Universidad de las Ciencias Informáticas

Facultad 6

Título: Sistema de gestión integral de la planificación y control del servicio de comedores UCI: análisis y diseño del módulo “Gestión de Comensales”.

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

Autores:

Ariel Hernández Cejas

Yurielkis Matos Pérez

Tutora:

Lic. Maybel Anido Bada

Habana, Junio 2008

“Año 50 del triunfo de la revolución”

El éxito de los hombres no se mide por su éxito inmediato, sino por su éxito definitivo:- no se mide por el dinero que acumularon, sino por el resultado de sus obras.

José martí

DECLARACIÓN DE AUTORÍA

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

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

Yurielkis Matos Pérez	Ariel Hernández Cejas	Lic. Maybel Anido Bada
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

Agradecimientos

Al concluir la realización de nuestro trabajo de diploma, no queríamos pasar por alto los que de una forma u otra contribuyeron a que el mismo se desarrollara de forma satisfactoria.

Un agradecimiento especial a nuestra tutora Maybel Anido Bada por su valiosa asesoría y el tiempo que nos ha dedicado en todo el desarrollo del proyecto.

Agradecimientos a todo el equipo de profesores y trabajadores que han contribuido al desarrollo del proyecto" Alimentos UCI". Al profesor Alieski Sarmiento Almenares por su valiosa cooperación. A nuestros amigos y compañeros de estudios por su apoyo incondicional y por estar siempre presentes.

Un eterno agradecimiento a nuestros familiares que han estado atentos al desarrollo de este empeño y que nos han brindado un amplio apoyo.

A los que de una forma u otra contribuyeron a la elaboración de este trabajo.

A todos

¡Muchas Gracias!

Dedicatoria

Dedico este Trabajo de Diploma a mis Padres Angel Luis Matos Orduñez e Isolina Pérez Campos por ser mi fuente de inspiración y vida.

A mis hermanos, que sigan este ejemplo.

A mis abuelos Benigno Pérez Pérez, Noemí Orduñez Orduñez, Angel Matos Matos y Zoila Campos Cervantes, por su cariño y apoyo constante.

A mis tíos, primos y toda la familia en sentido general que tanto han hecho por mí para que pudiera llegar a este día.

A Indira, Ariel y todos mis amigos por estar siempre presente.

A todos los que han esperado ansiosos este día.

Yurielkis Matos

A mis padres Aracelis Cejas y Juan Hernández por brindarme su amor, cariño, comprensión y apoyo constante. ¡Gracias por existir!

A mi hermana Arelys por ser tan especial para mí… te quiero mucho huesito!

A mis amigos de la universidad y compañeros de aula por los momentos compartidos… siempre estarán en mi corazón.

En especial a mis amigos de siempre: Indira, Yurielkis, Alain y Jesús.

A todos los que de una manera u otra han transcurrido por mi vida dejando una huella que el tiempo no podrá borrar.

Ariel Hernández

Resumen

La Vicerrectoría de Logística (VRL) en la Universidad de las Ciencias Informáticas (UCI) cuenta con la Dirección General de Alimentos (DGA), en la que se llevan a cabo diferentes procesos, los cuales son fundamentales para un correcto desarrollo logístico en la UCI.

La DGA tiene definido el proceso de Planificación y Control del Servicio de Comedores en el cual intervienen los siguientes subprocesos: Planificación de Menú, Gestión de Comensales y Análisis Económico, los cuales requieren mucho tiempo, recursos y están expuestos a constantes reajustes.

Actualmente la Planificación de Comensales representa un punto crítico en el desarrollo de este proceso debido a que no es eficiente ni conveniente para la rapidez que los eventos asociados requieren introduciendo así algunas imprecisiones; por lo que la DGA requiere una herramienta que los asista en este proceso.

Esta investigación tiene el objetivo de realizar el análisis y diseño de la herramienta que permita la gestión de comensales en el *Sistema de Gestión Integral de la Planificación y Control del Servicio de Comedores UCI*.

Palabras claves: Dirección General de Alimentos, menú, comensales.

Índice

Índice de tablas

Índice de figuras

Introducción

La VRL en la UCI cuenta con una Dirección en la que se llevan a cabo diferentes procesos clasificados en: Estratégicos, Claves y de Apoyo (Ver anexo 1). Dentro de los procesos Claves se encuentra definido el servicio de comedores y a su vez este agrupa varios subprocesos relacionados, entre ellos se encuentra la planificación y control, el cual da origen a esta investigación.

La planificación y el control del servicio de comedores cubre varias actividades como:

- Confección del menú a partir de las existencias.
- Cálculo de la cantidad de alimento a utilizar.
- Planificación de la cantidad de comensales.
- Reajuste de la planificación diaria.
- Asignación de alimentos para un periodo.

Estas actividades están vinculadas principalmente a dos áreas dentro de la VRL, específicamente a la DGA y ATM, siendo la DGA un cliente de ATM. (Ver anexo 2)

Los comedores de la UCI prestan servicio en los eventos establecidos (desayuno, almuerzo, comida y meriendas), cuenta con tres complejos de comedores para la distribución de los estudiantes y trabajadores debido al número considerable de usuarios. Actualmente este servicio presenta algunos problemas de eficiencia, por ejemplo:

- Se cometen varios errores a la hora de planificar los comensales debido a que este proceso se realiza manualmente.
- Existe descentralización de la información referente a la planificación de comensales, que en ocasiones ni siquiera se registra, lo que hace imposible obtener indicadores estadísticos de los procesos.
- Las decisiones que se toman hoy se fundamentan con la experiencia y apreciación de los directivos del sistema logístico, lo que trae como consecuencia que el proceso se lleve a cabo de una manera más lenta e ineficiente.

- Existen irregularidades en ATM, planificando en muchas ocasiones con existencias inciertas o escasas, lo que provoca que halla que realizar continuos reajustes en la planificación del menú y de los comensales.
- La generación de reportes e informes es lenta y con diversos formatos, por lo que se hace muy difícil controlar los recursos.
- Un punto crítico en el proceso de planificación es el cálculo de la cantidad de comensales, la cual se lleva a cabo sin utilizar métodos de pronósticos.

Todo esto influye en que diariamente ocurran pérdidas cuantiosas de alimentos, originando una situación en el área que atenta contra el correcto funcionamiento económico de la Universidad. En el curso 2005-2006 las pérdidas en raciones de comida se aproximan a la cifra de 1 720 000.

En la actualidad existen varias empresas con grandes experiencias en el proceso de desarrollo de software de planificación, confiables y de alta calidad. La mayoría de estas empresas para comercializar su software los venden a precios muy elevados, comprarlos y mantenerlos crea dependencias con el proveedor, con costos adicionales.

En nuestro país algunas empresas e instituciones se proyectan en este sentido, como resultado se han desarrollado algunos software, pero su uso en nuestra universidad no sería factible, pues no resolverían todos los problemas a los que se enfrenta la DGA.

Se puede concluir que el proceso de gestión integral de la planificación y control de los comensales no fluye de manera eficiente, por lo que se identifica como **problema científico**: ¿Cómo la gestión de comensales contribuye a la planificación del menú en la UCI?

El problema planteado se enmarca en el **objeto de estudio**: Los procesos de gestión de la información en el Servicio de Comedores.

El objeto delimita el **campo de acción**: Los procesos de gestión de la información en la planificación del menú a partir de la gestión de comensales en la UCI.
Para dar solución al problema se define como **objetivo general**: Desarrollar el análisis y el diseño del módulo “Gestión de Comensales” para el Sistema de gestión integral de la planificación y el control de servicios de comedores en la UCI.

Objetivos Específicos:

- Analizar los procesos que se llevan a cabo en la gestión de comensales del sistema de gestión integral de la planificación y el control de servicios de comedores en la UCI.
- Identificar las funcionalidades que debe cumplir el Módulo: Gestión de Comensales.
- Diseñar el Módulo: Gestión de Comensales.

Para lograr los objetivos se desarrollarán varias **tareas** como:

1. Realización de entrevistas al cliente para lograr la familiarización con el proceso de gestión de comensales del sistema de gestión integral de la planificación y el control del servicio de comedores en la UCI.
2. Estudio del Informe del Proyecto de Gestión Integral del servicio de comedores.
3. Estudio de los Sistemas de Gestión de la Información para la planificación y control del servicio de comedores.
4. Estudio de las tecnologías que se emplearán para el desarrollo de la investigación.
5. Selección de las tecnologías que se emplearán para el desarrollo de la investigación.
6. Modelación del proceso de gestión de comensales.
7. Obtención, descripción y validación de los requerimientos.
8. Realización del diseño de las clases candidatas a la implementación.

El documento esta estructurado de la siguiente manera:

Capítulo 1. Fundamentación teórica.

El capítulo presenta un análisis teórico sobre las diferentes tendencias informáticas que existen para la gestión de comensales. Se describe la justificación de las tecnologías y metodologías utilizadas para la solución de la problemática planteada.

Capítulo 2. Características del sistema.

En este capítulo se hace una definición del objeto de estudio del problema, se plantean los objetivos estratégicos de la organización y los procesos del negocio que los soportan. Se describen los procesos que serán objeto de automatización, así como el funcionamiento general de la propuesta del sistema.

Se define el Modelo del Negocio. Se especifican de los requisitos y casos de uso del sistema.

Capítulo 3. Análisis y diseño del sistema.

En este capítulo se obtienen los artefactos correspondientes al flujo de trabajo “Análisis y Diseño”. Se modela el diseño, donde se representan las realizaciones de los casos de uso del sistema, a través de las clases del diseño con extensiones Web además de las descripciones de las mismas.

Capítulo 1: Fundamentación Teórica

El capítulo presenta un análisis teórico sobre las diferentes tendencias informáticas que existen para la gestión de comensales. Se describe la justificación de las tecnologías y metodologías utilizadas para la solución de la problemática planteada.

1.1 Aplicaciones Informáticas relacionadas con el campo de acción.

1.1.1 Sistemas de Gestión de la Información.

La información es un elemento vital para el desarrollo, por lo que a partir de la década de los años 80 su gestión ha ocupado un espacio fundamental en la economía de los países. Debe gestionarse de manera eficiente para poder usarla en la toma de decisiones y que estas sean cada vez más acertadas.

1.1.2 Sistemas similares a nivel internacional.

La planificación y el control de los servicios que se prestan en los comedores, al que acceden grandes cantidades de comensales, se hace cada vez más necesaria. En el mundo ya existen empresas que se dedican al desarrollo de aplicaciones que contribuyan a optimizar estos procesos. Algunos ejemplos son:

SATN2007*, de* Cadre Distribución S.L que está enfocado fundamentalmente al área de la salud. Este software cuenta con una comprensiva base de información y conocimiento de los alimentos, proporciona análisis completos de dietas, recetas y menues.

EquiLibra Professional*,* es un software que proporciona más de 1500 platos nacionales e internacionales con sus respectivos componentes nutricionales y la preparación culinaria. Dispone de un mecanismo de intercambios nutricionales utilizando todos los platos y alimentos del sistema, lo que permite personalizar dietas al gusto y la disponibilidad de los alimentos. Tiene un acceso automático a búsquedas de alimentos y platos preparados de la base de datos para equilibrar en una forma fácil y automática su ingesta, de esta manera poder suplementar el déficit a través de los alimentos adecuados; maneja y controla en forma histórica un número ilimitado de usuarios o grupos de usuarios, entre otras funcionalidades.

La empresa *DMS S.A.C* lanzó el 14 de septiembre del 2007 soluciones desarrolladas para Control de Asistencia ***VISUAL ASIST XXI*** y Control de Comedores ***COMSOFT***. Estas soluciones son utilizadas con éxito por gran parte de las empresas peruanas y en el extranjero.

Otra empresa especializada en la prestación de servicios de alimentación y administración de comedores industriales es la *Acoyci* (Administración de Comedores y Cafetines Industriales), C.A.

ASSETS NS es un sistema comercializado por la firma panameña D' Marco S.A. Este es un sistema de Gestión Integral estándar que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos. Dispone, además, de métodos novedosos para administración y planificación de inventarios, así como una amplia gama de Análisis y Consultas que le permitirán no sólo conocer exactamente la situación actual, sino proyectar decisiones futuras. En él se facilita el uso de funcionalidades para adaptarse a las exigencias de cada entidad en particular, garantizando que sus reportes tengan la forma y el contenido que el usuario les defina. Este sistema está diseñado con una estructura organizativa a varios niveles, en la que podrán existir: Grupo Corporativo, Grupo de Agrupaciones, Almacenes y Centros de Costos. Para entidades con esta estructura se brinda un Módulo de Comunicaciones que facilita poder intercambiar información entre ellas, con el fin de consolidar la información sobre la Gestión Comercial y Contable, pudiéndose obtener los Estados Financieros, Resúmenes de Compras, Ventas, entre otros, a distintos niveles. En Cuba ASSETS es utilizado fundamentalmente en grandes almacenes. En la UCI se usa en ATM debido que es el almacén central, en el cual están concentrados la mayor parte de los productos con los que cuenta la universidad.

Ninguna de estas aplicaciones informáticas pueden ser utilizadas en nuestra universidad debido a que no cubren las funcionalidades del proceso de planificación y control del servicio de comedores en la UCI, ejemplo: no contemplan la relación entre los comensales y el menú, no tienen implementada una funcionalidad que permita planificar los comensales. O sea, que usando alguna de estas herramientas la DGA continuaría sin resolver los problemas a los que se enfrenta.

1.1.3 Sistema de gestión de la información a nivel nacional.

En Cuba algunas instituciones se han proyectado hacia el desarrollo de software que contribuya a lograr una buena planificación y control de los servicios de comedores. En el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE), por ejemplo, se utiliza el Sistema de Control de Comedores, *SISCOMED*. Este usa un sistema de tarjetas magnéticas, en el cual los usuarios registrados deben presentarla antes de ser atendidos. Además, permite obtener la cantidad de comensales registrados que reciben cada servicio de manera diaria o mensualmente, en los comedores incluyendo los comensales registrados que reciben doble ración en el mismo evento (desayuno, almuerzo, comida) del día. Este sistema no pudo ser usado en nuestra universidad debido a que: existen grandes diferencias entre el ISPJAE y esta, ya sea por el dinamismo que vive la universidad, así como que la base da datos de esta tiene un tamaño mucho mayor que la base de datos que soporta el software. No brinda la posibilidad de gestionar la información referente al Plan de Carga, no permite obtener el Listado de productos disponibles ni la confección de la Demanda de materiales para un periodo, de la misma forma que no está integrado al sistema ASSETS, además no contempla la integración entre la asignación de alimentos y el menú.

1.1.3.1 Sistemas de apoyo a la gestión de comensales en la UCI.

En la universidad se han obtenido algunas aplicaciones informáticas que contribuyen al proceso de gestión de comensales. Algunas de estas aún se encuentran en periodo de prueba por lo que no se ha puesto en práctica su total funcionamiento. Estas herramientas son:

Sistema de Reportes. Permite llevar un control de cuántos comensales han pasado por cada comedor, puerta o complejo, así como establecer pronósticos a través de los diferentes reportes que emite a través del *Sistema de Control de Acceso*.

***Sistema de control de acceso*.** Es una aplicación de escritorio que permite llevar a cabo un control de los comensales que acceden por cada puerta de los comedores, garantizando que ninguno reciba el servicio dos veces en el mismo evento del día. Controla además que cada usuario acceda al comedor por la puerta que le fue asignada a través del sistema de distribución.

***Sistema de Distribución*.** Este sistema trabaja con la información de la base de datos de toda la comunidad universitaria con el objetivo fundamental de llevar a cabo una distribución de los comensales por cada complejo, comedor y por puerta para así tener una ubicación equitativa de todos los usuarios por todos los complejos de comedores.

***Sistema de Reservación*.** La aplicación permite que se le cancele el servicio de comedor a los trabajadores que no se encuentran en la universidad y de esta manera contribuye a que se tenga un mejor control y que la planificación de los comensales se lleve a cabo de una manera más eficiente.

1.2 Análisis de la solución propuesta.

Actualmente una de las formas que revolucionan las empresas es la automatización de sus procesos, con el objetivo de aumentar los indicadores de eficiencia, y por ende las ganancias.

La propuesta de realizar una Aplicación Web para la Gestión Integral de los Servicios de Comedores está basada en el uso que se le da hoy en día a las nuevas tecnologías lo que ayudaría a mejorar los procesos de planificación y control de los recursos que se utilizan dentro del área de comedores, debido a que permite interactuar con las bases de datos y distribuir la información. Su uso es más fácil, pues no necesitan ser instaladas, solamente se requiere de un servidor Web al cual se accederá desde cualquier parte, a cualquier hora y por múltiples usuarios al mismo tiempo, además no se precisa de conocimientos de computación para su uso. Contribuiría al ahorro de tiempo y de raciones en gran medida, ya que será un paso de avance en la eficiencia de la gestión de comensales y de la cantidad de alimentos a utilizar, y significaría un gran ahorro económico para la Universidad. Los cálculos que se llevan a cabo se realizarán de forma automática y en un menor tiempo, además se permitirá emitir reportes de los diferentes procesos. Con una buena gestión de comensales el reajuste del menú será de una manera más eficiente, ya que teniendo un número de comensales lo más acertado posible, se llevará a cabo una mejor planificación del menú y se evitará que este tenga que ser reajustado varias veces.

1.3 Proceso de desarrollo de software.

El **proceso de desarrollo de software** “es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño

implementado en código, el código es probado, documentado y certificado para su uso operativo. Concretamente define quién está haciendo qué, cuándo y cómo alcanzar un cierto objetivo" (1). El proceso de desarrollo de software requiere un conjunto de conceptos, una metodología y un lenguaje propio. Este proceso es también conocido como ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición.

1.3.1 Proceso Unificado de Desarrollo (*Rational Unified Process, RUP*).

Este es uno de los procesos más generales que existe, esta enfocado a cualquier tipo de proyecto así no sea de software, se basa en la documentación generada en cada una de sus fases.

El Proceso Unificado de Desarrollo Software (RUP) se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y ser iterativo e incremental, es un marco de desarrollo compuesto de cuatro fases:

- Inicio.
- Elaboración.
- Construcción.
- Transición.

Cada una de estas fases está dividida en una serie de iteraciones que ofrecen como resultado un incremento del producto desarrollado, que añade o mejora las funcionalidades del sistema en desarrollo. En estas iteraciones se realizan las actividades definidas en el ciclo de vida clásico. Aunque todas ellas suelen incluir trabajo en la mayoría de estas actividades (Ver anexo 3), el grado de esfuerzo dentro de cada una varía a lo largo del proyecto y una actividad se puede realizar completamente en más de una iteración en dependencia de la complejidad del proyecto. RUP tiene definido cuatro elementos: los roles, que responden al ¿Quién?, los artefactos, que responden al ¿Cómo?, las actividades que responden al ¿Qué? y los flujos de trabajo, que responden al ¿Cuándo?

El **rol** define las responsabilidades de los individuos del equipo de trabajo, cada individuo puede desempeñar varios roles y un rol puede ser representado por varios individuos. La **actividad** es una unidad de trabajo que un individuo que desempeñe un rol puede realizar. Las actividades tienen un objetivo concreto, crear o actualizar algún producto. El **artefacto** es un fragmento de información que

es producido y usado durante el proceso de desarrollo de software. Los artefactos son los resultados tangibles del proyecto que se van creando y usando hasta obtener el producto final. El **flujo de trabajo** es una relación de actividades que producen resultados observables dado por una secuencia de actividades realizadas por los diferentes roles. RUP tiene definidos los siguientes flujos de trabajo:

- Modelado del negocio
- Requerimientos.
- Análisis y Diseño.
- Implementación.
- Pruebas.
- Despliegue.
- Gestión del proyecto.
- Configuración y control de cambios.
- Entorno.

