {
        this.claseDP = personaDP;
        this.sgdb =
this.claseDP.getUsuarioDP().getSgdb();
        this.utilitarioMD = new UtilitarioMD();
    }

    /**
     * @param personaDP
     * */
    public TipoDePersonaMD(DP.TipoDePersonaDP
personaDP) {
        this.claseDP = personaDP;
        this.accionDP = new
DP.AccionDP(this.claseDP.getUsuarioDP());
        this.sgdb =
this.claseDP.getUsuarioDP().getSgdb();
        this.utilitarioMD = new UtilitarioMD();
    }
```

```
    // <editor-fold defaultstate="collapsed"
desc="CRUD">
    /**
     * Verifica si existe el registro.
     * @return boolean
     * */
    public boolean verificar() {
        boolean i = false;
        try {
            this.conecta =
this.utilitarioMD.getConexion(sgdb);
            this.stm = conecta.createStatement();
            String sentencia = ""
                    + "SELECT * "
                    + "FROM " + nombreDeTabla + " "
                    + "WHERE "
                    + "peticodi=" +
this.claseDP.getCodigo();
            rs = stm.executeQuery(sentencia);
            if (rs.next()) {

this.claseDP.setNombre(rs.getString(2).trim());

this.claseDP.setEstado(rs.getString(3).charAt(0));
                i = true;
            }
        } catch (SQLException e) {

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "verificar");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "verificar");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return i;
        }
    }

    /**
     * Guarda el registro.
     * @return boolean
     * */
    public boolean guardar() {
        boolean band = false;
        try {
            conecta =
this.utilitarioMD.getConexion(sgdb);
            stm = conecta.createStatement();
            String sentencia = ""
                    + "INSERT INTO " +
nombreDeTabla + " values("
                    + this.claseDP.getCodigo() +
",'"
                    + this.claseDP.getNombre() +
"','"
                    + this.claseDP.getEstado() +
"')";
//            System.out.println(sentencia);
            stm.execute(sentencia);
            band = true;

            // <editor-fold
defaultstate="collapsed" desc="Acción guardar">
            if (band) {
                DP.TablaDP tablaDP = new
DP.TablaDP();

tablaDP.setNombre(this.nombreDeTabla);
                this.accionDP.setTabla(tablaDP);

this.accionDP.setLlave(this.claseDP.getCodigo());

this.accionDP.setLogin(this.claseDP.getUsuarioDP().
getLogin());
                this.accionDP.setFechaLogin(new
java.util.Date());
                this.accionDP.setAccion("Guardar");
                this.accionDP.setConexion(conecta);
                if (accionDP.guardar()) {
                    band = true;
                }
            }// </editor-fold>

        } catch (SQLException e) {

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "guardar");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "guardar");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return band;
        }
    }

    /**
     * Elimina el registro de forma lógica no
física.
     * @return boolean
     * */
    public boolean eliminar() {
        boolean i = false;
        try {
            conecta =
this.utilitarioMD.getConexion(sgdb);
            stm = conecta.createStatement();
            String sentencia = ""
                    + "update " + nombreDeTabla + "
"
                    + "set "
                    + "petiesta='" +
this.claseDP.getEstado() + "' "
                    + "where "
                    + "peticodi=" +
this.claseDP.getCodigo() + "";
//            System.out.println(sentencia);
            stm.executeUpdate(sentencia);
            i = true;
        } catch (SQLException e) {

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "eliminar");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "eliminar");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return i;
        }
    }

    /**
     * Editar el registro
     * @return boolean
     **/
    public boolean editar() {
        boolean band = false;
        try {
            this.conecta =
this.utilitarioMD.getConexion(sgdb);
            this.stm = conecta.createStatement();
            String sentencia = "";
            sentencia = ""
                    + "UPDATE " +
this.nombreDeTabla + " "
                    + "set petinomb='" +
this.claseDP.getNombre() + "',"
                    + "petiesta='" +
this.claseDP.getEstado() + "' "
                    + "WHERE "
                    + "peticodi=" +
this.claseDP.getCodigo();
//            System.out.println(sentencia);
            stm.executeUpdate(sentencia);
            band = true;
        } catch (SQLException e) {

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "editar");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "editar");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return band;
        }
    }