RUP utiliza el Lenguaje Unificado de Modelado (*Unified Modeling Language, UML*) como leguaje de notación.

1.3.2 *Extreme Programming* (Metodología XP).

Extreme Programming es una de las metodologías de desarrollo de software más exitosas, en la actualidad utilizada para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto (2). La metodología se basa en:

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir.

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

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Analizando lo referente a estas tecnologías, determinando el alcance de esta investigación y consultando la opinión de los especialistas y el personal involucrado se decidió utilizar RUP, debido a que el proyecto es complejo y de larga duración. Además durante este proceso se generan una serie de documentos necesarios tanto para el cliente como para el desarrollador.

1.4. Roles y Artefactos desarrollados en el trabajo.

Durante el desarrollo del proyecto en el cual está enmarcada esta investigación, se han tenido en cuenta una serie de roles y artefactos que están dentro de los flujos de trabajo definidos por RUP y que se implementarán en el mismo:

1- Modelo de Negocio.

Roles:

- Analista de los procesos del negocio.
- Diseñador del negocio.

Artefactos:

- Actor del negocio.
- Trabajadores del negocio.
- Casos de uso del negocio.
- Descripción de los casos de uso del negocio.
- Diagrama de casos de uso del negocio.
- Diagrama de Actividades.
- Modelo de objetos del negocio.
- Reglas del negocio.

2- Requerimientos

Roles:

- Analista del sistema.
- Especificador de Requerimientos.

Artefactos:

- Actores del sistema.

- Casos de uso del sistema.
- Diagrama de casos de uso del sistema.
- Descripciones textuales de los casos de uso del sistema.
- Especificación de requerimientos (Requisitos funcionales y no funcionales).
- Vista de casos de uso.

3- Análisis y Diseño.

Roles:

- Diseñador de Interfaz de Usuario.
- Diseñador.

Artefactos:

- Mapa de navegación.
- Prototipos de interfaz de usuario.
- Paquetes de diseño.
- Clases de diseño.
- Realización de casos de uso del sistema.
- Diagrama de despliegue.
- Vista lógica.

1.5 UML como lenguaje de Modelado.

El Lenguaje de Modelado Unificado (UML) es un lenguaje estándar, consolidado, Orientado a Objetos, para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software (3). Permite una comunicación fluida entre los diversos actores acerca del modelo y además es fácil de aprender. Los elementos de UML se muestran mediante diagramas que presentan múltiples vistas del sistema, ese conjunto de vistas son conocidas como modelos. Los diagramas de UML representan un aspecto del sistema. UML se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones. Este lenguaje recomienda utilizar los procesos que otras metodologías tienen definidos. Podemos decir que una parte de UML define una abstracción con significado de un lenguaje para expresar otros modelos, es decir, otras abstracciones de un sistema o un conjunto de unidades conectadas que se organizan para conseguir un propósito. Esto en principio puede parecer complicado

pero no es la realidad si logramos comprender que uno de los objetivos de este lenguaje es definir modelos, no solo establecerlos.

Para el desarrollo de este proyecto se decidió utilizar UML como lenguaje de modelado, por todas las ventajas que posee y principalmente por ser utilizado por la metodología seleccionada.

1.6 Herramientas Case.

Las Herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar el diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras.

Algunos de los componentes de las herramientas CASE permiten:

- Confeccionar la definición de requerimientos de los usuarios.
- Mejorar el diseño de los sistemas.
- Mejorar la eficiencia en la programación (por su generación automática de códigos).
- Otorgar a la administración un mejor soporte en la documentación.

En el mundo existen diferentes herramientas CASE para el modelado de artefactos, entre ellas podemos ver: *Rational Rose, Visual Paradigm, Umbrello*.

Rational Rose: es una herramienta con plataforma independiente. Una de las grandes ventajas del Rational es que utiliza la notación estándar en la arquitectura de Software (UML), la cual permite visualizar el sistema completo utilizando un lenguaje común. Otra ventaja de Rose es que se puede modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. Además proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente. Esta herramienta es líder en el mercado para el análisis, Modelamiento, diseño y construcción orientado a objetos (3).

***Umbrello*:** es una herramienta libre que ayuda en el proceso de desarrollo de software para editar y crear diagramas UML. Aunque funciona en otros entornos de escritorio, está diseñado para KDE principalmente. Permite la generación de código de varios lenguajes y es fácil de usar favoreciendo así la elaboración de un producto de alta calidad.

*"**Visual Paradigm*** para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML."(4)

Principales características de *Visual Paradigm*:

- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

Para la realización de este trabajo se decide utilizar *Visual Paradigm* Versión 6.0 pues además de todas las características anteriores presenta una licencia comercial, la cual es pagada por la Universidad. Es fácil de instalar y actualizar. Soporta ingeniería inversa y aplicaciones Web. Además presenta un ambiente gráfico agradable al usuario.

1.7 Herramienta para la gestión de requisitos.

Los requisitos de software son las necesidades de los clientes para resolver un determinado problema por lo cual la gestión de requisitos, es una de las partes más importantes del proceso de desarrollo de software. El proceso de gestión de requisitos, no se basa únicamente en la deducción de los mismos, si no que implica más tareas, entre las cuales podemos destacar:

- Recogida de requisitos
- Formalización de requisitos
- Revisión
- Gestión

Las herramientas que dan soporte a estas tareas son las conocidas como Herramientas de Gestión de Requisitos (5). Entre estas herramientas podemos encontrar:

Open Source Requirement Management Tool (OSRMT), es una herramienta de software libre, llevada a cabo por un único desarrollador y que no ofrece ningún soporte empresarial. Esta herramienta permite la descripción avanzada de diversos tipos de requisitos y garantiza la trazabilidad entre todos los documentos relacionados con la ingeniería de requisitos (funcionalidades, requisitos, casos de uso, casos de prueba).

Características y funcionalidades básicas de la herramienta

La herramienta integra diversos módulos como son:

- Administración y Configuración
- Gestión de documentos de ingeniería de requisitos (funcionalidades, requisitos, casos de uso, casos de prueba).
- Trazabilidad entre documentos de trabajo
- Informes y estadísticas

Algunas de las funcionalidades que ofrece la herramienta son:

- Gestión de requisitos, diferenciando entre Requisitos:
  - Funcionalidades
  - Requisitos técnicos
  - Casos de prueba
- Trazabilidad entre todos los documentos de trabajo:
  - Requisito-Requisito (control de versiones)
  - Requisito-Requisito (dependencia entre requisitos)

- Requisito-Funcionalidad
- Requisito-Caso de Prueba
- Visualización de la matriz de trazabilidad
- Árbol de trazabilidad para facilitar las auditorias
- Gráfico de dependencias entre documentos de trabajo para poder determinar el impacto de un cambio

➢ Personalización y configuración:
- Definición de los atributos de una funcionalidad
- Definición de los atributos de un requisito
- Definición de los atributos de un caso de prueba
- Valores predefinidos para cada usuario (prioridades por defecto, estado por defecto, etc)
- Personalización de vistas

➢ Representación jerárquica de los documentos de trabajo
➢ Descripción de un requisito mediante la secuencia de pasos de su caso de uso
➢ Herramientas de migración para los diversos cambios de versiones
➢ Múltiples idiomas (importación y exportación para dar soporte a diversos idiomas)

Esta herramienta nos permite obtener una visualización de los requisitos en forma jerárquica, la cual es intuitiva y fácil de manejar. Su licencia es GPL, es multiplataforma. Como herramienta *open source* de gestión de requisitos no tiene mucha competencia en cuanto a la funcionalidad ofrecida. Tiene una buena documentación pese a tratarse de una herramienta muy reciente. A pesar de ser llevada a cabo por un único desarrollador, el ritmo de mejoras y nuevas versiones es constante. Existen muchas opciones para configurar y personalizar la herramienta a las necesidades concretas de una organización, además lleva incorporado un sistema de gestión de la configuración que permite definir líneas base. Existe un gran soporte para mantener la trazabilidad entre los documentos, así como mecanismos que facilitan la importación y exportación de la información en XML. (5)

Requisite Management (REM), esta es una herramienta experimental gratuita de Gestión de Requisitos diseñada para soportar la fase de Ingeniería de Requisitos de un proyecto de desarrollo software de acuerdo con la metodología definida en la Tesis Doctoral "Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información", realizada por el profesor Amador Durán del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

Características y funcionalidad básica

Los diversos módulos integrados en la herramienta son:

- Gestión de documentos de ingeniería de requisitos
- Trazabilidad entre documentos de trabajo
- Informes y estadísticas
- Generación de documento HTML

Algunas de las funcionalidades ofrecidas por la herramienta son:

- Gestión de requisitos, diferenciando entre:
  - Requisitos
  - Funcionalidades
  - Requisitos técnicos
  - Casos de prueba
- Trazabilidad entre todos los documentos de trabajo:
  - Requisito-Requisito (control de versiones)
  - Requisito-Requisito (dependencia entre requisitos)
  - Requisito-Funcionalidad
  - Requisito-Caso de Prueba
  - Visualización de la matriz de trazabilidad
  - Árbol de trazabilidad para facilitar las auditorias
  - Gráfico de dependencias entre documentos de trabajo para poder determinar el impacto de un cambio
- Representación jerárquica de los documentos de trabajo
- Definición de casos de prueba mediante pruebas para cada uno de los pasos del caso de uso
- Descripción de un requisito mediante la secuencia de pasos de su caso de uso
- A la vez que se introduce la información va generando un documento HTML. (5)

Esta herramienta tiene una interfaz muy intuitiva, por lo que la curva de aprendizaje es mínima debido a lo cual se recomienda como una muy buena herramienta para comenzar con el hábito de realizar la gestión de requisitos, ya que es muy sencilla, permite incluir la trazabilidad entre los requisitos y soporta dependencias entre estos.

Enterprise Architect (EA), es una herramienta flexible, completa y potente de modelado UML bajo la plataforma Windows. Provee lo más nuevo en desarrollo de sistemas, administración de proyectos y análisis de negocio. Esta herramienta abarca integralmente el ciclo de vida, cubriendo el desarrollo de

software desde el levantamiento de los requisitos, a través de las etapas de análisis, modelos de diseño, testeo y finalmente el mantenimiento y re-uso. Es una herramienta utilizada para el desarrollo de varios tipos de software para un amplio rango de industrias, incluyendo: bancos, ingeniería, finanzas, medicina, investigación y muchas más. *Enterprise Architect* combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación. Con un gran conjunto de características y un valor sin igual para el dinero, *EA* puede equipar a su equipo entero, incluyendo analistas, evaluadores, administradores de proyectos, personal del control de calidad, equipo de desarrollo y más, por una fracción del costo de algunos productos competitivos. (6)

EA es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. El manual de usuario está disponible en línea.

Se escoge como herramienta para la gestión de requisitos *OSRMT* debido a que es *open source*, multiplataforma, características que la diferencian de EA y de *REM*, además de que esta última no es una herramienta comercial (No tiene soporte), tiene algunos errores, es lenta cuando se trabaja con muchos objetos, no permite controlar la numeración de los requisitos y no permite dar formato al texto y en el caso de *EA* a pesar de tener múltiples ventajas como se mencionaba anteriormente, es mucho más complejo el trabajo con la misma debido a que no es muy fácil su aprendizaje.

1.8 Herramientas para el diseño de los prototipos de interfaz no funcionales.

Realizar una correcta selección de la herramienta a utilizar para el diseño web, brinda gran ayuda para los de desarrolladores, ya que se les facilita y optimiza el trabajo para el desarrollo de un sitio web con calidad. Entre estas herramientas tenemos: *FrontPage*, *Macromedia Dreamweaver* con todas sus versiones, entre otras.

Macromedia Dreamweaver 8 incluye las versiones más recientes de *Dreamweaver, Flash, Fireworks* y *FreeHand* (7). Permite la creación de aplicaciones avanzadas, pues presenta la capacidad de programar bajo lenguajes como: ASP, CSS, PHP, JSP y XML; lo que la convierte en una herramienta altamente compatible con las principales tecnologías de servidor, esto y el amplio desarrollo sobre sitios web, hace que mantenga en la sombra a herramientas como *FrontPage*. Soporta la mayoría de las tecnologías servidor y HTML

- Realiza el trabajo con capas
- Presenta una página de inicio para lograr un mejor acceso a documentos, recursos y archivos
- Brinda la posibilidad de crear botones *Flash*, formularios, *JavaScripts* y más, es de gran ayuda. Además de poder insertar elementos web, encontramos una gran precisión en la importación de información de *Word* y *Excel*, con las funciones de copiar y pegar. (8)
- *JavaScript* para la creación de efectos e interactividades
- Inserción de archivos multimedia

Es ampliable y abierto, cuenta con las tecnologías y productos que el usuario desea para tener una mayor libertad a la hora de elegir las tecnologías más convenientes. Permite crear el trabajo en corto tiempo al agilizar el flujo de trabajo. Esta herramienta se puede actualizar tanto con componentes fabricados por Macromedia como con componentes desarrollados por otras compañías, lo cual permite la realización de acciones más avanzadas.

Frontpage es una herramienta para la edición de páginas web de *Microsoft*. Creado hace ya muchos años, ha tenido multitud de versiones que han ido mejorando su funcionamiento. Está orientado a personas inexpertas y sin conocimientos de HTML. Sus capacidades son semejantes a las de otros editores, como el crear mapas de imágenes, gestionar la arborescencia de las páginas del sitio, entre otras. Lamentablemente, al ser un producto *Microsoft*, está orientado a construir páginas optimizadas para *Internet Explorer*. Por esta misma razón, al insertar algún elemento activo en una página web, como es el caso de los controles *ActiveX*, o los *scripts* de cliente, sólo suele funcionar en *Internet Explorer*. Conseguir páginas que se vean bien en *Netscape Navigator* puede ser complicado con este programa, lo que es un serio inconveniente.

Por las características de las herramientas y basándonos en la experiencia que se ha tenido con el trabajo de las mismas, se decide utilizar como herramienta para el diseño de los prototipos no funcionales y los de interfaz de usuario *Macromedia Dreamweaver 8* por todas las facilidades y características para crear documentos web adaptables a las necesidades. Además es el editor más utilizado mundialmente, por sus prestaciones, su sencillez y facilidades de trabajo.

1.9 Patrones

Los patrones son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software, pretenden formalizar un vocabulario común, así como evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.

1.9.1. Patrones de Casos de Uso.

Los patrones de casos de uso se utilizan como técnicas o herramientas, resultantes de la experiencia de desarrolladores, que nos permiten resolver problemas que se presentan en la modelación de sistemas, obteniendo modelos de mayor calidad de forma más rápida.
Entre los patrones de casos de uso podemos encontrar:

- **Patrón CRUD**(*Creating, Reading, Updating and Deleting*): propone identificar un CU , llamado "Información CRUD" o "Administrar Información", que modela todas las operaciones que se pueden realizar sobre una parte de información de cierto tipo (o sea en una misma entidad), tal como crearla, leerla, actualizarla y eliminarla.
- **Múltiples actores**: Roles comunes (*Multiple Actors: Common Role*): consiste en que dos actores juegan el mismo papel hacia el caso de uso. Este rol es representado por otro actor, heredado por los actores que comparten este rol.

1.9.2 Patrones de arquitectura.

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (9)
Para el desarrollo de la aplicación se utilizará el *framework Symfony* el cual está basado en el patrón conocido como arquitectura Modelo-Vista-Controlador (MVC). Este patrón está formado por 3 niveles (Ver anexo 4):

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (10)

1.9.3 Patrones de diseño.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular, identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades.

La arquitectura MVC implementa internamente los patrones de asignación de responsabilidades GRASP.

”Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.” (11)

Los patrones GRASP son nueve, de ellos cinco principales y cuatro adicionales, de estos patrones se usan:

Principales:

- Bajo acoplamiento, se basa en la idea de tener las clases lo menos ligadas entre sí posible. De forma tal que en caso de producirse una modificación en alguna de ellas, la repercusión en el resto de clases sea mínima, potenciando así la reutilización, y disminuyendo la dependencia entre estas.

- Alta Cohesión, este patrón permite asignar una responsabilidad para mantener alta la cohesión, es decir, que cada elemento debe realizar una labor única en el sistema, la cual no debe ser desempeñada por ningún otro elemento del mismo y debe estar relacionada con la clase que la realiza.

- Experto, es el patrón más usado pues el asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la “intuición” de que los objetos hacen cosas relacionadas con la información que poseen.

- Creador, propone asignar la responsabilidad de crear nuevos objetos de una clase a otra si esta última: contiene la infromación necesaria para realizar la creación del objeto, o si usa

directamente las instancias creadas del objeto o si almacena o maneja varias instancias de la clase.

- Controlador, este patrón sirve de intermediario entre una interfaz y el algoritmo que la implementa, siendo el que recibe los datos del usuario y los envía a las distintas clases según el método llamado.

Adicionales:

- Fabricación pura, este patrón se basa en la problemática de ¿cómo proceder cuando las soluciones encontradas comprometen la cohesión y el acoplamiento? es decir, ¿Qué objetos deberían tener la responsabilidad cuando no se quiere violar los objetivos de los patrones de alta cohesión y bajo acoplamiento, u otros, o cuando las soluciones que ofrece el experto, por ejemplo, no son adecuadas? Propone como solución asignar un grupo de responsabilidades altamente cohesivas a una clase artificial o de conveniencia, sin que esta represente ningún concepto del dominio del problema. Tal clase es una *fabricación* de la imaginación e idealmente las responsabilidades asignadas a esta fabricación soportan alta cohesión y bajo acoplamiento, de manera que el diseño de la fabricación es muy limpio o *puro*.

Del grupo de patrones GoF se clasifican en: creacionales, estrucutrales y de comportamiento se hará uso de:

Creacionales:

Singleton, el cual provee una única instancia global gracias a que:

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de clase.
- Declara el constructor de clase como privado para que no sea instanciable directamente.

Estructurales:

Decorator, añade objetos individuales de forma dinámica y transparente siendo aplicable cuando:

- La responsabilidades de un objeto pueden ser retiradas.
- La extensión mediante la herencia no es viable.

- Hay una necesidad de extender la funcionalidad de una clase, pero sin poder hacerlo mediante la herencia.
- Hay la necesidad de extender dinámicamente la funcionalidad de un objeto.

Facade, sirve para proveer de una interfaz única y sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas, además puede:

- Hacer una biblioteca de software más fácil de usar y entender, pues implementa métodos convenientes para tareas comunes.
- Hacer el código que usa la librería más legible.
- Reducir la dependencia de código externo en los trabajos internos de una librería.

Composite, permite la construcción de objetos complejos a partir de otros más simples y similares entre sí, debido a la composición recursiva y a la estructura en forma de árbol, lo que hace más simple el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan de la misma manera.

Comportamiento:

Chain of Responsibility, permite establecer una cadena de objetos receptores a través de los cuales se pasa una petición formulada por un objeto emisor. Cualquiera de los objetos receptores puede responder a la petición en función de un criterio establecido.

1.10 *Framework* a utilizar: *Symfony*

Los *frameworks* están diseñados con el propósito de facilitar el desarrollo de software. Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Proporciona estructura al código fuente, forzando al programador a crear código más legible y más fácil de mantener y a su vez facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. Está desarrollado completamente con PHP 5 y se basa en el patrón de diseño web MVC.