    /**
     * Retorna un nuevo codigo.
     * @return NuevoCodigos
     */
    public int getNuevoCodigo() {
        int n = 0;
        try {
            conecta =
this.utilitarioMD.getConexion(sgdb);
            stm = conecta.createStatement();
            String sentencia = ""
                    + "SELECT max(peticodi) FROM "
+ nombreDeTabla;
//            System.out.println(sentencia);
            rs = stm.executeQuery(sentencia);
            if (rs.next()) {
                n = rs.getInt(1);
            }
        } catch (SQLException e) {
```

```
this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "getNuevoCodigo");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "getNuevoCodigo");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return (n + 1);
        }
    }// </editor-fold>

    // <editor-fold defaultstate="collapsed"
desc="Consultas">

    /**
     * Devuelve un TreeMap.
     * @return
     */
    public java.util.TreeMap<Integer, String>
getTreeMap() {
        java.util.TreeMap<Integer, String> treeMap
= new java.util.TreeMap<Integer, String>();
        try {
            conecta =
this.utilitarioMD.getConexion(sgdb);
            stm = conecta.createStatement();
            String sentencia = ""
                    + "SELECT peticodi,petinomb "
                    + "FROM " + nombreDeTabla + " "
                    + "WHERE petiesta='A'";
            rs = stm.executeQuery(sentencia);
            while (rs.next()) {
                treeMap.put(rs.getInt(1),
rs.getString(2).trim());
            }
            //System.out.println(sentencia);
        } catch (SQLException e) {

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "getTreeMap");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "getTreeMap");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return treeMap;
        }
    }

    /**
     * @return List
     */
    public java.util.List<DP.TipoDePersonaDP>
dameLista() {
        java.util.List<DP.TipoDePersonaDP> lista =
new java.util.ArrayList<DP.TipoDePersonaDP>();
        try {
            conecta =
this.utilitarioMD.getConexion(sgdb);
            stm = conecta.createStatement();
            String sentencia = ""
                    + "SELECT * "
                    + "FROM " + this.nombreDeTabla
+ " "
                    + "ORDER BY peticodi ASC";
            rs = this.stm.executeQuery(sentencia);
            DP.TipoDePersonaDP aux;
            while (rs.next()) {
                aux = new
DP.TipoDePersonaDP(claseDP.getUsuarioDP());

aux.setCodigo(rs.getInt("peticodi"));

aux.setNombre(rs.getString("petinomb"));

aux.setEstado(rs.getString("petiesta").charAt(0));
                lista.add(aux);
            }
        } catch (SQLException e) {

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "dameLista");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "dameLista");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return lista;
        }
    }

    /**
     * @param newParametro
     * @param newBuscar
     * @return List
     */
    public java.util.List<DP.TipoDePersonaDP>
consultarPorParametro(String newParametro, String
newBuscar) {
        String buscar =
this.utilitarioMD.verificarParametro(newBuscar,
newParametro);
        String parametro = "peti"+ newParametro;
        java.util.List<DP.TipoDePersonaDP> lista =
new java.util.ArrayList<DP.TipoDePersonaDP>();
        try {
            conecta =
this.utilitarioMD.getConexion(sgdb);
            stm = conecta.createStatement();
            String sentencia = ""
                    + "SELECT * "
                    + "FROM " + this.nombreDeTabla
+ " "
                    + "WHERE " + parametro + buscar
+" "
                    + "ORDER BY peticodi ASC";
//            System.out.println(sentencia);
            rs = this.stm.executeQuery(sentencia);
            DP.TipoDePersonaDP aux;
            while (rs.next()) {
                aux = new
DP.TipoDePersonaDP(claseDP.getUsuarioDP());

aux.setCodigo(rs.getInt("peticodi"));

aux.setNombre(rs.getString("petinomb"));

aux.setEstado(rs.getString("petiesta").charAt(0));
                lista.add(aux);
            }
        } catch (SQLException e) {

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "consultarPorParametro");
        } catch (java.lang.NullPointerException e)
{

this.utilitarioMD.mostrarMensajeDeError(e.getMessag
e(), this.getClass(), "consultarPorParametro");
        } finally {