"*Symfony* es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir

el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Características de *Symfony*

Se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas *Windows* y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.
- Automatización de características de proyectos web.(12)

1.11 Consideraciones del capítulo.

En este capítulo se realiza un análisis de las diferentes aplicaciones para planificación y control de servicios de comedores, utilizadas hoy en dia en el mundo y en Cuba, asi como la necesidad de utilizar en la UCI un sistema de información que permita contribuir al mejor funcionamiento económico de la Universidad. Además se describien las características fundamentales de las metodologías más utilizadas en el desarrollo de aplicaciones, se analizan sus características, ventajas y desventajas y se llegó a la conclusión de que la propuesta consiste en:

- Desarrollar una aplicación Web que facilite la gestión de la información en la planificación y control de comensales en la UCI.
- Utilizar RUP como metodología de desarrollo para guiar el proceso de modelación, análisis y diseño y UML para describir todo el proceso.

- *Visual Paradigm* como herramienta CASE.
- *Macromedia Dreamweaver 8* como herramienta de diseño web.
- Symfony como *framework*.

Capitulo 2: Características del sistema.

En este capítulo se hace una definición del objeto de estudio del problema, se plantean los objetivos estratégicos de la organización y los procesos del negocio que los soportan. Se describen los procesos que serán objeto de automatización, así como el funcionamiento general de la propuesta del sistema. Se define el Modelo del Negocio. Se especifican de los requisitos y casos de uso del sistema.

2.1 Modelo del negocio actual.

Para poder entender con claridad todo el negocio es necesario modelar los procesos que se llevan a cabo en el mismo. Los requerimientos más importantes que puede tener el sistema se obtienen luego de haber elaborado el modelo del negocio. Con este modelo se pueden organizar y presentar con más detalle los procesos que se desarrollan en la gestión de comensales en la UCI.

2.1.1 Reglas del Negocio.

Las reglas del negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio. A continuación se muestran las identificadas en el presente negocio:

- No se pueden planificar los comensales, si no existe un menú previamente elaborado.
- Cada vez que se emite un menú hay que planificar los comensales para el mismo.
- Para un período determinado puede existir solo un menú.
- No se puede generar una orden de producción si no está confirmada la planificación del menú.
- Toda solicitud de materiales tiene que tener asociada una orden de producción.
- Todo vale de salida tiene que tener asociada una solicitud de materiales.
- No se pueden extraer los productos del almacén, sino existe la solicitud de materiales asociada al servicio del día en un evento determinado.
- No se puede obtener la demanda de materiales si no esta planificado el menú para un período.
- Una orden de producción va asociada a una Unidad Productiva.
- Una solicitud de materiales y una orden de producción van asociadas a un plato que este en el listado.

2.1.2 Actores y Trabajadores del negocio.

Actores del Negocio	Descripción
Cliente	Son las personas dentro de la universidad que solicitan un servicio al Director General de Alimentos, estas pueden ser: directivos (FEU, UJC, trabajadores).
Reloj	Representa un ente que inicia las actividades que se desarrollan con cierta periodicidad dentro de la DGA de la UCI.

Tabla 1 Actores del negocio

Trabajadores	Descripción
Planificador de comensales	Es el encargado de realizar la planificación de comensales para un menú determinado.
Jefe de Producción	Es el encargado recibir la orden de producción, la Solicitud de materiales y el vale de salida, para luego entregar estos documentos a las personas que los necesiten.
Planificador del menú	Es el responsable de confeccionar los documentos “Orden de producción” y “Solicitud de materiales”.
Jefe de almacén	Es el encargado de recibir la solicitud de materiales para elaborar el vale de salida.
Jefe de compras	Es la persona responsable de que se realice el documento Demanda de materiales.

Tabla 2 Trabajadores del Negocio

2.1.3 Diagrama de Casos de Uso del Negocio.

El diagrama de casos de uso del negocio permite representar gráficamente a los procesos del negocio y la interacción de estos con los actores del negocio

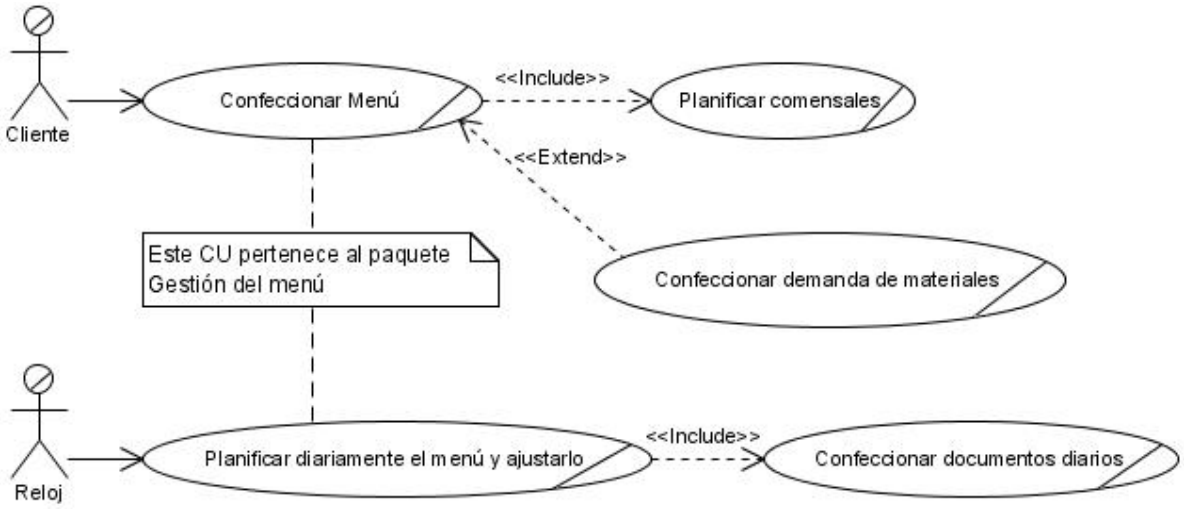

Figura 1 Diagrama de Casos de Uso del Negocio.

2.1.4 Descripciones de los casos de uso del negocio.

Para poder modelar los procesos que se llevan a cabo en el negocio actual es necesario centrar la atención en los procesos de Planificar comensales, confeccionar Orden de producción y Solicitud de materiales, así como confeccionar la Demanda de materiales. El primer paso para modelar estos procesos es identificar las actividades que se llevan a cabo en los mismos.

Caso de Uso:	Planificar comensales
Actores:	Cliente
Trabajadores:	Planificador de comensales, planificador del menú.
Resumen:	Este proceso es incluido del proceso Confeccionar Menú del paquete Gestión del menú, ya que existe una retroalimentación entre ambos, o sea, cada vez que se emite un menú se planifican los comensales. Cuando el cliente solicita un servicio de alimentos, el planificador del

	menú confecciona y emite un menú, luego el planificador de comensales determina un número global de comensales, analiza diferentes criterios de pronóstico y realiza la planificación. Luego archiva y envía al Planificador de Menú la planificación de comensales que se hizo para el menú emitido. Ahí termina el proceso.
Precondiciones:	Que se halla realizado el proceso de Confeccionar Menú.

Tabla 3 Descripción Textual del Caso de Uso “Planificar comensales”.

Caso de Uso:	Confeccionar documentos diarios
Actores:	Reloj
Trabajadores:	Planificador de menú, Jefe de producción , Jefe de almacén
Resumen:	Este proceso se realiza diario, se asocia a cada evento del día. Es un Caso de Uso (CU) incluido del CU Planificar diariamente el menú y ajustarlo. Inicialmente el reloj indica que inicie el proceso, luego el planificador del menú planifica y reajusta el menú diario, posteriormente analiza este menú y el listado de platos para elaborar la orden de producción y la solicitud de materiales, luego envía estos documentos al jefe de producción , el cual envía la solicitud de materiales al jefe de almacén, este la recibe y emite el vale de salida , recibiéndolo el jefe de producción , archivándolo en conjunto con la orden de producción y la solicitud de materiales. Termina el proceso.
Precondiciones:	Que se halla realizado el proceso “Planificar diariamente el menú y reajustarlo”.

Tabla 4 Descripción Textual del Caso de Uso “Confeccionar documentos diarios”.

Caso de Uso:	Confeccionar demanda de materiales.
Actores:	Cliente
Trabajadores:	Jefe de Compras, Planificador del menú.
Resumen:	Este Caso de Uso (CU) es extendido del CU Confeccionar Menú del paquete Gestión del Menú, ya que se pude realizar el proceso cada vez que sea emitido un menú para más de un día.

	Inicialmente el cliente solicita un servicio de alimentos, luego el planificador del menú confecciona un menú. Posteriormente decide si se debe o no confeccionar una demanda de materiales. En caso de que decida crearla, se lo indica al jefe de compras, el cual verifica la existencia del menú, si existe lo analiza y a su vez analiza el listado de platos. Elabora el documento "Demanda de materiales "y lo archiva. En caso de no existir el menú termina el proceso.
Precondiciones:	Que exista el menú

Tabla 5 Descripción Textual del Caso de Uso "Confeccionar demanda de materiales".

Para ver el flujo normal de los eventos de cada proceso, remitirse al expediente de proyecto.

2.1.5 Diagramas de actividades.

Los diagramas de actividades muestran el flujo de trabajo de un CU, desde el punto de inicio hasta el punto final, detallando las distintas rutas que pueden irse desencadenando.

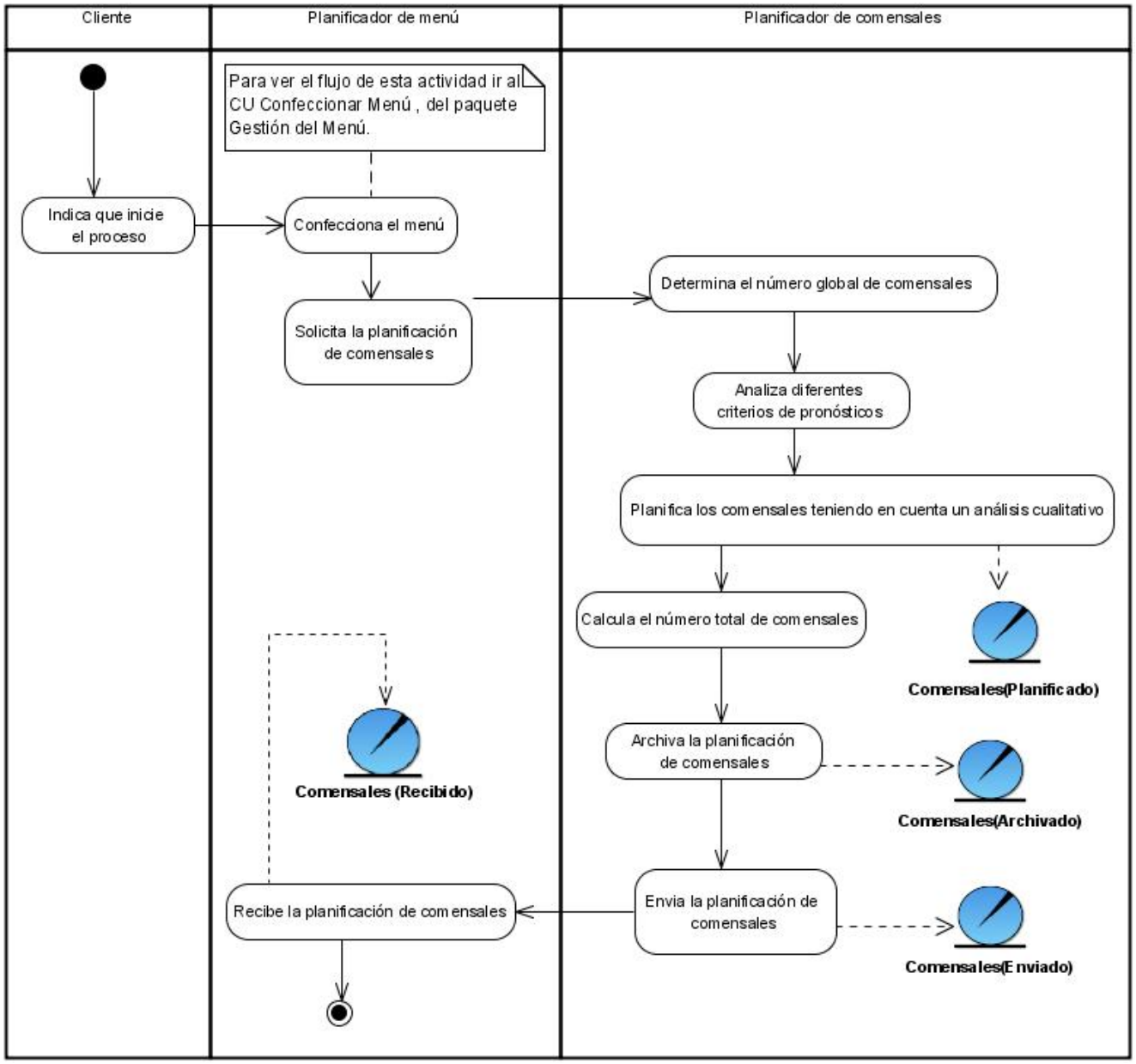

Figura 2 Diagrama de actividades del CU del negocio "Planificar comensales".

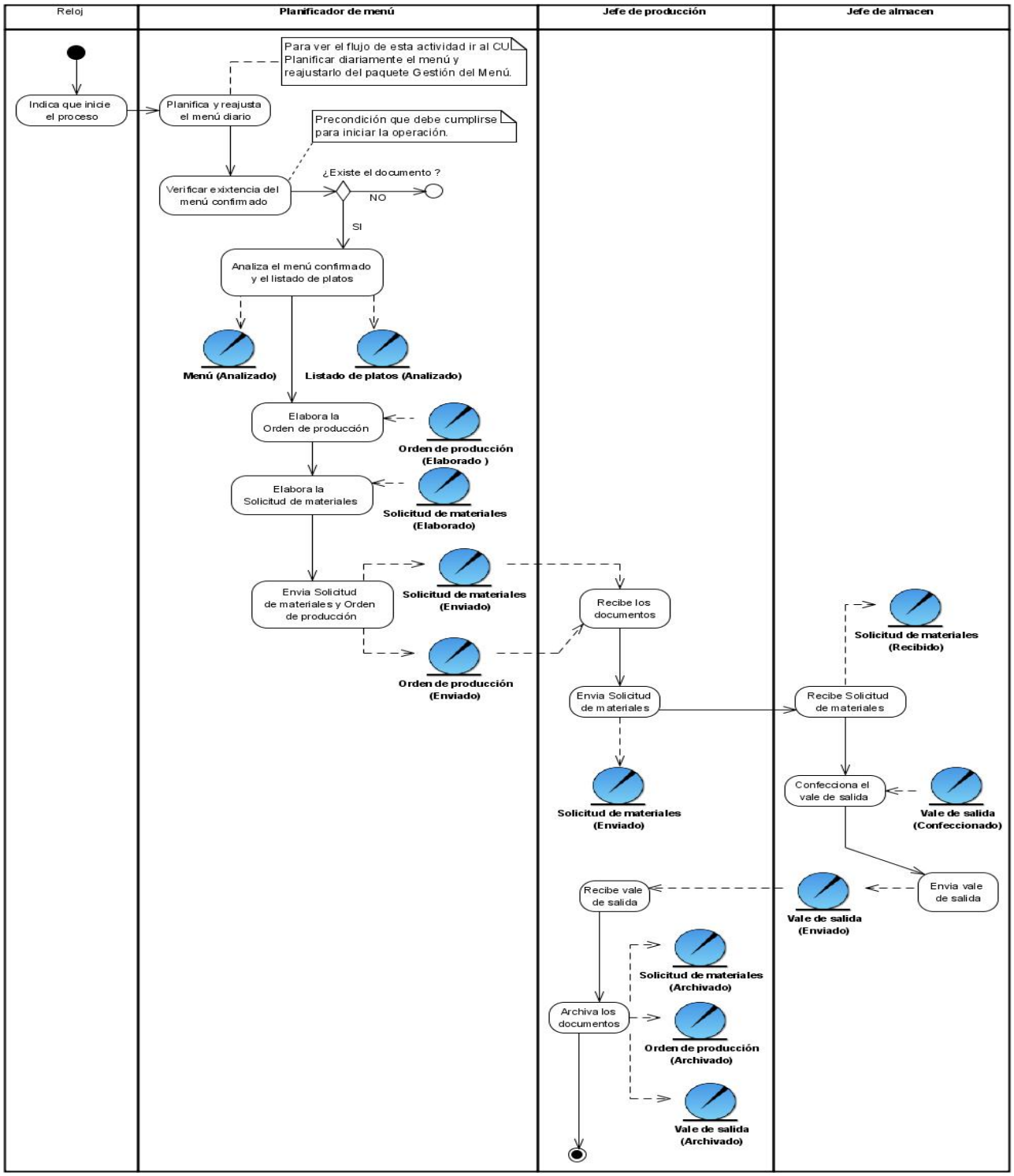

Figura 3 Diagrama de actividades del CU del negocio " Confeccionar documentos diarios".

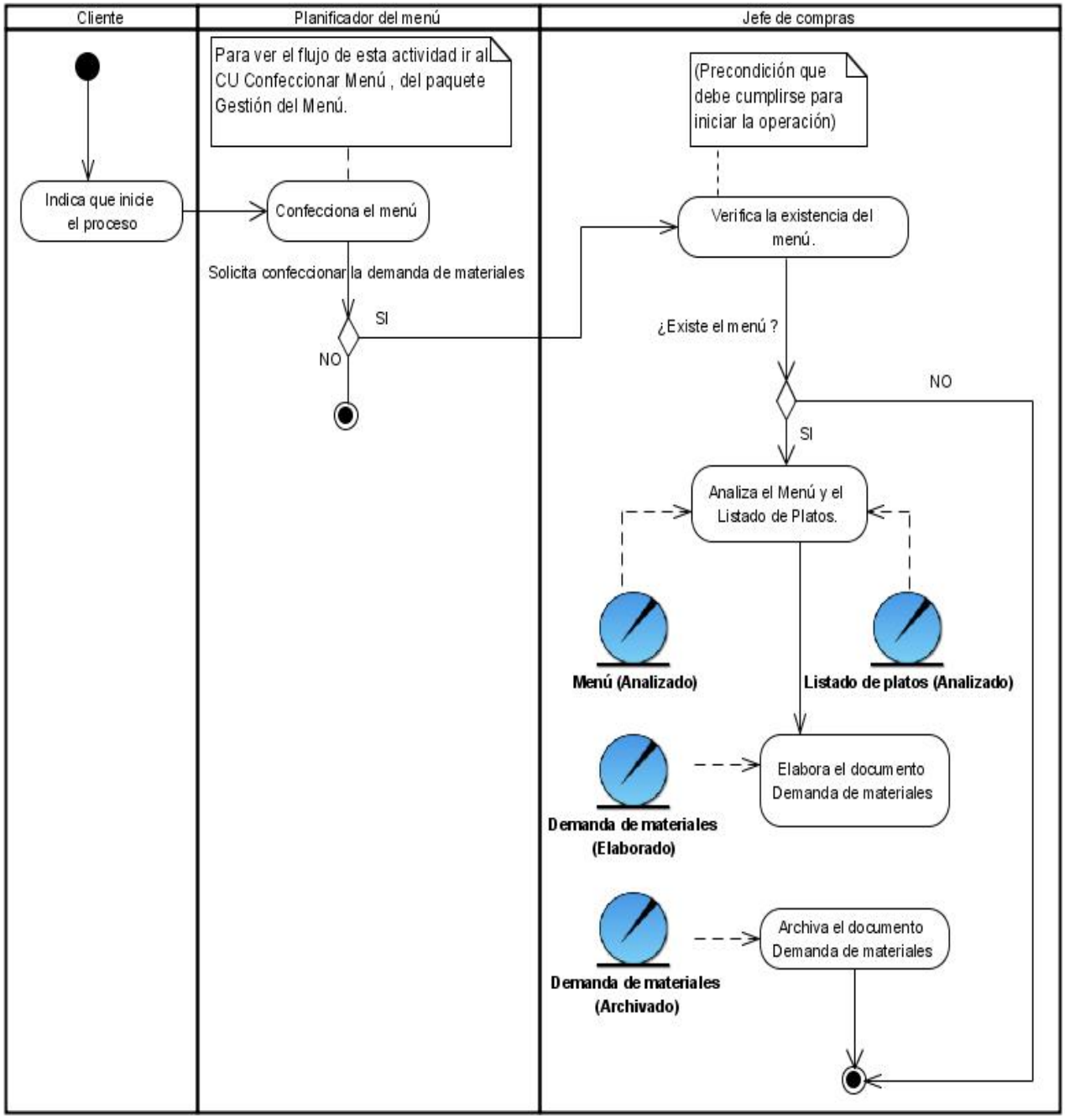

Figura 4 Diagrama de actividades de CU del negocio "Confeccionar demanda de materiales".