this.utilitarioMD.cerrarConexionResulsetStatement(c
onecta, rs, stm);
            return lista;
        }
    }
    // </editor-fold>
}
```

ANEXO D

Plan de instalación de software.

PLAN DE INSTALACIÓN DE SOFTWARE

SISTEMA DE DIMENSIONAMIENTO DE RACIONES

Requerimientos técnicos del sistema

Sistema operativo:	Windows XP ó Windows 7
Java:	1.6 ó superior
Base de datos relacional:	MySQL 5.1
RAM:	128 MB mínimo

Características generales del sistema

- Posee interfaz 100% JAVA.
- Siguió el marco de referencia de la norma ISO 12207.
- Contempla el control de acceso a usuarios.
- Maneja módulos administrativos tales como: tipo de personas, dietas, tipos de dieta, unidad de medida, etiquetas, raciones, regímenes, menús.
- Maneja el único módulo de proceso de dimensionamiento.
- Permite la recuperación de información a través de consultas genéricas o por parámetro.

Objetivos específicos del sistema:

- Ordenar y resguardar la información.
- Apoyar en la elaboración de recetas.
- Apoyar en la planificación de menús.
- Generar informe con el detalle de la cantidad de ingredientes necesarios para elaborar un menú.

Descripción de los módulos del sistema:

Módulo		Descripción	Tarea permitidas
Administración	Tipo de persona	Permite la creación de nombre de grupo de personas tales como: pacientes, personal.	Creación, edición y eliminación
	Dieta	Permite crear un nombre de conjunto de alimentos que se consumen.	Creación, edición y eliminación
	Tipo de dieta	Permite crear una sub clasificación de la dieta.	Creación, edición y eliminación
	Ingrediente	Permite crear sustancias sólidas, líquidas que formarán parte en la elaboración de una ración.	Creación, edición y eliminación
	Unidad de medida	Permite la creación de unidades físicas para dimensionar los ingredientes.	Creación, edición y eliminación
	Etiqueta	Crea un nombre que permite la rápida identificación de una ración.	Creación, edición y eliminación
	Ración	Elabora una comida, ración o plato que se servirán las personas.	Creación, edición y eliminación
	Régimen	Permite agrupar las raciones.	Creación, edición y eliminación
	Menú	Agrupa los régimenes en un momento del día.	Creación, edición y eliminación
	Usuario	Permite "crear" personas que operarán el sistema.	Creación, edición y eliminación
Proceso	Dimensionamiento	Permite dimensionar la cantidad de ingredientes necesarios para elaborar un menú.	Creación y eliminación

Consultas	Consultas	Permite la recuperación de información.	Solicitar y recuperar información.

Descripción del sistema

- SIDRA es el sistema principal, puesto que en él descansa la responsabilidad de registrar, resguardar, procesar y entregar toda la información del dimensionamiento de raciones que permite gestionar la operación del Departamento de Nutrición del Hospital de Niños "Baca Ortiz".

- Su orientación es apoyar la ejecución de dimensionar las raciones; recibir y controlar la información de ingresos de menús; efectuar el proceso, que permite suministrar información indispensable para las decisiones de la gestión del departamento de nutrición que involucran utilización de recursos alimenticios vitales tanto para los pacientes del hospital como para el personal, hecho del que dependerá casi toda la decisión y acción relevante para los fines del departamento.

Entrada de datos

El sistema deberá contemplar opciones de ingreso para la siguiente información:

- Tipos de dieta que se recetan.
- Sustancias líquidas y sólidas.
- Porciones de alimento que una persona debe consumir.
- Unidades físicas de medida como litros, gramos, kilogramos etc.
- Recetas de las comidas que se preparan.
- Menús planificados a detalle.

Consultas y salidas

- Listado de tipo de persona.
- Listado de dieta.
- Lista de tipo de dieta.
- Listado de ingrediente.
- Listado de unidad de medida.

- Listado de raciones.
- Listado de menús.
- Listado de régimenes.
- Listado de dimensionamientos.
- Reporte de dimensionamiento.

Relación con otro sistema

- No tiene relación con otro sistema.

Actividades anexas

Poblamiento mínimo de la base de datos:

- Ingreso de usuarios.
- Ingreso de ingredientes.
- Ingreso de unidad de medida.
- Ingreso de etiquetas.