2.1.6 Modelo de Objetos del Negocio.

Este modelo representa los trabajadores del negocio y su interacción con las entidades del mismo en el flujo de trabajo “Modelado del Negocio”.

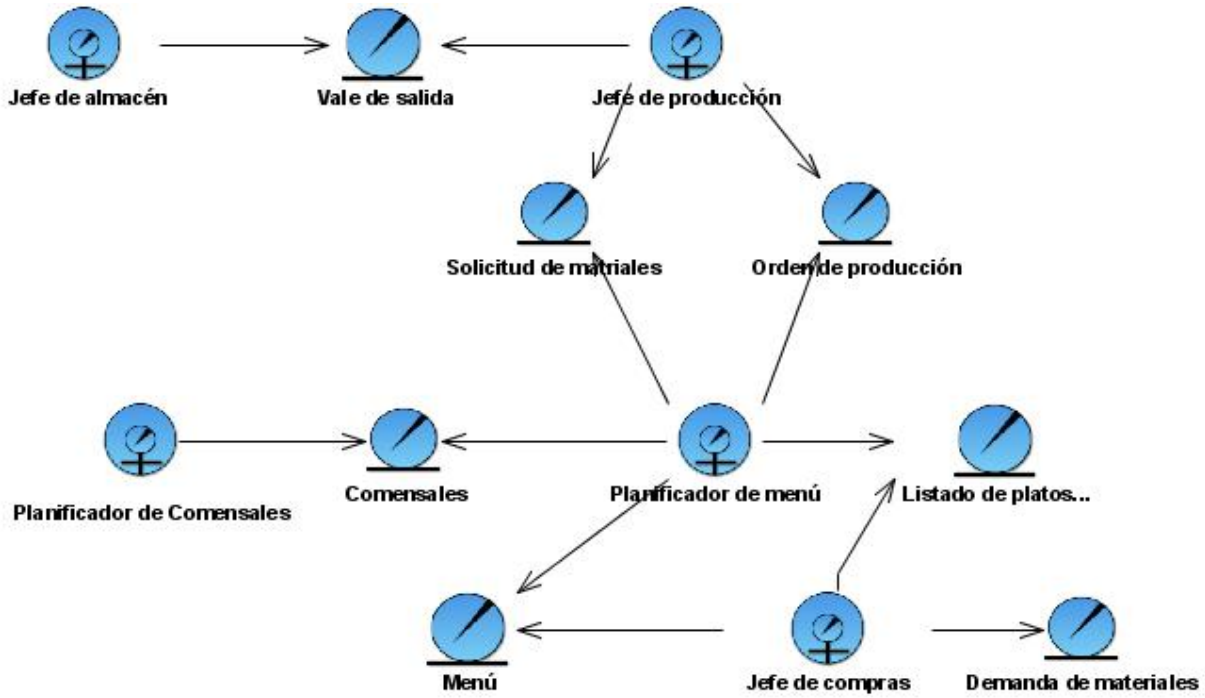

Figura 5 Modelo de Objeto del Negocio.

Luego de haber modelado los procesos que se llevan a cabo en el negocio actual, se procede a identificar los procesos y funcionalidades fundamentales que se han identificado para el sistema.

2.2 Modelado del Sistema.

2.2.1 Requerimientos Funcionales.

Los Requerimientos funcionales definen el comportamiento interno que tendrá el software que se desea construir. Muestran como los casos de uso del sistema serán llevados a la práctica.

RF 1. Verificar comensales.
RF 2. Mostrar comensales planificados y reales.

2.1. Imprimir reporte de comensales.

RF 3. Obtener Demanda de alimentos para un período.

3.1. Imprimir demanda de alimentos para un período.

RF 4. Obtener la Demanda Real de alimentos.

4.1. Imprimir demanda real de alimentos de un periodo.

RF 5. Obtener Orden de producción.

5.1. Imprimir Orden de producción.

RF 6. Obtener Solicitud de Materiales.

6.1. Imprimir Solicitud de Materiales.

RF 7. Buscar vale de salida.

RF 8. Buscar Balance Nutricional planificado.

8.1 Imprimir Balance Nutricional.

RF 9. Buscar Balance Nutricional real.

9.1 Imprimir Balance Nutricional real.

RF 10. Buscar Normas Nutricionales.

10.1 Imprimir Normas Nutricionales.

2.2.2 Requerimientos no Funcionales.

Los requerimientos no funcionales especifican las propiedades que debe de tener el sistema.

- **Requerimientos de Software**

Para la instalación del sistema se debe disponer del Sistema Operativo *Windows* 98 o superior, o cualquier distribución de Linux. Con el objetivo de lograr que las interfaces Web puedan visualizarse, las computadoras clientes usarán uno de los siguientes navegadores: *Internet Explorer* 5.5 o superior, *Netscape*, *Mozilla* 1.7 o superior o *FireFox* 0.9.3 o superior.
Se debe instalar un servidor Web Apache 1.3 o superior.

- **Requerimientos de Hardware**

Para el correcto funcionamiento de la aplicación es necesario que haya conectividad.
La máquina que se usará como servidor Web debe tener alta disponibilidad además de un rendimiento adecuado. Este servidor debe presentar las siguientes características: Procesador

Pentium IV superior a 2.0 GHz, 512 MB de memoria RAM (incluye la utilizada por el Sistema Operativo) y como mínimo 80.0 GB de capacidad en disco duro.
Las computadoras clientes deben contar con al menos una impresora para garantizar la impresión de los documentos necesarios sin ninguna dificultad y deben tener como mínimo un procesador Pentium IV superior a 1.0 GHz, 256 Mb de memoria RAM. Estas máquinas deben de estar conectadas en red con el servidor.

- **Requerimientos de apariencia o interfaz externa**

Las páginas no se cargarán con mucha información, poseerán las imágenes necesarias y poseerán colores claros, correspondiéndose a los estándares establecidos para lograr un buen diseño. Se contará en la página principal con la información necesaria para guiar al usuario y con una interfaz diferente para cada tipo de usuario que se registre.

- **Requerimientos de Usabilidad**

La aplicación será utilizada por el personal de la VR de logística de la UCI específicamente por los que laboren en la DGA, ATM y las diferentes unidades de producción y servicios.

Se impartirá una preparación a los usuarios con la explicación de cómo se realizará el trabajo con la aplicación y la misma contendrá un manual de usuario que será usado como ayuda para su desempeño.

- **Requerimientos de Soporte**

Una vez que la aplicación esté terminada se instalará en la UCI para realizar las pruebas necesarias y comprobar el correcto funcionamiento de la misma.
Cada cierto tiempo se realizará por los administradores del sistema un mantenimiento y en caso de ser necesario se realizarán otras versiones para mejorar la aplicación diseñada.

- **Requerimientos de Seguridad**

Para acceder a cualquiera de las páginas de la aplicación los usuarios deben ser sometidos a la autenticación, especificando usuario y contraseña de los mismos.

Cada usuario tendrá permisos de acceso a las páginas de la aplicación de acuerdo con la responsabilidad de los mismos, teniendo definido un solo rol para el manejo de la aplicación, únicamente en caso extremo se le asignará más de un rol a un usuario.

La contraseña de los usuarios debe de tener un mínimo de 7 caracteres y debe tener una fortaleza media.

Los usuarios deberán cambiar su contraseña cada 45 días como máximo, debido a lo cual 5 días antes de que se cumpla el plazo máximo serán avisados.

Las comunicaciones entre los clientes y el servidor serán seguras.

- **Restricciones en el diseño y la implementación**

El análisis y el diseño de la aplicación se harán bajo las condiciones que impone la Metodología RUP utilizando como lenguaje de modelado a UML. Como herramienta CASE se utilizará *Visual Paradigm* 6.0 para realizar el modelado de los artefactos que se obtienen como consecuencia de los flujos de trabajo que define RUP y como herramienta para la confección de los prototipos no funcionales del sistema se utilizará *Macromedia Dreamweaver* 8. Además se utilizará a *Symfony* como *framework* y debido a que este utiliza como lenguaje de programación PHP 5 será este el lenguaje que se utilice para la implementación Web de la aplicación, *JavaScript* para la programación de escritorio y *PostgreSQL* como gestor de Base de Datos.

- **Requerimientos de Rendimiento**

A pesar de que no es necesario que las solicitudes de los clientes tengan una velocidad de respuesta como los sistemas de tiempo real, sí es necesario garantizar la rapidez de respuesta del sistema. Debido a que la aplicación está concebida para un ambiente Cliente/Servidor, debe garantizarse que los tiempos de respuestas sean generalmente rápidos al igual que la velocidad de procesamiento de la información.

- **Requerimientos de Portabilidad**

La aplicación podrá ser usada sobre los sistemas operativos *Windows* y *Linux*.

- **Requerimientos Políticos-Culturales**

En la aplicación se usará el idioma español. Las imágenes que se usarán estarán relacionadas con el tema de estudio. En caso de ser necesario realizar algún cambio en la aplicación este será tramitado por la VR de logística o la DGA.

- **Requerimientos de extensibilidad**

Se debe lograr un diseño adaptable, con la capacidad de poder soportar funcionalidades adicionales o modificar las funcionalidades existentes sin afectar a los restantes requerimientos contemplados en el sistema.

2.2.3 Casos de Uso del Sistema.

1. Verificar Comensales.
2. Obtener Orden de Producción y Solicitud de Materiales.
3. Buscar Balance Nutricional Real y Planificado.
4. Buscar Normas Nutricionales.
5. Buscar Vale de salida.
6. Mostrar comensales planificados y reales.
7. Obtener Demanda de alimentos para un periodo.
8. Obtener Demanda Real de alimentos.

2.2.4 Actores del Sistema.

Actores:	Descripción:
Planificador del Menú	Manipula toda la información referente a la Orden de producción, Solicitud de materiales y Demanda de alimentos.
Económico de UPS (Unidad de Producción y servicios)	Realiza consultas de información en el sistema.
Gestor_Información	Es una generalización del rol de Planificador del menú y Económico de la Dirección General de Alimentos. Es el encargado de realizar consultas relacionadas con los comensales planificados y reales, Buscar Balance Nutricional Real y Planificado, Buscar Normas Nutricionales y

	Obtener Demanda de alimentos.
Sistema de control de acceso	Sistema que brinda reportes de los comensales que realmente accedieron al comedor, así como los que reciben el servicio dos veces.
ASSETS-NS	Sistema que brinda información referente al Vale de salida.

Tabla 6 Descripción los Actores del Sistema

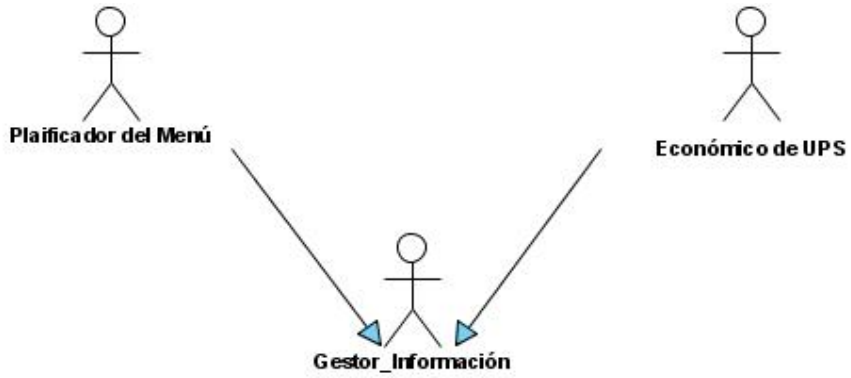

Figura 6 Diagrama de generalización de los actores del sistema.

2.2.5 Matriz de Trazabilidad

Para el logro de una exitosa gestión de los requisitos de un software se hace necesario hacer una correcta trazabilidad de los mismos, pues mediante esta podemos conocer los elementos que se afectan al ocurrir algún cambio en algún otro elemento que tenga relación con el primero. Para hacer más sencilla esta tarea se utiliza la matriz de trazabilidad, ya que tiene como objetivo verificar si los casos de uso que se tienen satisfacen todos los requerimientos del sistema, o sea:

La matriz de trazabilidad es una herramienta que ayuda a seguir, ordenar y mostrar como realizamos la validación de un sistema partiendo de los requerimientos de usuarios y mostrando en cada etapa de calificación como verificamos los mismos (13).

Esta herramienta nos proporciona numerosos beneficios, dentro de los que podemos destacar:

- Facilita la administración y seguimiento de los elementos críticos del análisis de riesgo.

- Permite una mejor visualización del alcance de la calificación y de los test.
- Ayuda a demostrar que la validación se encuentra completa.
- Ayuda a la administración de cambios y la visualización del impacto de los mismos en la calificación.
- Posee un gran valor agregado en el momento de las auditorias y las inspecciones mostrando a través de la misma un mapa general de la validación. (13)

Para ver la Matriz de Trazabilidad remitirse al el expediente de proyecto.

2.2.6 Diagrama de Casos de Uso del Sistema.

Los diagramas de Casos de Uso del sistema representan la interacción que existe entre los elementos externos al sistema (sus actores) con el propio sistema.

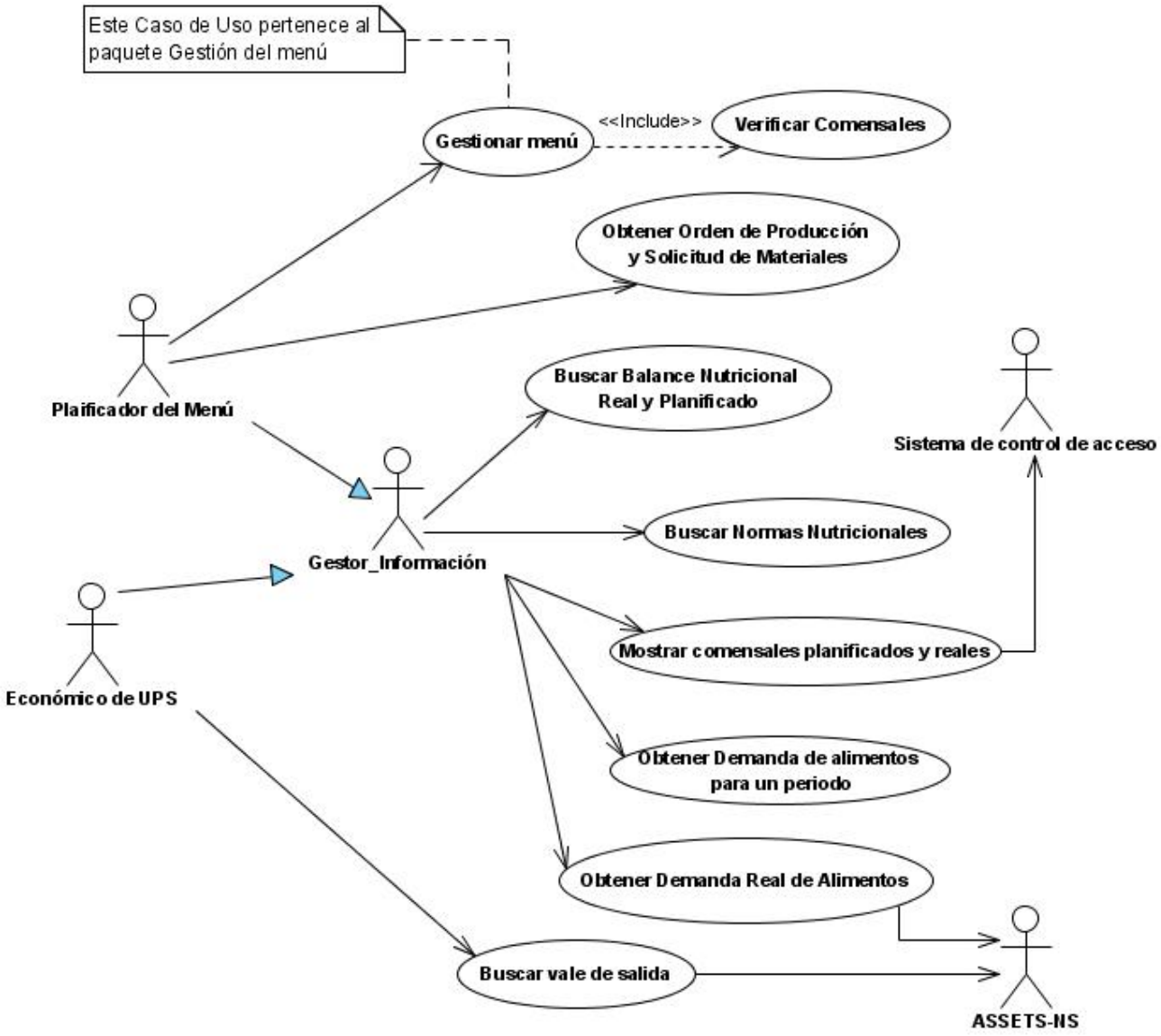

Figura 7 Diagrama General de Casos de uso del Sistema.

2.2.7 Descripciones de los Casos de Uso del Sistema.

Nombre del Caso de Uso	Verificar comensales.
Actores	Planificador del Menú
Propósito	Comprobar si el menú planificado cubre la cantidad de comensales propuesta.
Resumen	Este proceso es incluido del CU “Gestionar Menú”, del paquete

	Gestión del menú ya que existe una retroalimentación entre ambos, o sea, cada vez que se va a confeccionar un menú se comprueba si este cubre la cantidad planificada de comensales. Cuando el Planificador del menú inicia el proceso de gestionar un menú, solicita verificar si el menú elaborado cubre la cantidad de comensales planificada. Luego el sistema calcula la cantidad de materias primas necesarias para elaborar el menú y comprueba esta cantidad con el listado de productos disponibles. En caso de que no alcance algún producto se muestra un mensaje indicando que se debe reajustar el menú ya que determinado(s) producto(s) no cubre(n) el número planificado de comensales. Luego el Planificador del menú continúa con el proceso de gestionar el menú.
Referencias	RF 1, CU Gestionar Menú (Inclusión).
Precondiciones	Que sean introducidos correctamente todos los campos del menú.
Poscondiciones	Continúa el proceso de gestionar menú.
Requerimientos especiales	

Flujo normal de los eventos	
Acción del actor	**Respuesta del Sistema**
	1. Teniendo en cuenta el menú entrado y el listado de platos determina para cada plato que conforma el menú, la cantidad de producto (CantP) que se necesita para cubrir 100 comensales, que es el número para el que esta conformado el listado de plato.
	2. Determina para cada producto la cantidad total que se necesita para cubrir los comensales planificados para el evento, usando la fórmula CantTotal = (CantP * CantComens) / 100 • CantTotal = Cantidad Total de cada producto que se necesita para cubrir los comensales planificados.