- Ingreso de raciones.
- Ingreso de parámetros: dirección de respaldo, título del reporte, subtítulo del reporte y nombre del reporte.

ANEXO E

Plan de capacitación de usuarios

CAPACITACIÓN DE USUARIOS

SISTEMA DE DIMENSIONAMIENTO DE RACIONES

Involucra un proceso educacional entre el desarrollador y el usuario o los usuarios que operarán el sistema:

Objetivos de la capacitación

- Explicar los requerimientos técnicos que necesita el sistema.
- Dar a conocer las características generales del sistema.
- Describir el funcionamiento del sistema.
- Explicar cada módulo y la función que desempeña dentro del sistema.
- Dar una demostración del funcionamiento del sistema con datos reales.

Sitios de capacitación

- Departamento de Nutrición.

Materiales de capacitación

- Laptop la cual albergue el sistema funcionando al cien por ciento.
- Manuales de operación del sistema.
- Reporte con los casos de uso general del sistema.
- Prototipo de un reporte del proceso.

Sistema

Nombre:	SIDRA
Versión:	1
Icono:	
Dirección física :	C:\Sistemas\SIDRA\
	Archivos dentro de SIDRA:

	SIDRA.exe: es el archivo ejecutable que abre la aplicación. Lib: contiene un conjunto de librerías adicionales que permite el correcto desempeño del sistema. Reportes: contiene los reportes generados y el respaldo del sistema.

Beneficios esperados

- SIDRA será capaz de manejar la información de elementos como: tipos de persona, dietas, tipos de dieta, unidades de medida, etiquetas, raciones, regímenes, menús, usuarios y el dimensionamiento de raciones.

- El proceso de dimensionamiento tendrá la capacidad de generar un dimensionamiento de un menú completo. Es decir, permitir el ingreso de un número de personas (pacientes o personal), el ingreso de un menú; previamente elaborado de un día, y seguidamente generar la cantidad total de ingredientes necesarios para elaborar dicho menú para ese número de personas. La precisión de dimensionamiento dependerá de las porciones para elaborar las raciones.

- El sistema agilitará cálculos que son susceptibles de errores si se los lleva de forma manual. Además del ahorro en tiempo de planificación y estimación de ingredientes necesarios para elaborar las comidas.

Unidades relacionadas

- El sistema estará en operación de forma completa en el Departamento de Nutrición del Hospital de Niños “Baca Ortiz”.

Descripción general del sistema

SIDRA:

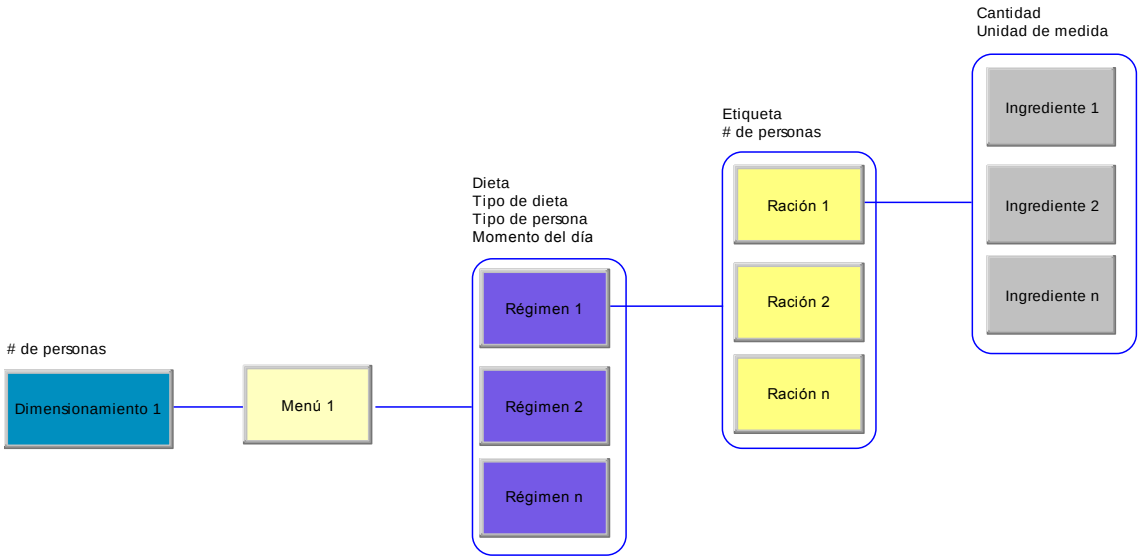
- Posee interfaz 100% JAVA.
- Siguió el marco de referencia de la norma ISO 12207.
- Contempla el control de acceso a usuarios.
- Permite el intercambio de información ágil y rápida.
- Maneja módulos administrativos tales como: tipo de personas, dietas, tipos de dieta, unidad de medida, etiquetas, raciones, regímenes, menús.
- Maneja el único módulo de proceso de dimensionamiento.
- Permite la recuperación de información a través de consultas genéricas o por parámetro.

Presentación de los módulos del sistema

Presentación del sistema

Procesamiento

Para llevar a cabo la ejecución del proceso de dimensionamiento es necesario ingresar toda la información en los módulos de administración. Es decir, ir creando uno a uno los registros de tipos de persona, unidades de medida, ingredientes, etiquetas, raciones, dietas, tipos de dieta, régimenes y menús.
El proceso de dimensionamiento contempla el ingreso de un menú completo para operar:



Como se aprecia en la gráfica anterior, el dimensionamiento se relaciona a un menú, pero el menú contiene un conjunto de régimenes que a su vez contiene un conjunto de raciones con todos los ingredientes que se necesitaron para elaborar dicha ración con su respectiva cantidad y unidad de medida.

Presentación de consultas y salidas

Actividades anexas

Poblamiento mínimo de la base de datos:

- Ingreso de usuarios.
- Ingreso de ingredientes.
- Ingreso de unidad de medida.
- Ingreso de etiquetas.
- Ingreso de la dirección de respaldo, título del reporte, subtítulo del reporte y nombre del reporte.

ANEXO F

Modelo del plan de aceptación de software.

MODELO DEL PLAN DE ACEPTACIÓN DE SOFTWARE

SISTEMA DE DIMENSIONAMIENTO DE RACIONES

SIDRA
Versión 1.0

Este sistema brinda al usuario la capacidad de calcular de manera automática el proceso de dimensionamiento de raciones facilitando el trabajo y haciendo más exacto uno de los procesos más importantes que el Departamento de Nutrición del Hospital "Baca Ortiz" realiza diariamente.

Historial de Revisión

Fecha	Versión	Descripción	Autor	Participantes
2011- 05-23	SIDRA 1.0	Pruebas con el jefe a cargo del Departamento de Informática.	Cristina Matheu Carlos Cargua	Ing. Marcel Quishpe
2011- 05-30	SIDRA 1.0	Verificación de escenarios. • Se realizará todo el menú con el mismo número de personas. • Se realizará los regímenes con diferente número de personas.	Cristina Matheu Carlos Cargua	Ecónomo Santiago Acurio
2011- 06-05	SIDRA 1.0	Pruebas Ecónomo y Directora del Departamento.	Cristina Matheu Carlos Cargua	Ecónomo Santiago Acurio Lda. Jessy Beltrán

Documento del Plan de Aceptación

Se informa que el sistema SIDRA es capaz de manejar la información de:

Actividad	Cumplimiento
Administración:	
Tipo de Personas	
Ingredientes	
Unidad de Medida	
Dieta	
Tipo de Dieta	
Etiqueta de la Ración	
Ración	
Régimen	
Menú	
Proceso	
Dimensionamiento de Raciones	
Consulta General y por Parámetro	
Tipo de Personas	
Ingredientes	
Unidad de Medida	
Dieta	
Tipo de Dieta	
Etiqueta de la Ración	
Ración	
Administración de Régimen	
Administración de Menú	
Administración de Dimensionamiento	
Reporte	
Cantidad de ingredientes necesarios para elaborar un menú.	

La entrega comprendió el levantamiento de información, desarrollo, implementación e implantación de la aplicación por el señor Carlos Cargua y la señorita Cristina Matheu.

ANEXO G

Equivalencias Ingredientes Raciones.

Se anexa en CD.

ANEXO H

Diagramas de Casos de Uso, Actividad, de Secuencia y de Clases.
Se anexan en CD

ANEXO I

Plan de Pruebas de Software

Se anexa en CD

8. GLOSARIO

~ **ISO :** Organismo no gubernamental e independiente creado tras la Segunda Guerra Mundial. Es una red de los institutos de normas nacionales de 160 países, sobre la base de un miembro por país, con una Secretaría Central en Ginebra coordina el sistema.

Esta encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional con el propósito de facilitar el comercio, el intercambio de información y contribuir con normas comunes al desarrollo y a la transferencia de tecnologías.

~ **HTML** (**HyperText Markup Language)** *Lenguaje de Marcado de Hipertexto* es usado para la elaboración de páginas web, para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>), también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

~ **Visual Basic .NET** (**VB.NET**) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. El lenguaje VB.NET no es compatible hacia atrás con Visual Basic, pero el manejo de las instrucciones es similar a versiones anteriores de Visual Basic, facilitando así el desarrollo de aplicaciones más avanzadas con herramientas modernas.

~ **C# :** Es un lenguaje orientado a objetos moderno, simple y poderoso. Fue desarrollado por Microsoft y posteriormente estandarizado. Fue modelado principalmente en base a C++ y Java.
C# es el lenguaje que Microsoft desarrollo principalmente para la plataforma .Net. Para su creación se usaron conceptos de C, C++, Smalltalk, Modula 2 y Java.

Fue creado para desarrollar aplicaciones orientadas a objetos. Incorpora las características de un lenguaje de última generación. Está en continuo desarrollo y tiene el soporte de una de las empresas más grandes del sector.

~ **JAVA:** Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Entre el año 2006 y 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es

ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

~ **ANSI :** Es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos. ANSI es miembro de la Organización Internacional para la Estandarización (ISO) y de la Comisión Electrotécnica Internacional (International Electrotechnical Commission, IEC). La organización también coordina estándares del país estadounidense con estándares internacionales, de tal modo que los productos de dicho país puedan usarse en todo el mundo.