	• CantP = Cantidad de Producto, calculada en la actividad anterior. • CantComens = Número de comensales planificados.
	3. Busca en cada producto de los diferentes platos y suma la CantTotal de los que sean iguales.
	4. Obtiene un listado con productos (nombre, cantidad) existentes en el almacén, a través de una conexión a la base de datos (BD).
	5. Compara la fecha del menú, del que se esta comprobando la existencia, con la fecha de arribo de cada producto del Plan de carga, y si la del Plan de carga es menor, se selecciona el producto, y de esta forma se obtiene un listado de productos de posible existencia.
	6. Con los nombres y las cantidades de los productos obtenidos en el paso 4 y 5, y se obtiene el listado de productos disponibles.
	7. Compara la CantTotal de cada producto obtenida en el paso 3 con el listado de productos disponibles obtenido en el paso anterior, para comprobar si es mayor o menor.
	8. Si la CantTotal de todos los productos es menor o igual que la cantidad disponible, ir a la actividad 2.2 de la Sección "Crear menú" del caso de uso Gestionar menú, del paquete Gestión del menú.
Flujos alternos	
Acción del actor	**Respuesta del Sistema**
	8.1. Si la CantTotal de algunos de los productos es mayor que la cantidad existente en el almacén, muestra un mensaje indicando que se debe reajustar el menú, debido a que el o los productos x no alcanzan para cubrir los

	comensales planificados. • X = productos(s) cuya CantTotal es mayor que la existencia en almacén.
	8.2. ir a la actividad 1.4 de la Sección "Crear menú" del caso de uso Gestionar menú, del paquete gestión del menú.
Prioridad	Crítico
Prototipo	

Tabla 7 Descripción Textual del Caso de Uso del Sistema “Verificar comensales”.

Nombre del Caso de Uso	Obtener Orden de Producción y Solicitud de Materiales.
Actores	Planificador de Menú (inicia).
Propósito	Obtener la Orden de Producción y la Solicitud de Materiales para un menú determinado.
Resumen	El caso de uso se inicia cuando el Planificador de menú solicita obtener la Orden de producción y la Solicitud de Materiales. El sistema muestra la interfaz correspondiente a la solicitud y ejecuta las acciones necesarias.
Referencias	RF 5, RF 5.1, RF 6, RF 6.1
Precondiciones	Que se haya reajustado el menú del día, para el cual se va a obtener la Orden de producción y la Solicitud de Materiales. Que el planificador del menú este autentificado en el sistema.
Poscondiciones	
Requerimientos especiales	

Flujo normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. Solicita obtener la Orden de Producción y la Solicitud de Materiales.	2. Muestra una interfaz para que el usuario escoja la fecha para la cual desea obtener la Orden de Producción y la Solicitud de Materiales.
3. Escoge una fecha.	4. Muestra una interfaz con los menues confirmados existentes para esa fecha.
5. Selecciona el menú para el cual desea obtener la Orden de Producción y la Solicitud de Materiales y solicita ejecutar la acción.	6. Teniendo en cuenta el menú previamente seleccionado y el listado de platos, confecciona la Orden de Producción.
	7. A partir de la Orden de Producción obtenida, se obtiene la Solicitud de Materiales, agrupando y sumando las cantidades de los productos iguales de la Orden de Producción.
	8. Muestra la Orden de Producción y la Solicitud de Materiales, brindando la posibilidad de guardarlas e imprimirlas.

9. Solicita guardar e imprimir la Orden de Producción y la Solicitud de Materiales.	10. Guarda e imprime la Orden de Producción y la Solicitud de Materiales, finalizando así el Caso de Uso.
Flujos alternos	
Acción del actor	**Respuesta del Sistema**
7.1. No desea imprimir la Orden de Producción y la Solicitud de Materiales y sale del sistema	
Prioridad	Crítico
Prototipo	

Tabla 8 Descripción Textual del Caso de Uso del Sistema “Obtener Orden de Producción y Solicitud de Materiales”.

Nombre del Caso de Uso	Obtener Demanda de alimentos para un período
Actor(es).	Gestor_Información (inicia).
Propósito	Obtener la demanda de alimentos para un periodo determinado.
Resumen	El caso de uso inicia cuando el Gestor_Información solicita obtener la demanda de alimentos para un periodo. El sistema verifica que exista un menú planificado para el periodo seleccionado, y obtiene la demanda de alimentos. Muestra una interfaz con el resultado obtenido.
Referencias	RF 3, RF 3.1
Precondiciones	Que exista el menú planificado. Que el Gestor_Información este autentificado en el sistema.
Poscondiciones	
Requerimientos especiales	

Flujo normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. Solicita obtener la demanda de alimentos para un periodo.	2. Muestra una interfaz dando la opción de seleccionar el periodo para el cual desea obtener la demanda.
3. Selecciona el período para obtener la demanda, y solicita ejecutar la acción.	4. Realiza una búsqueda del menú para el periodo seleccionado.
	5. Teniendo en cuenta el menú se confecciona la demanda de alimentos, sumando de cada plato que conforma el menú, la cantidad de productos que se necesita para el periodo.
	6. Muestra la Demanda de alimentos obtenida para el periodo que abarca el menú seleccionado, brindando la opción de guardar e imprimir.

7. Solicita guardar e imprimir la demanda	8. Guarda e imprime la demanda de alimentos finalizando así el Caso de Uso.
Flujos alternos	
7.1. No solicita realizar ninguna acción y sale de la sección.	5.1 En caso de que el menú no cubra todo el periodo seleccionado, muestra un mensaje indicando que escoja otro periodo para obtener la demanda.
Prioridad	Crítico
Prototipo	

Tabla 9 Descripción Textual del CU del Sistema “Obtener Demanda de alimentos para un período”.

Nombre del Caso de Uso	Buscar Balance Nutricional Real y Planificado.
Actores	Gestor_Información (inicia).
Propósito	Consultar de la Base de Datos (BD) el Balance Nutricional Real y Planificado.
Resumen	El Caso de Uso inicia cuando el Gestor_Información, solicita consultar de la BD el Balance Nutricional Real y Planificado. El sistema realiza la búsqueda y muestra la información.
Referencias	RF 8, RF 8.1, RF 9, RF 9.1
Precondiciones	Que el usuario este autentificado en el sistema, que llene todos los criterios de búsqueda solicitados, que la fecha entrada sea una fecha en tiempo pasado.
Poscondiciones	
Requerimientos especiales	

Tabla 10 Descripción textual de Caso de Uso del Sistema “Buscar Balance Nutricional Real y Planificado”.

Nombre del Caso de Uso	Buscar Normas Nutricionales
Actores	Gestor_Información (inicia).
Propósito	Consultar de la Base de Datos (BD) las Normas Nutricionales.
Resumen	El Caso de Uso inicia cuando el Gestor_Información, solicita consultar las Normas Nutricionales. El sistema muestra la información de la BD.
Referencias	RF 10, RF 10.1
Precondiciones	Que el usuario este autentificado en el sistema.
Poscondiciones	
Requerimientos especiales	

Tabla 11 Descripción Textual del Caso de Uso del Sistema “Buscar Normas Nutricionales”.

Nombre del Caso de Uso	Mostrar comensales planificados y reales.
Actores	Gestor_Información (inicia), Sistema de Control de Acceso.
Propósito	Consultar el comportamiento de los comensales.
Resumen	El caso de uso inicia cuando el Gestor_Información solicita ver el reporte de comensales planificados y los comensales que realmente accedieron al comedor, el sistema muestra una interfaz con la información solicitada brindando la opción de imprimir el reporte.
Referencias	RF 2, RF 2.1
Precondiciones	Que el usuario este autentificado en el sistema.
Poscondiciones	
Requerimientos especiales	

Tabla 12 Descripción Textual del Caso de Uso del Sistema “Mostrar comensales planificados y reales”.

Nombre del Caso de Uso	Obtener Demanda Real de alimentos.
Actores	Gestor_Información (Inicia), ASSETS-NS.
Propósito	Obtener la Demanda Real de alimentos para un período determinado.
Resumen	El caso de uso se inicia cuando el Gestor_Información solicita al sistema obtener la Demanda Real de alimentos para un período determinado. El sistema muestra la interfaz correspondiente y se ejecuta la acción solicitada.
Referencias	RF 4, RF 4.1
Precondiciones	Que existan los vales de salida correspondientes al periodo o evento para el cual se quiere confeccionar la demanda. Que el Gestor_Información este autentificado en el sistema e ingrese todos los datos solicitados.
Poscondiciones	
Requerimientos especiales	

Tabla 13 Descripción Textual del Caso de Uso del Sistema “Obtener Demanda Real de alimentos”.

Nombre del Caso de Uso	Buscar vale de salida
Actores	Económico de UPS (inicia), ASSETS-NS.
Propósito	Consultar el vale de salida del almacén.
Resumen	El económico solicita ver el vale de salida asociado a una solicitud de materiales ingresando el código de esta. El sistema muestra una interfaz con el vale de salida.
Referencias	RF 7
Precondiciones	Que exista la solicitud de materiales asociada. Que el Económico este autentificado en el sistema.
Poscondiciones	
Requerimientos especiales	

Tabla 14 Descripción Textual del Caso de Uso del Sistema “Buscar vale de salida”.

Para ver las descripciones de cada uno de los flujos de eventos de cada caso de uso del sistema remitirse al expediente de proyecto.

2.2.8 Vista de casos de Uso.

La vista de la arquitectura del modelo representa los casos de uso que describen alguna funcionalidad crítica para el sistema, implican algun requisito significativo que deba desarrollarse pronto, o que cubre una funcionalidad importante de la arquitectura del sistema final.

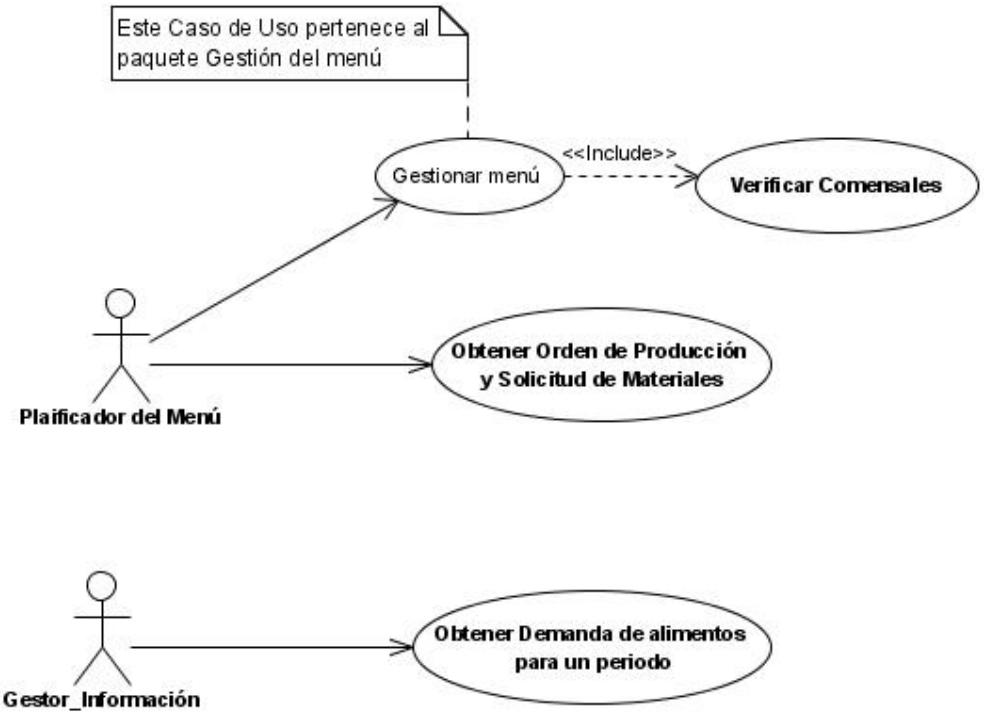

Figura 8 Vista de Casos de Uso del Sistema

2.3 Consideraciones del capitulo.

Este capítulo especifica todo lo referente al flujo de trabajo de modelado del negocio y del sistema, especificando los diferentes artefactos de cada uno. Se detalla el modelo de negocio actual, así como las reglas y trabajadores del negocio, mostrando los diagramas correspondientes. Además para el modelado del sistema se describen los requerimientos funcionales y no funcionales que debe tener el software, los casos de uso y actores del sistema, así como los diagramas correspondientes.

Capitulo 3: Análisis y Diseño

En este capítulo se obtienen los artefactos correspondientes al flujo de trabajo “Análisis y Diseño”. Se modela el diseño, donde se representan las realizaciones de los casos de uso del sistema, a través de las clases del diseño con extensiones Web además de las descripciones de las mismas.

3.1 Arquitectura del sistema.

La arquitectura del sistema es la organización fundamental de un sistema incorporada en sus componentes y los principios que guían su diseño y evolución, facilita una descripción entendible de la arquitectura del sistema software. Contiene varias vistas entre las que se encuentran la Vista de Casos de Uso, Vista Lógica, Vista de Despliegue, Vista de Procesos y la Vista de Implementación, en esta investigación se hará uso de la Vista Lógica y la Vista de Despliegue.

3.1.1 Vista Lógica

La vista lógica representa de manera independiente los componentes principales del diseño y sus relaciones. Muestra un subconjunto del modelo del diseño, así como los elementos más importantes para la arquitectura del sistema. Describe las clases más importantes, su organización en paquetes y subsistemas y estos en capas.

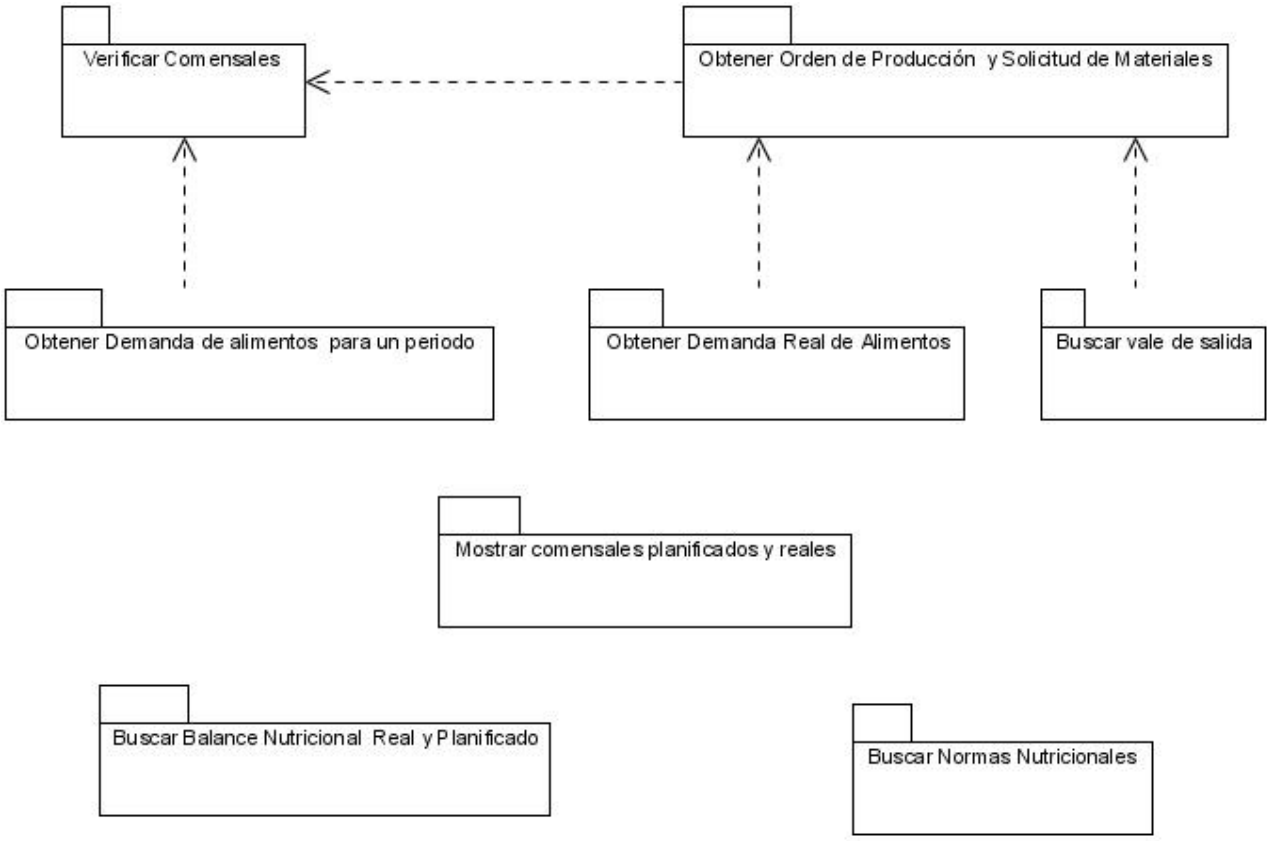

Figura 9 Diagrama de la Vista Lógica.

3.1.2 Vista de despliegue.

La Vista de despliegue suministra una base para la comprensión de la distribución física de un sistema a través de nodos. Se ocupa de los requerimientos no funcionales.

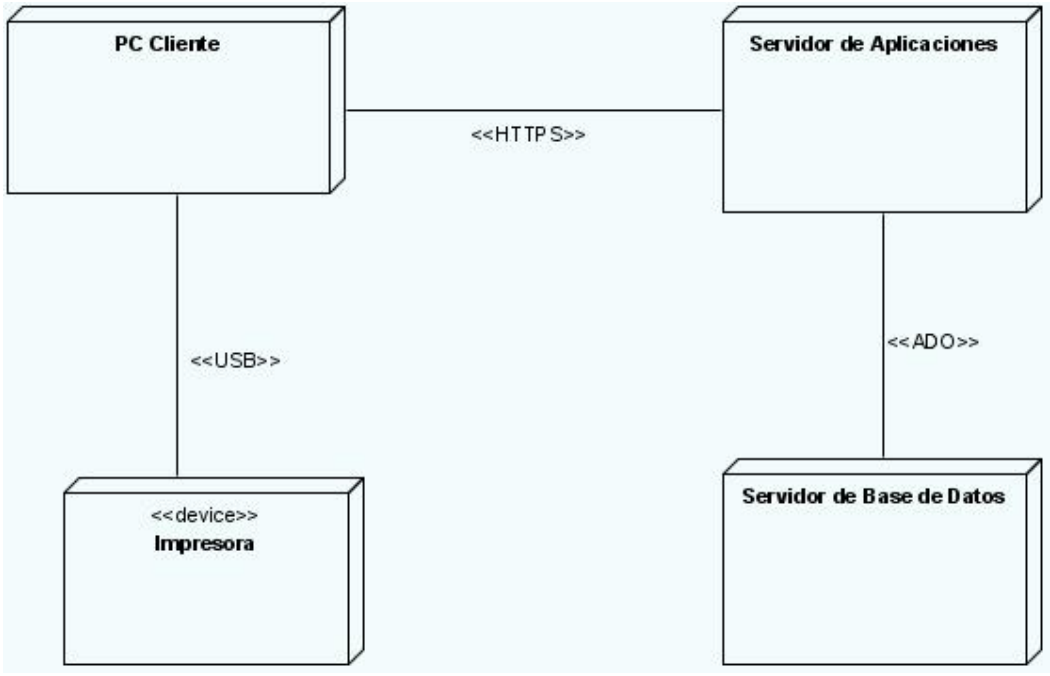

Figura 10 Diagrama de la Vista de Despliegue.

3.2 Prototipos de la interfaz de usuario.

Los Prototipos de interfaz no funcionales representan el punto de interacción entre el usuario y la computadora. Proporcionan una visión más acertada de lo que será el producto, dando claridad de lo que se requiere y lo que es factible. Se construyen con el objetivo de explotar y validar el diseño de la aplicación, asegurando que este sea correcto y no se pierda tiempo en el desarrollo.

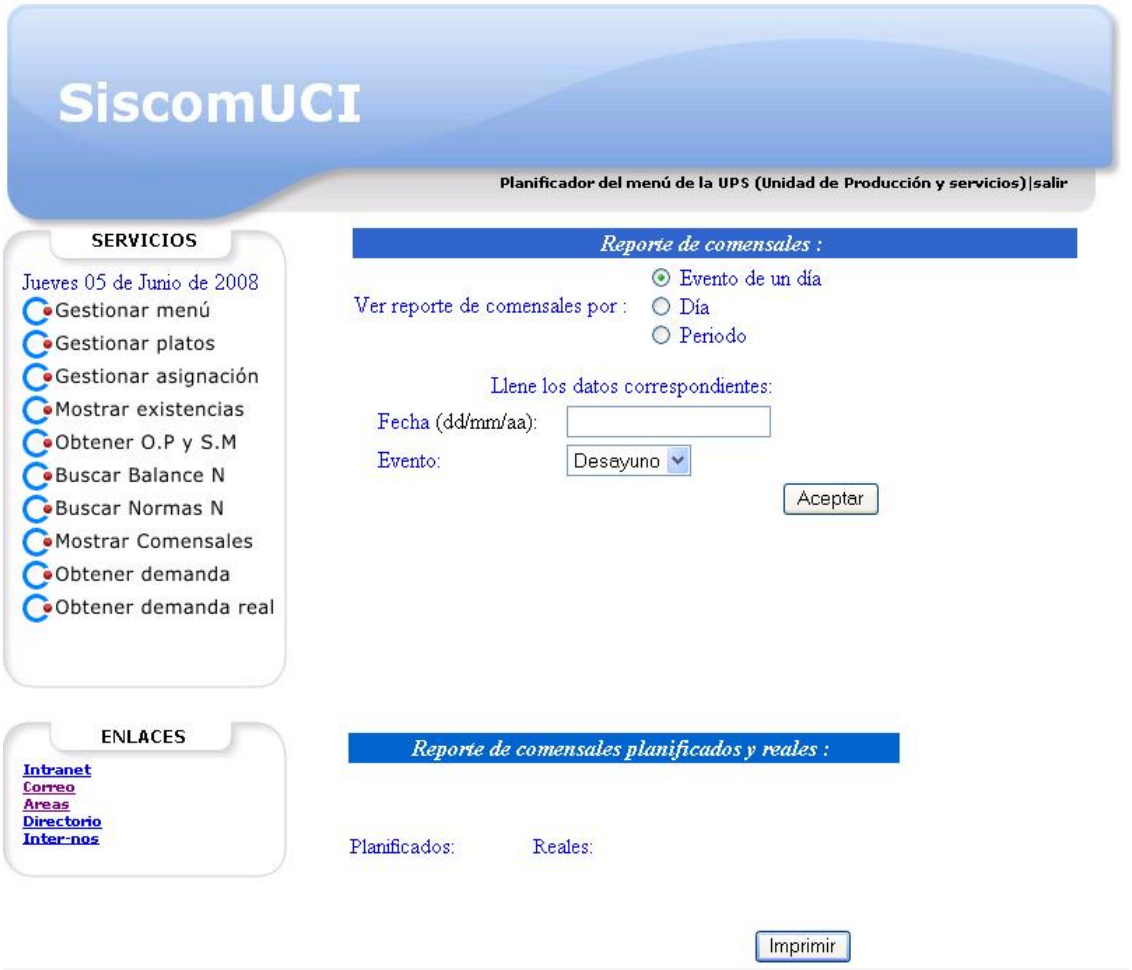

Figura 11 Prototipo de interfaz de usuario del CU Mostrar comensales planificados y reales.

Figura 12 Prototipo de interfaz de usuario del CU Buscar Balance Nutricional Real y Planificado.

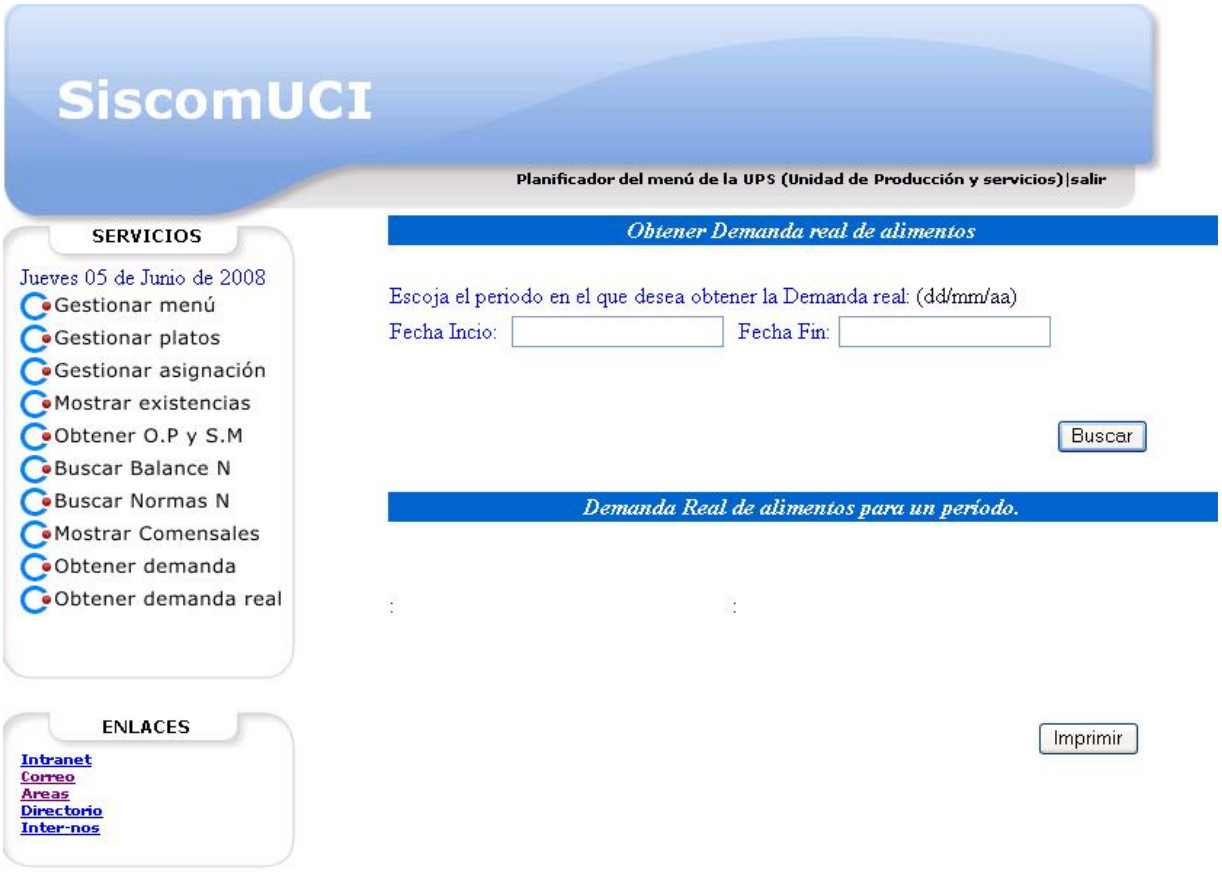

Figura 13 Prototipo de interfaz de usuario del CU Obtener Demanda real de alimentos

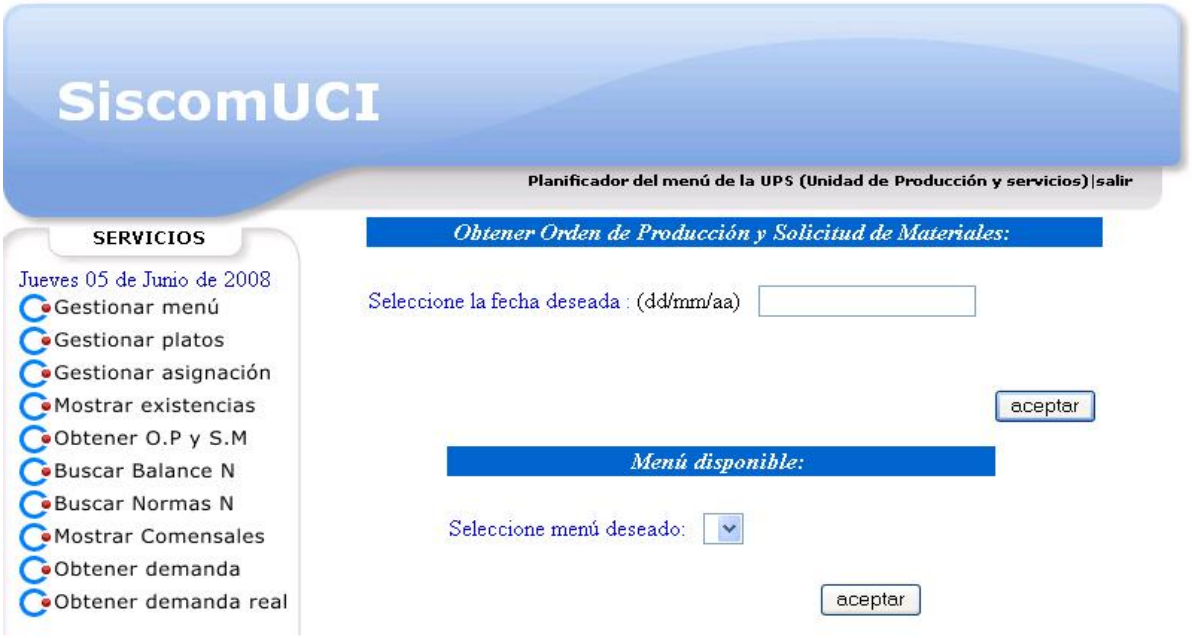

Figura 14 Prototipo de interfaz de usuario del CU Obtener Orden de Producción y Solicitud de Materiales

3.3 Mapa de navegación.

Un mapa de navegación es la representación gráfica de la organización de la información de una estructura web. Permite elaborar escenarios de comportamiento de los usuarios y expresa todas las relaciones de jerarquía y secuencia.

La importancia de elaborar un mapa de navegación del sitio web radica en la comprensión del orden de presentación de las páginas web y la flexibilidad de moverse entre ellas (hipervínculos).

En el diseño del mapa de navegación se debe seleccionar la página de entrada al sitio web, ordenar de manera jerárquica las páginas con los contenidos (por niveles o categorías) y establecer los vínculos entre ellas permitiendo una navegación hipertextual.

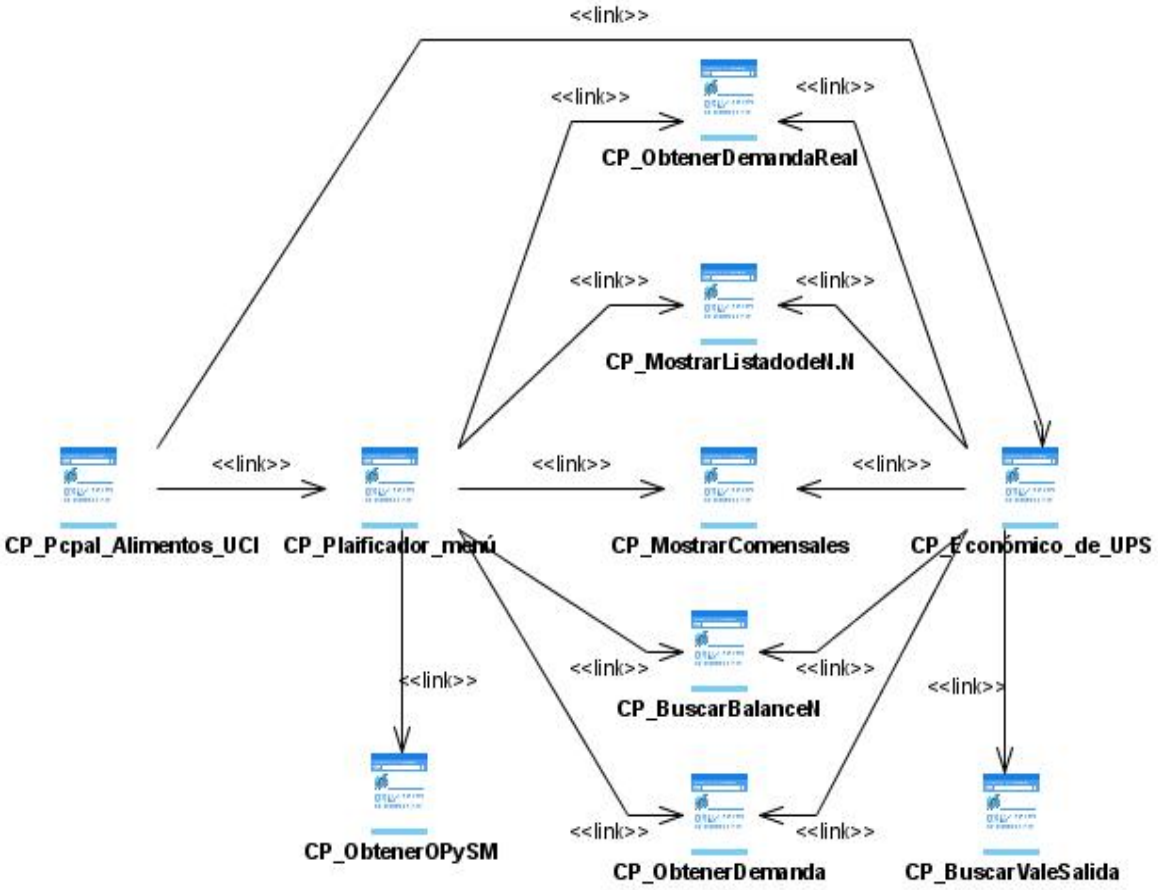

Figura 15 Mapa de navegación.

3.4 Aplicación de los patrones de diseño.

En el presente epírafe se explica el uso de los patrones de diseño mencionados en el primer capítulo , detallando donde se aplica cada uno.

El **Bajo acoplamiento** se utiliza para desacoplar las vistas de los modelos y desacoplar los modelos de la forma en que se muestran e ingresan los datos, además se pone en evidencia el uso de este patrón por ejemplo, en la clase Actions quien hereda solamente de sfActions lográndo un bajo acoplamiento de clases.

La **Alta cohesión** se evidencia en casa elemento diseñado ya que cada cual está altamente especializado en su tarea por ejemplo la vista en mostrar los datos al usuario, el controlador en las entradas y el modelo en su objetivo del negocio. Symfony admite la asignación de responsabilidades con una alta cohesión, como ejemplo de ellos se tiene la clase Actions la cual es responsable de

definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones (instanciar objetos y acceder a las properties), es decir, que está formada por diferentes funcionalidades las que se encuentran estrechamente relacionadas lo que hace posible que el software sea flexible frente a grandes cambios

El **Experto** es uno de los más utilizados, por ejemplo: en Propel, el cual es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, se encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

El **Controlador** se pone de manifiesto en el controlador frontal pues este maneja todas las peticiones Web que se realizan al sistema, ya que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando este recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

El **Creador**, se evidencia en la clase Intermedia se encuentran las acciones definidas para dar respuesta a cada funcionalidad del sistema. En la misma se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Intermedia es "creador" de dichas entidades.

La **Fabricación pura** se pone de manifiesto en la cla Intermedia ya que el objetivo de la misma es manejar el acceso a los datos de las clases del Modelo para lo cual tiene definido un grupo de funcionalidades que responden a una acción que debe ejecutar el sistema y que son llamadas a través de la clase Actions.

Debido a la arquitectura MVC y a la composición que tienen cada una de estas partes en Symfony se hará uso implícitamente del patrón ***Decorator*** , específicamente en la Vista ya que esta está compuesta por diversas partes dentro de las que está la plantilla y el *layout*. Este último es el que almacena el código común para todas las páginas de la aplicación, por lo que el contenido de la plantilla se integra a él, es decir, es el que decora la plantilla.

El patrón ***Singlenton*** se utiliza en la clase Intermedia, diseñada en el Modelo y es la encargada de la lógica del negocio, pues la misma tiene un atributo de tipo Singleton (instancia) a través del cual la clase Actions accede a todas las funcionalidades de esta clase. Además se evidencia en el controlador frontal en el cual se puede hacer llamada a sfContext::getInstance(), en una acción, el método getContext(), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony.

El ***Composite*** es usado fundamentalmente en las vistas, pues estas están compuestas por diferentes elementos como son: el layout, la plantilla, los elementos parciales.

El patrón ***Facade*** se utiliza en la clase Intermedia con el objetivo de simplificar el acceso a un conjunto de clases, las del modelo, para ello en esta se define un grupo de métodos, uno para cada operación permitida de modo que sean estos métodos los que internamente hagan las operaciones con el fin de llevar a cabo la correcta lógica de la aplicación.

El patrón ***Chaing of Responsibility*** es usado en todas las clases diseñadas puesto que cada una de ellas tendrá una responsabilidad en el proceso solicitud – respuesta de una petición realizada por un usuario.

3.5 Modelo del diseño.

En el flujo de trabajo "Análisis y Diseño" el Modelo de Diseño es uno de los artefactos fundamentales ya que es una abstracción de la implementación del sistema, como resultado de un análisis realizado a los requerimientos no funcionales del mismo. El diseño no debe ser ambiguo para obtener un modelo final suficiente para la implementación. Este modelo es el más importante del flujo de trabajo, consiste en colaboraciones de clases que pueden ser agrupadas en subsistemas y paquetes además de describir la realización de los casos de uso.

3.5.1 Realización de Casos de Uso

El artefacto "Realización de casos de Uso", describe como un caso de uso particular es realizado dentro del modelo de diseño, en términos de colaboración de paquetes (14).

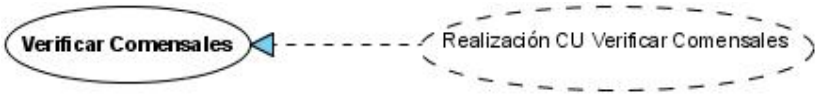

Figura 16 Realización del CU verificar Comensales.

Figura 17 Realización del CU Obtener orden de Producción y Solicitud de Materiales

.

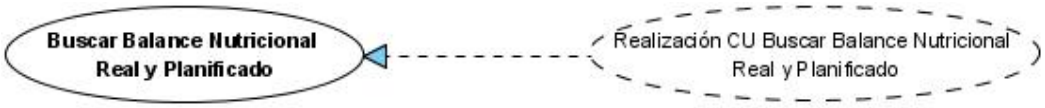

Figura 18 Realización del CU Buscar Balance Nutricional Real y Planificado.

.

Figura 19 Realización del CU Buscar Normas Nutricionales.

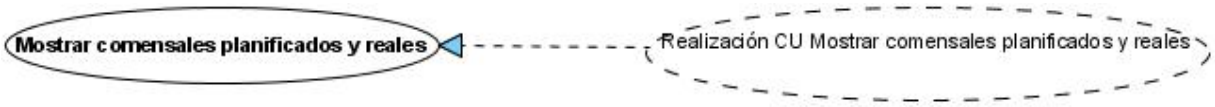

Figura 20 Realización del CU Mostrar comensales planificados y reales.

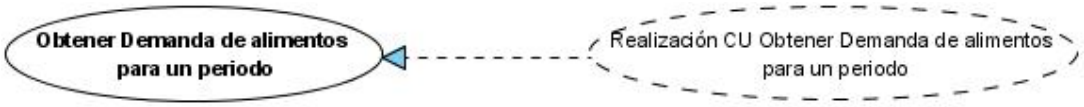

Figura 21 Realización del CU Obtener Demanda de alimentos para un periodo.