ANSI acredita a organizaciones que realizan certificaciones de productos o de personal de acuerdo con los requisitos definidos en los estándares internacionales. Los programas de acreditación ANSI se rigen de acuerdo a directrices internacionales en cuanto a la verificación gubernamental y a la revisión de las validaciones.

~ **RTF(Rich Text Format): Formato de texto enriquecido** es un formato de archivo informático desarrollado por Microsoft en 1987 para el intercambio de documentos multiplataforma. La mayoría de procesadores de texto son capaces de leer y escribir documentos RTF.

RTF es un formato de 8 bits. Debido a ello tiene el mismo límite que ASCII, pero RTF puede codificar caracteres más allá de ASCII mediante secuencias de escape. Los caracteres escape son de dos tipos: escapes de página de código y escapes Unicode.

~ **GNU GPL(General Public License)** Licencia Pública General de GNU fue creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

~ **OMG (Object Management Group):** *Grupo de Gestión de Objetos*) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas. El grupo está formado por diversas compañias y organizaciones con distintos privilegios dentro de la misma.

~ **Release:** Es la distribución de un software, su documentación y materiales de soporte. Es una versión de un software, pero no todas las versiones del software son release. La versión release es una versión de lanzamiento, es decir, que el software se hace público. En ocasiones una versión del software puede ser una release candidate o candidata a lanzamiento; es decir, es una versión previa al lanzamiento definitivo de dicho software .

~ **Trigger** (disparador) Es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación en una Base de Datos. Dependiendo de la base de datos, los triggers pueden ser de inserción (INSERT), actualización (UPDATE) o borrado (DELETE). Algunas bases de datos pueden ejecutar triggers al crear, borrar o editar usuarios, tablas, bases de datos u otros objetos.

~ **XML** (*e**X**tensible **M**arkup **L**anguage*) Lenguaje de marcas extensible es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades

Propone un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo.

~ **C/C++ C++** Es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Se suele decir que el C++ es un lenguaje de programación multiparadigma puesto que se añadieron facilidades de programación genérica al igual que programación estructurada y programación orientada a objetos.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

~ **Perl:** Es un lenguaje de programación diseñado que toma características del lenguaje C, del lenguaje interpretado shell (sh), AWK, sed, Lisp y, en un grado inferior, de muchos otros lenguajes de programación.

Estructuralmente, Perl está basado en un estilo de bloques como los del C o AWK, y fue ampliamente adoptado por su destreza al procesar el texto y no tener ninguna de las limitaciones de los otros lenguajes de script.

~ **Python:** Es un lenguaje de programación de alto nivel cuya filosofía hace énfasis en una sintaxis muy limpia y que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y multiplataforma.

~ **Tcl:** Es un lenguaje multiplataforma interpretado, y su código puede ser creado y modificado dinámicamente. Sus reglas sintácticas son extremadamente simples y posee reglas de alcance dinámico. Permite escribir código fácil de mantener. Los "scripts" Tcl son a menudo más compactos y legibles que los programas funcionalmente equivalentes en otros lenguajes de programación. Es un lenguaje

~ **PHP** *(Hypertext Pre-processor)* inicialmente *PHP Tools*, o, *Personal Home Page Tools*). Fue creado originalmente por Rasmus Lerdorf en 1994, sin embargo la implementación principal de PHP es producida ahora por The PHP Group. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

PHP tiene un gran parecido con lenguajes de programación estructurada, como C y Perl.

~ **Lisp(LIS**t **P**rocessing**):** Proceso de LIStas es una familia de lenguajes de programación de tipo funcional, Lisp es el segundo más viejo lenguaje de programación de alto nivel.

Lisp fue creado originalmente como una notación matemática práctica para los programas de computadora, basada en el cálculo lambda Se convirtió rápidamente en el lenguaje de programación favorito en la investigación de la inteligencia artificial (AI).