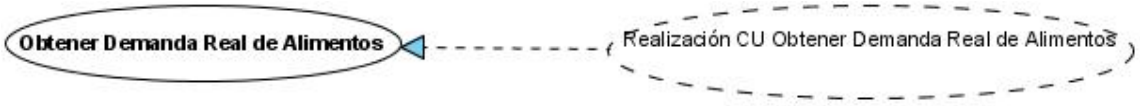

Figura 22 Realización del CU Obtener Demanda Real de Alimentos.

.

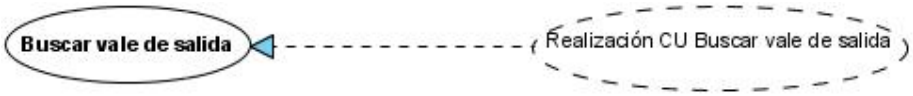

Figura 23 Realización del CU Buscar vale de salida.

3.5.2 Descripciones de las Clases del Diseño.

Nombre:	ProductoPeer	
Tipo de clase	Clase Peer (Permite la comunicación con la Base de Datos, es decir proporciona los medios necesarios para obtener los registros de las tablas)	
Atributo		**Tipo**
Para cada responsabilidad:		
Nombre:	doSelect()	
Descripción:	Permite obtener un listado de objetos con todos los datos del Producto que se encuentran en las tablas (en dependencia del objeto tipo Criteria que se le pase por parámetro)	

Tabla 15 Descripción de la Clase del diseño de Aplicaciones Web: ProductoPeer

Nombre:	Obtener_OP_SM_Actions	
Tipo de clase	Clase Actions (Clase que utiliza el modelo y crea variables para la vista)	
Atributo		**Tipo**
Nombre:	executeObtenerOPySM()	
Descripción:	Permite obtener la orden de producción y la solicitud de materiales.	

Tabla 16 Descripción de la Clase del diseño de Aplicaciones Web: Obtener_OP_SM_Actions.

Nombre:	Intermedia_Obtener_OP_SM	
Tipo de clase	Clase encargada de la lógica del negocio	
Atributo		**Tipo**
$instancia		Singlenton
Nombre:	obtenerOPySM()	
Descripción:	Permite obtener la orden de producción y la solicitud de materiales.	
Nombre:	listarMenusConfirmados()	
Descripción:	Permite mostrar los menues confirmados.	
Nombre:	getInstancia()	
Descripción:	Devuelve la instancia	

Tabla 17 Descripción de la Clase del diseño de Aplicaciones Web: Intermedia_Obtener_OP_SM

Nombre:	FP_ObtenerOPySM
Tipo de clase	Formulario (Formulario de la página cliente ObtenerOPySM)
Atributo	**Tipo**
fecha	textfield
AceptarF	button
menu	select
aceptarM	button
GuardarImprimir	button

Tabla 18 Descripción de la Clase del diseño de Aplicaciones Web: FP_ObtenerOPySM.

Nombre:	ComensalesReales
Tipo de clase	Clase de tipo Interface que interactúa con el sistema externo
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	obtenercomensalesRealesEvento()
Descripción	Permite mostrar los comensales que realmente accedieron al comedor en un evento.
Nombre:	obtenercomensalesRealesDia()
Descripción	Permite mostrar los comensales que realmente accedieron al comedor en un dia.
Nombre:	obtenercomensalesRealesPeriodo()
Descripción	Permite mostrar los comensales que realmente accedieron al comedor en un periodo.

Tabla 19 Descripción de la Clase del diseño de Aplicaciones Web: ComensalesReales.

Nombre:	Vale_Salida
Tipo de clase	Clase de tipo Interface que interactúa con el sistema externo
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Vale_Salida()
Descripción:	Permite consultar el vale de salida asociado a una solicitud de materiales.
Nombre:	obtenerDemandaR()
Descripción:	Permite obtener la demanda real de alimentos para un periodo.

Tabla 20 Descripción de la Clase del diseño de Aplicaciones Web: Vale_Salida.

Nombre:	BaseEvento
Tipo de clase	Clase Base (Clase que se genera directamente a partir del esquema)

Atributo	**Tipo**
$id_evento	
$tipo_evento	
$nn_evento	

Para cada responsabilidad:	
Nombre:	getTipoevento()
Descripción:	Devuelve el tipo de evento
Nombre:	setTipoevento()
Descripción:	Cambia el tipo de evento
Nombre:	getNormanutric()
Descripción:	Devuelve la norma nutricional de un evento
Nombre:	setNormanutric()
Descripción:	Cambia el norma nutricional de un evento
Nombre:	getIdevento()
Descripción:	Devuelve el identificador de un evento
Nombre:	setIdevento()
Descripción:	Cambia el identificador de un evento
Nombre:	save()
Descripción:	Función propia de la Base para salvar en la entidad los nuevos datos que se entran
Nombre:	dosave()
Descripción:	Función llamada por la función save()

Tabla 21 Descripción de la Clase del diseño de Aplicaciones Web: BaseEvento.

Nombre:	BaseMenu
Tipo de clase	Clase Base (Clase que se genera directamente a partir del esquema)

Atributo	**Tipo**
$id_evento	
$id_up	
$id_menu	
$estado	
$fecha	
$cant_comensales	
$ingresos	

Para cada responsabilidad:	
Nombre:	getIdmenu()
Descripción:	Devuelve el identificador de un menú
Nombre:	setIdMenu()
Descripción:	Cambia el identificador de un menú
Nombre:	getUP()
Descripción:	Devuelve el identificador de la unidad productiva
Nombre:	setUP()
Descripción:	Cambia el identificador de la unidad productiva
Nombre:	getEvento()
Descripción:	Devuelve el identificador de un evento
Nombre:	setEvento()
Descripción:	Cambia el identificador de un evento
Nombre:	getFecha()
Descripción:	Devuelve la fecha un menú
Nombre:	setFecha()
Descripción:	Cambia la fecha de un menú
Nombre:	getEstado()
Descripción:	Devuelve el estado de un menú
Nombre:	setEstado()
Descripción:	Cambia el estado de un menú

Nombre:	getCantcomensales()
Descripción:	Devuelve la cantidad de comensales
Nombre:	setCantcomensales
Descripción:	Cambia la cantidad de comensales
Nombre:	getIngresos()
Descripción:	Devuelve los ingresos
Nombre:	setIngresos()
Descripción:	Cambia los ingresos
Nombre:	save()
Descripción:	Función propia de la Base para salvar en la entidad los nuevos datos que se entran
Nombre:	dosave()
Descripción:	Función llamada por la función save()

Tabla 22 Descripción de la Clase del diseño de Aplicaciones Web: BaseMenu.

Nombre:	BasePeriodo
Tipo de clase	Clase Base (Clase que se genera directamente a partir del esquema)

Atributo	**Tipo**
$fecha_inicial	
$fecha_final	
$id_periodo	

Para cada responsabilidad:	
Nombre:	getFechaInicio()
Descripción:	Devuelve la fecha de inicio de un período
Nombre:	setFechaInicio()
Descripción:	Cambia la fecha de inicio de un período
Nombre:	getFechaFin()
Descripción:	Devuelve la fecha del fin de un período
Nombre:	setFechaFin()
Descripción:	Cambia la fecha del fin de un período
Nombre:	getIdPeriodo ()
Descripción:	Devuelve el identificador de un período
Nombre:	setIdPeriodo()
Descripción:	Cambia el identificador de un período
Nombre:	save()
Descripción:	Función propia de la Base para salvar en la entidad los nuevos datos que se entran
Nombre:	dosave()
Descripción:	Función llamada por la función save()

Tabla 23 Descripción de la Clase del diseño de Aplicaciones Web: BasePeriodo.

Nombre:	BasePlato_Menu
Tipo de clase	Clase Base (Clase que se genera directamente a partir del esquema)

Atributo	**Tipo**
$id_plato	
$id_menu	
$cantidad_plato	
$peso_plato	

Para cada responsabilidad:	
Nombre:	getPlato()
Descripción:	Devuelve el identificador de un plato
Nombre:	setPlato()
Descripción:	Cambia el identificador de un plato
Nombre:	getIdmenu()
Descripción:	Devuelve el identificador de un menú
Nombre:	setIdmenu()
Descripción:	Cambia el identificador de un menú
Nombre:	getCantplatos()
Descripción:	Devuelve la cantidad de platos de un menú
Nombre:	setCantplatos()
Descripción:	Cambia la cantidad de platos de un menú
Nombre:	getPesoPlato()
Descripción:	Devuelve el peso de un plato
Nombre:	setPesoPlato()
Descripción:	Cambia el peso de un plato
Nombre:	save()
Descripción:	Función propia de la Base para salvar en la entidad los nuevos datos que se entran
Nombre:	dosave()
Descripción:	Función llamada por la función save()

Tabla 24 Descripción de la Clase del diseño de Aplicaciones Web: BasePlato_Menu.

Nombre:	BasePlato
Tipo de clase	Clase Base (Clase que se genera directamente a partir del esquema)

Atributo	**Tipo**
$id_plato	
$nombre_plato	
$coccion	
$tiempo	
$temperatura	
$estado	
$unidad_medida	

Para cada responsabilidad:	
Nombre:	getIdPlato()
Descripción:	Devuelve el identificador de un plato
Nombre:	setIdPlato()
Descripción:	Cambia el identificador de un plato
Nombre:	getNombre()
Descripción:	Devuelve el nombre de un plato
Nombre:	setNombre()
Descripción:	Cambia el nombre de un plato
Nombre:	getCoccion ()
Descripción:	Devuelve el parámetro cocción de un plato
Nombre:	setCoccion ()
Descripción:	Cambia el parámetro cocción de un plato
Nombre:	getTiempo()
Descripción:	Devuelve el tiempo de cocción de un plato
Nombre:	setTiempo()
Descripción:	Cambia el tiempo de cocción de un plato
Nombre:	getTemperatura()
Descripción:	Devuelve la temperatura a la que debe cocinarse un plato
Nombre:	setTemperatura()

Descripción:	Cambia la temperatura a la que debe cocinarse un plato
Nombre:	getEstado()
Descripción:	Devuelve el estado de un plato
Nombre:	setEstado()
Descripción:	Cambia el estado de un plato
Nombre:	getUM()
Descripción:	Devuelve la unidad de medida de un plato
Nombre:	setUM()
Descripción:	Cambia la unidad de medida de un plato
Nombre:	save()
Descripción:	Función propia de la Base para salvar en la entidad los nuevos datos que se entran
Nombre:	dosave()
Descripción:	Función llamada por la función save()

Tabla 25 Descripción de la Clase del diseño de Aplicaciones Web: BasePlato.

Nombre:	BaseProducto_Plato
Tipo de clase	Clase Base (Clase que se genera directamente a partir del esquema)

Atributo	**Tipo**
$ cant_producto	
$unidad_medida	
$id_plato	
$id_producto	

Para cada responsabilidad:	
Nombre:	getCantProd()
Descripción:	Devuelve la cantidad de un producto
Nombre:	setCantProd()
Descripción:	Cambia la cantidad de un producto
Nombre:	getUM()
Descripción:	Devuelve la unidad de medida de un producto
Nombre:	setUM()
Descripción:	Cambia la unidad de medida de un producto
Nombre:	getIdProducto
Descripción:	Devuelve el identificador del producto
Nombre:	setIdProducto
Descripción:	Cambia el identificador del producto
Nombre:	getIdPlato
Descripción:	Devuelve el identificador del plato
Nombre:	setIdPlato
Descripción:	Cambia el identificador de un plato
Nombre:	save()
Descripción:	Función propia de la Base para salvar en la entidad los nuevos datos que se entran
Nombre:	dosave()
Descripción:	Función llamada por la función save()

Tabla 26 Descripción de la Clase del diseño de Aplicaciones Web: BaseProducto_Plato.

Nombre:	BaseProducto
Tipo de clase	Clase Base (Clase que se genera directamente a partir del esquema)

Atributo	**Tipo**
$nombre_producto	
$id_producto	
$unidad_medida	
$norma_consumo	
$categoria	
$cant_disponible	

Para cada responsabilidad:	
Nombre:	getNombre()
Descripción:	Devuelve el nombre de un producto
Nombre:	setNombre()
Descripción:	Cambia el nombre de un producto
Nombre:	getID ()
Descripción:	Devuelve el identificador de un producto
Nombre:	setID ()
Descripción:	Cambia el identificador de un producto
Nombre:	getUM()
Descripción:	Devuelve la unidad de medida de un producto
Nombre:	setUM()
Descripción:	Cambia la unidad de medida de un producto
Nombre:	getNormaConsumo()
Descripción:	Devuelve la norma de consumo de un producto
Nombre:	setNormaConsumo()
Descripción:	Cambia la norma de consumo de un producto
Nombre:	getCategoria()
Descripción:	Devuelve la categoría de un producto
Nombre:	setCategoria()
Descripción:	Cambia la categoría de un producto
Nombre:	getCantidadDisponible()

Descripción:	Devuelve la cantidad disponible de un producto
Nombre:	setCantidadDisponible()
Descripción:	Cambia la cantidad disponible de un producto
Nombre:	save()
Descripción:	Función propia de la Base para salvar en la entidad los nuevos datos que se entran
Nombre:	dosave()
Descripción:	Función llamada por la función save()

Tabla 27 Descripción de la Clase del diseño de Aplicaciones Web: BaseProducto.

3.5.3 Diagramas de Clases del diseño

Una construcción similar en la implementación del sistema son las clases del diseño, pues usan el mismo lenguaje que el utilizado en la implementación teniendo una correspondencia directa con sus métodos y los de la implementación.

Los *diagramas de clases* muestran las clases del sistema, así como un conjunto de interfaces, colaboraciones y sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema.

Debido a la utilización del *framework Symfony* se introduce el uso del patrón de arquitectura Modelo-Vista-Controlador, por lo que los diagramas de clases del diseño correspondientes reflejarán cómo tienen lugar las interacciones entre los diferentes elementos y clases del sistema que los componen. Cada una de las peticiones realizadas por los usuarios es atendida por un controlador frontal (*Index*), el cual, pasándole el módulo y la acción a realizar, le encomienda a un componente de *Symfony* procesar la solicitud del usuario. Este componente además de verificar la seguridad del sistema, es decir, aparte de verificar que el usuario que realiza la petición tenga los privilegios necesarios para acceder a esa funcionalidad del sistema le informa a la clase *Actions* la acción que debe realizar, la misma es la encargada de recuperar la información necesaria de las clases contenidas en el modelo y de notificar a la plantilla para que construya, con los datos requeridos, las páginas clientes.

A continuación se presentan los diagramas de clases del diseño, mostrando en el diagrama de clases del diseño (web) para el caso de uso Buscar Normas Nutricionales cómo quedaría el paquete del modelo.

En los restantes diagramas, en el paquete del modelo se representa solo una de clases que contiene este paquete para cada CU, especificándose mediante una nota las que se utilizan en cada uno de los procesos, por la particularidad del *framework* utilizado pues este genera por cada tabla de la Base de Datos cuatro clases. Debido a esto se tiene un paquete Modelo general en que se encuentran todas las clases necesarias para la ejecución de los procesos relacionados con la gestión de comensales.

En los diagramas de clases del diseño (web) para los CU Obtener Demanda Real de Alimentos y Buscar vale de salida se utiliza el subsistema ASSETS-NS el cual es un sistema del cual se obtiene información referente a los productos que realmente se extrajeron del almacén, a través de la clase de tipo *interface* Vale_Salida, mostrando esta información a través de la confección del vale de salida y permitiendo además confeccionar la demanda real de alimentos para un periodo. En el diagrama correspondiente al CU Mostrar comensales planificados y reales se utiliza el subsistema Sistema de control de acceso del cual se extrae la información referente a los comensales que realmente accedieron al comedor a través de la clase de tipo *interface* ComensalesReales.

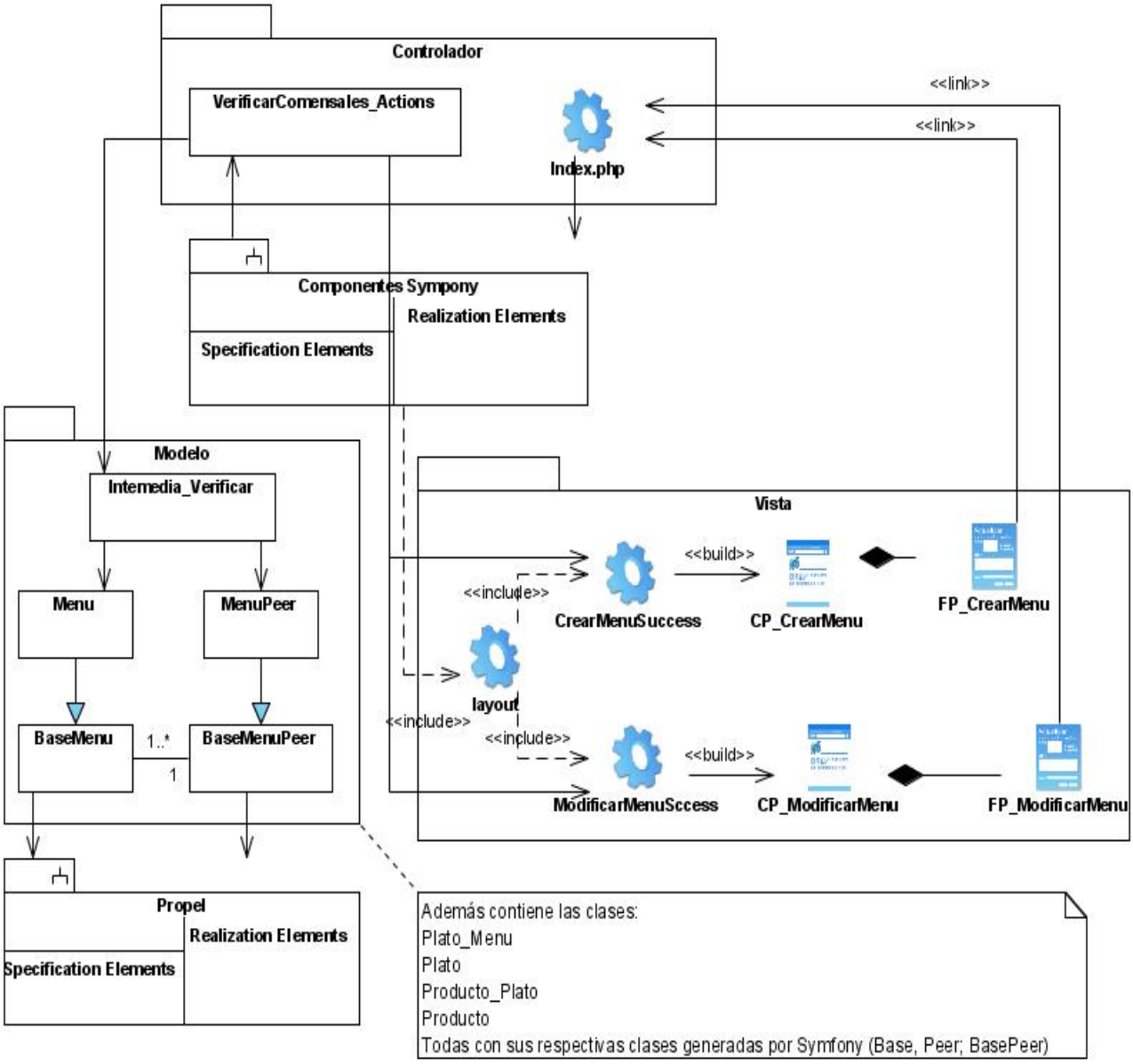

Figura 24 Diagrama de clases del diseño (Web) del CU Verificar Comensales.