Como uno de los primeros lenguajes de programación, Lisp fue pionero en muchas ideas en ciencias de la computación, incluyendo las estructuras de datos de árbol, el manejo de almacenamiento automático, tipos dinámicos, y el compilador auto contenido.

~ **Scheme:** Es un lenguaje funcional y un dialecto de Lisp. Fue desarrollado en la década de los setenta e introducido en el mundo académico a través de una serie de artículos conocidos como los Lambda Papers de Sussman y Steele.

Scheme proporciona el mínimo número posible de nociones primitivas, construyendo todo lo demás a partir de un reducido número de abstracciones.

~ **GUI** (Graphic User Interface) Interfaz Gráfica de Usuario: Conjunto de formas y métodos que permiten la interacción de un sistema con los usuarios utilizando formas gráficas como botones, íconos, ventanas, fuentes, que representan funciones, acciones e información.

Es una evolución de la línea de comandos tradicional (CLI) de los primeros sistemas operativos como la familia de sistemas MS-DOS.

~ **IBM (**International Business Machines**)** : Es una empresa multinacional estadounidense que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

Tiene una presencia principal en prácticamente todos los segmentos relacionados con las tecnologías de la información.

~ **BSD (**Berkeley Software Distribution**):**De esta forma se llamaron a las distribuciones de código fuente que se hicieron en la Universidad de Berkeley en California y que en origen eran extensiones del sistema operativo UNIX de AT&T Research. Varios proyectos de sistemas operativos de código abierto tienen su origen en una distribución de éste código conocida como 4.4BSD-Lite.

~ **.NET:** Es un framework de Microsoft que hace énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

~ **SQL(***structured query language***)** El lenguaje de consulta estructurado es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar

de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre ella.

~ **ODBC** Open DataBase Connectivity: Es un estándar de acceso a bases de datos desarrollado por SQL Access Group en 1992, el objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos, ODBC logra esto al insertar una capa intermedia (CLI) denominada nivel de Interfaz de Cliente SQL, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda

~ **Qt:** Es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con una interfaz gráfica de usuario, también es usada para el desarrollo de programas sin interfaz gráfica como herramientas para la línea de comandos y consolas para servidores.

9. REFERENCIAS

Documentos Impresos

[A] Desarrollo e Implementación del proceso que maneja el departamento de nutrición del hospital de niños Baca Ortiz

[B] Norma Técnica Peruana NTP – ISO/IEC 12207 : 2006

Páginas Web

[1] Organizadores Gráficos – Diagramas de Flujo
http://www.eduteka.org/modulos.php?catx=4&idSubX=116
7 de Abril del 2011

[2] Uml_diagram
http://es.wikipedia.org/wiki/Archivo:Uml_diagram-es.svg
8 de Abril del 2011

[3] Modelo físico de un sistema Orientado a Objetos
http://www.clikear.com/manuales/uml/modelofisico.aspx
9 de Abril del 2011

[4] Diagrama de Despliegue UML 2
http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html
10 de Abril del 2011

[5] Diagrama de Tiempo UML 2

http://www.sparxsystems.com.ar/resources/tutorial/uml2_timingdiagram.htm

10 de Abril del 2011

[6] Mantenimiento de software

http://alarcos.inf-cr.uclm.es/per/fruiz/cur/mso/trans/s3.pdf

4 de Mayo del 2011

[7] Ciclo de vida de software

http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_3/lpcu097%20-%2001.pdf

4 de Mayo del 2011

[8] Modelo de ciclo de vida en espiral

http://4.bp.blogspot.com/_bnp3w2DoBM/TBZrudzS4bI/AAAAAAAAABQ/sV48hGI462g/s1600/ciclo+de+vida+en+espiral.png

6 de Mayo del 2011

[9] Ciclo de vida de software

http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_3/lpcu097%20-%2001.pdf

8 de Mayo del 2011

[10] Eclipse (Software)

http://es.wikipedia.org/wiki/Eclipse_%28software%29

5 de Junio del 2011