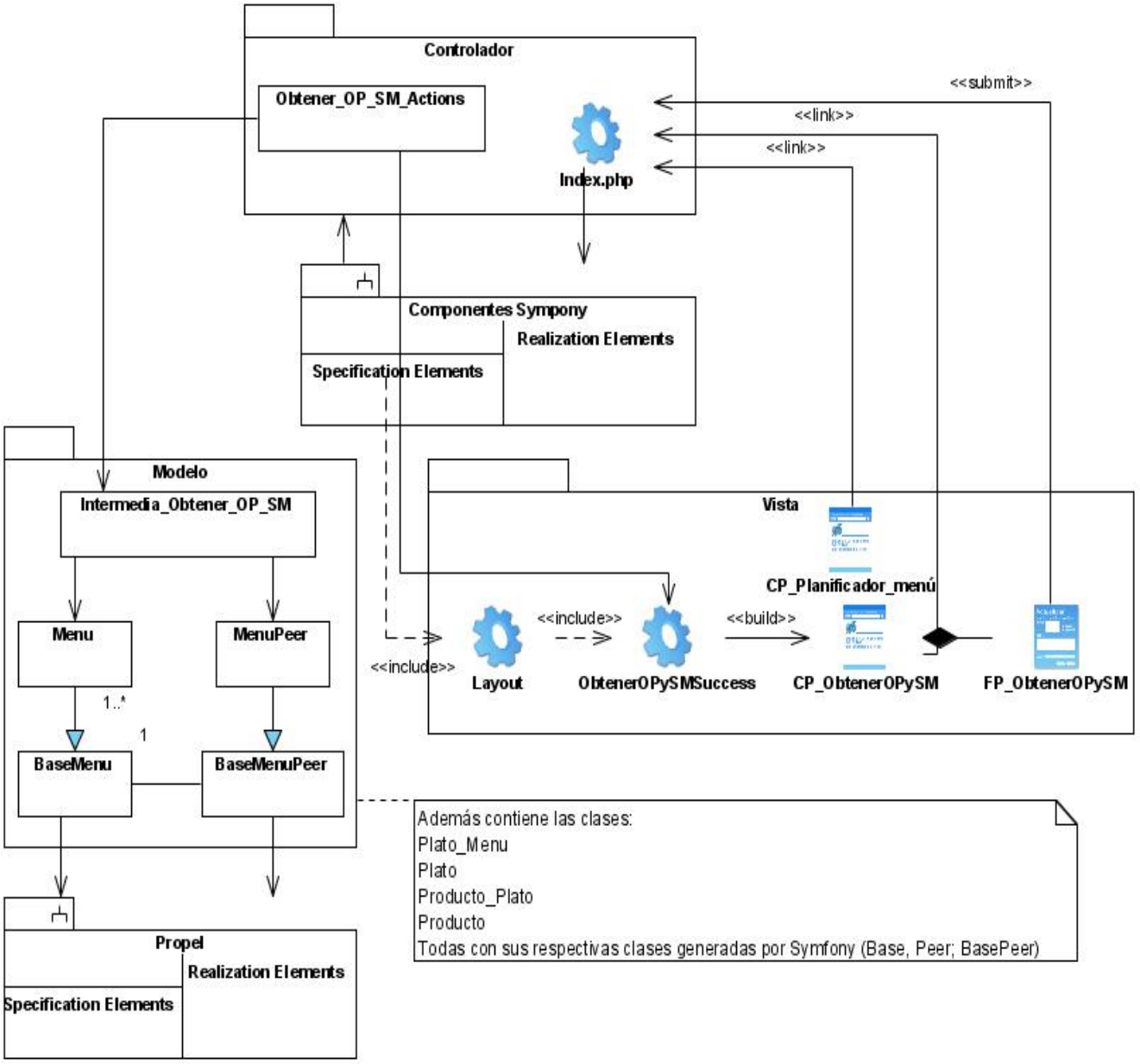

Figura 25 Diagrama de clases del diseño (Web) del CU Obtener Orden de Producción y Solicitud de Materiales.

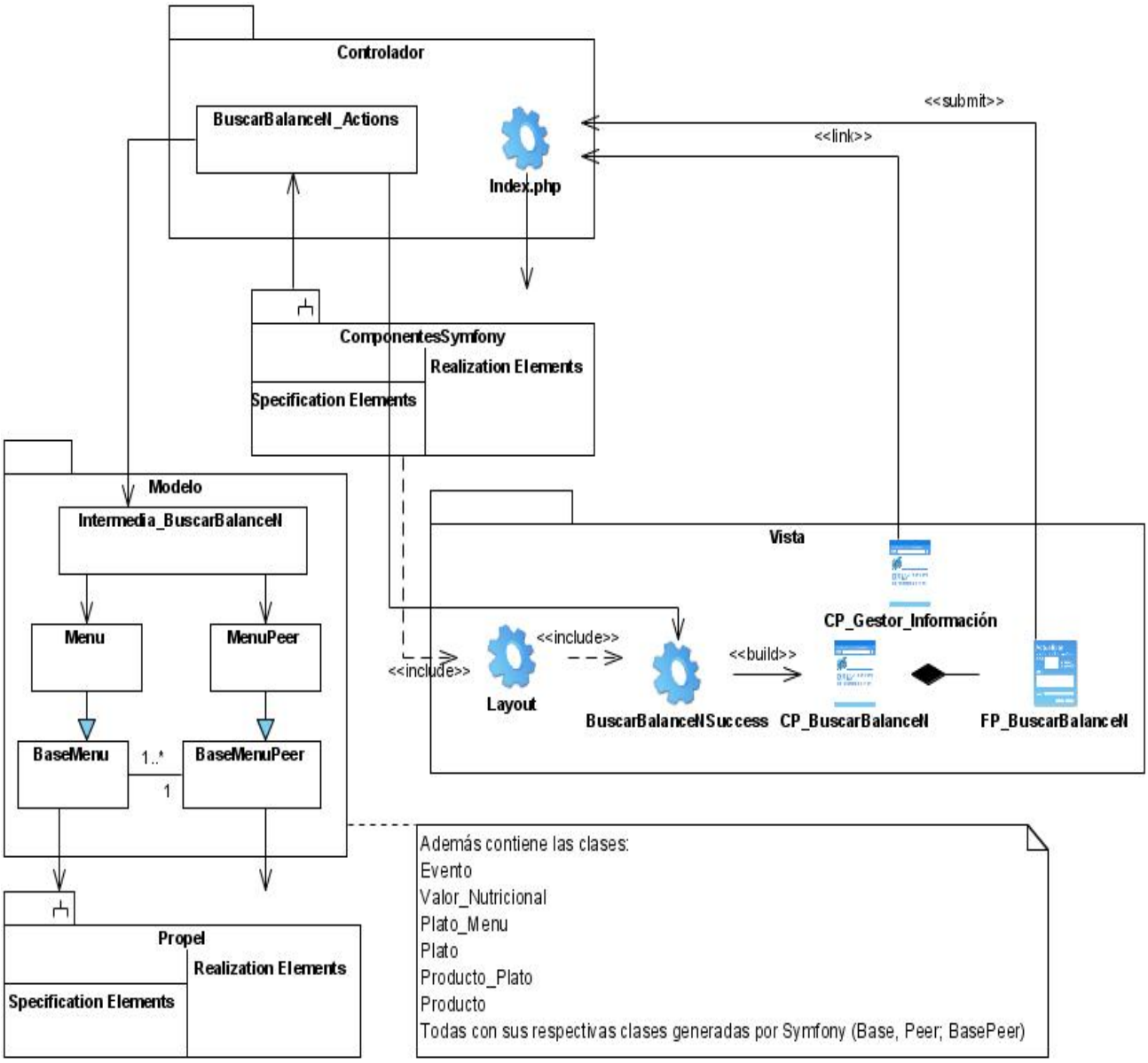

Figura 26 Diagrama de clases del diseño (Web) del CU Buscar balance Nutricional real y Planificado.

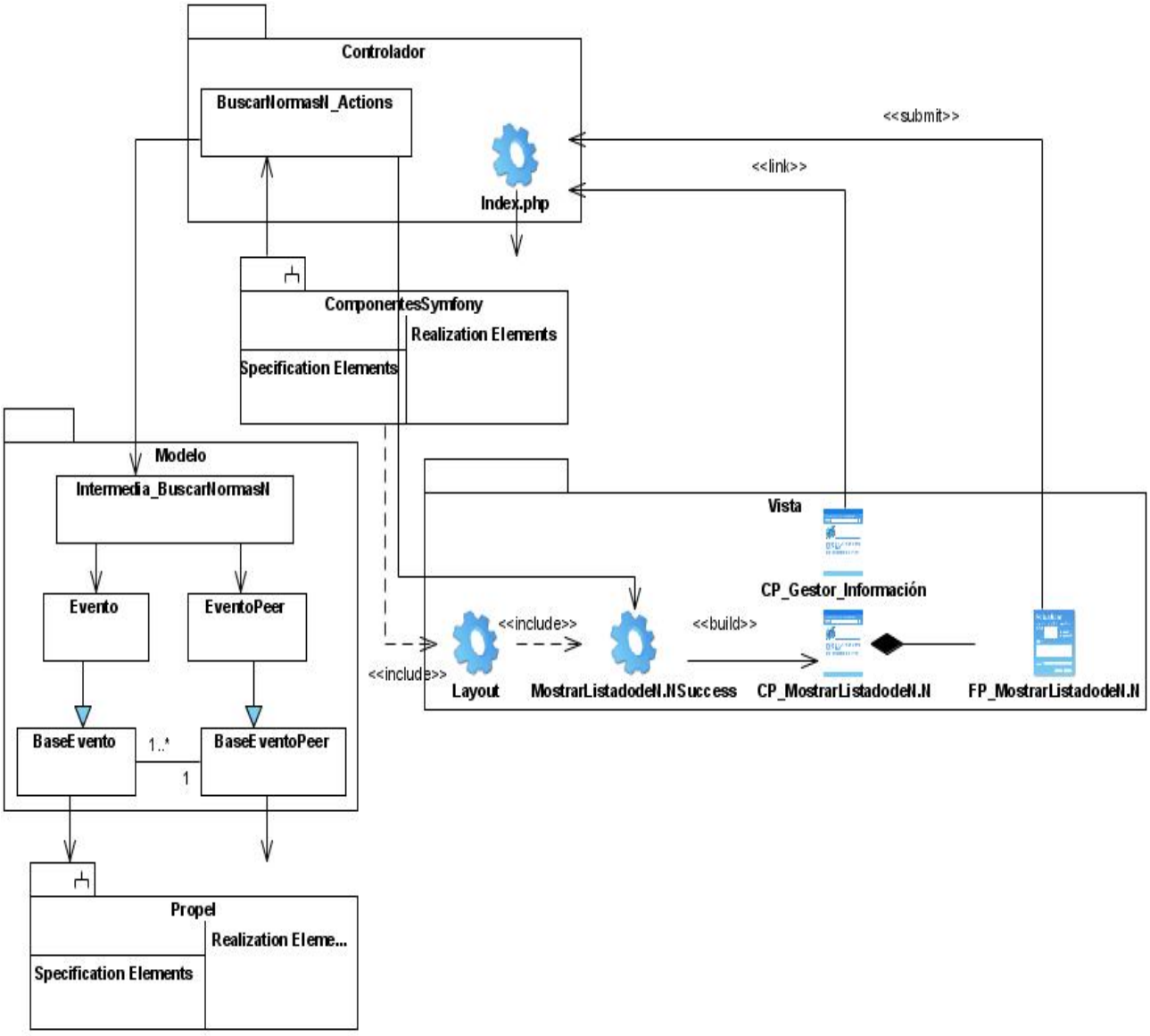

Figura 27 Diagrama de clases del diseño (Web) del CU Buscar Normas Nutricionales.

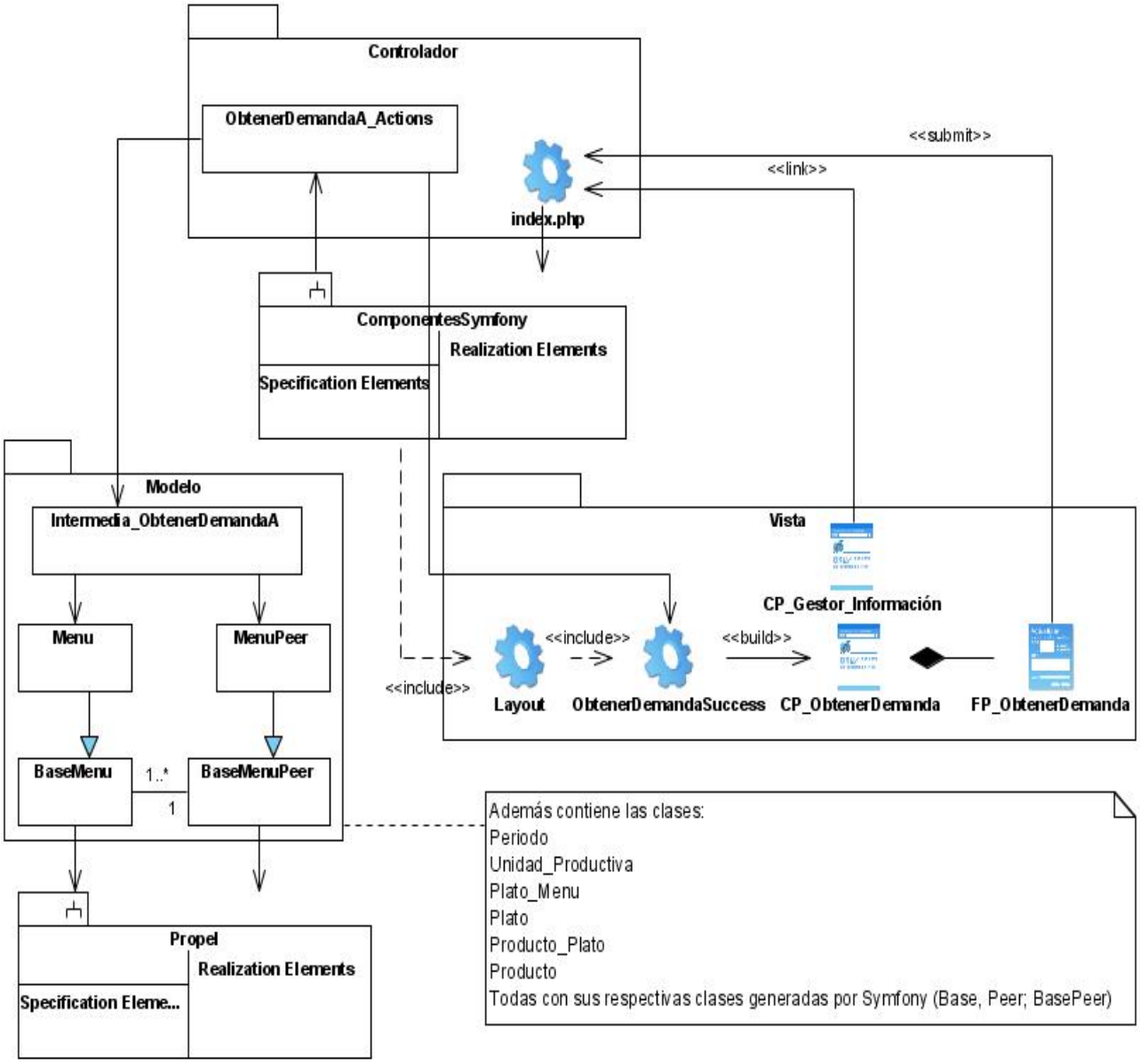

Figura 28 Diagrama de clases del diseño (Web) del CU Obtener Demanda de alimentos para un periodo.

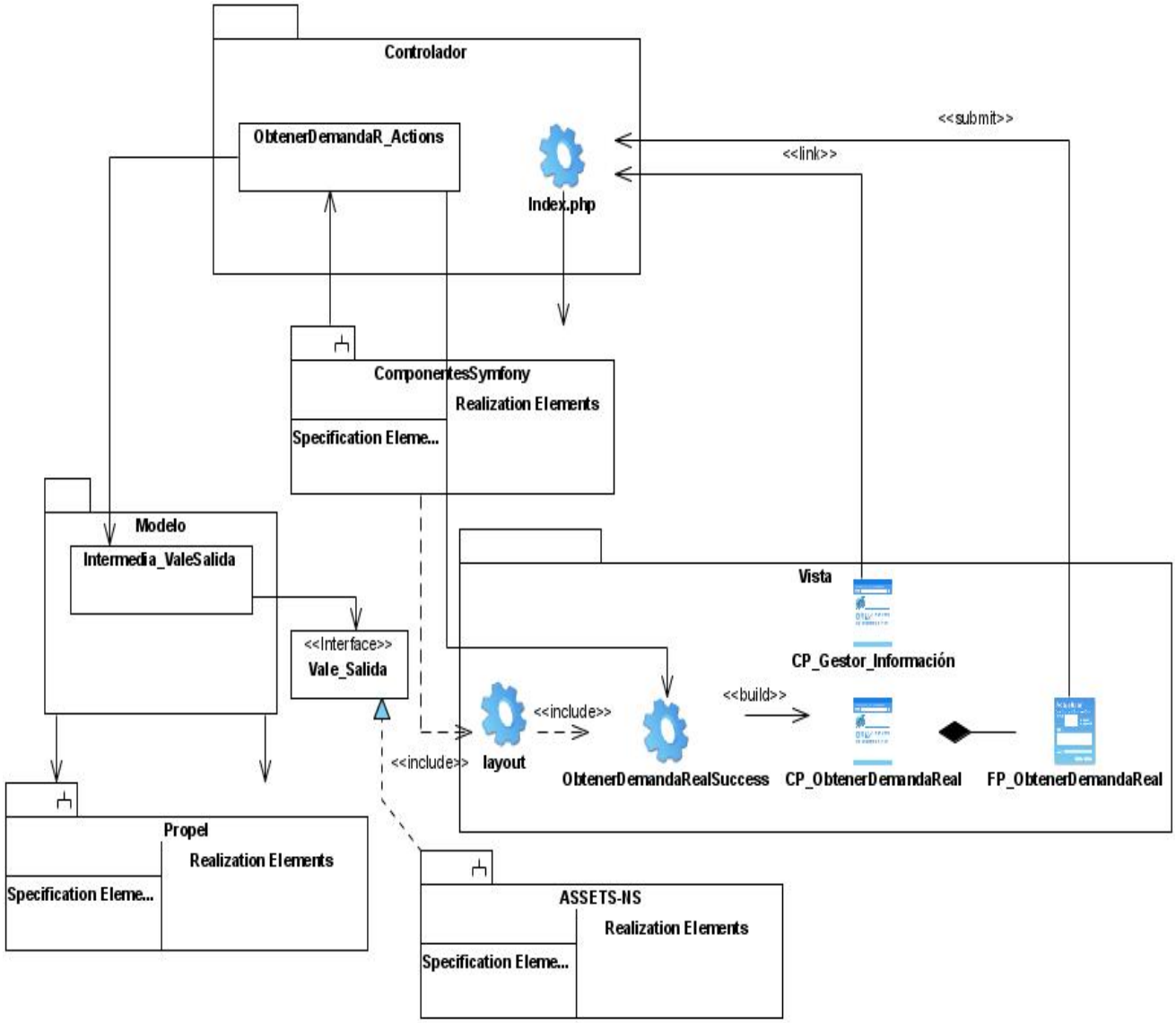

Figura 29 Diagrama de clases del diseño (Web) del CU Obtener Demanda Real de Alimentos.

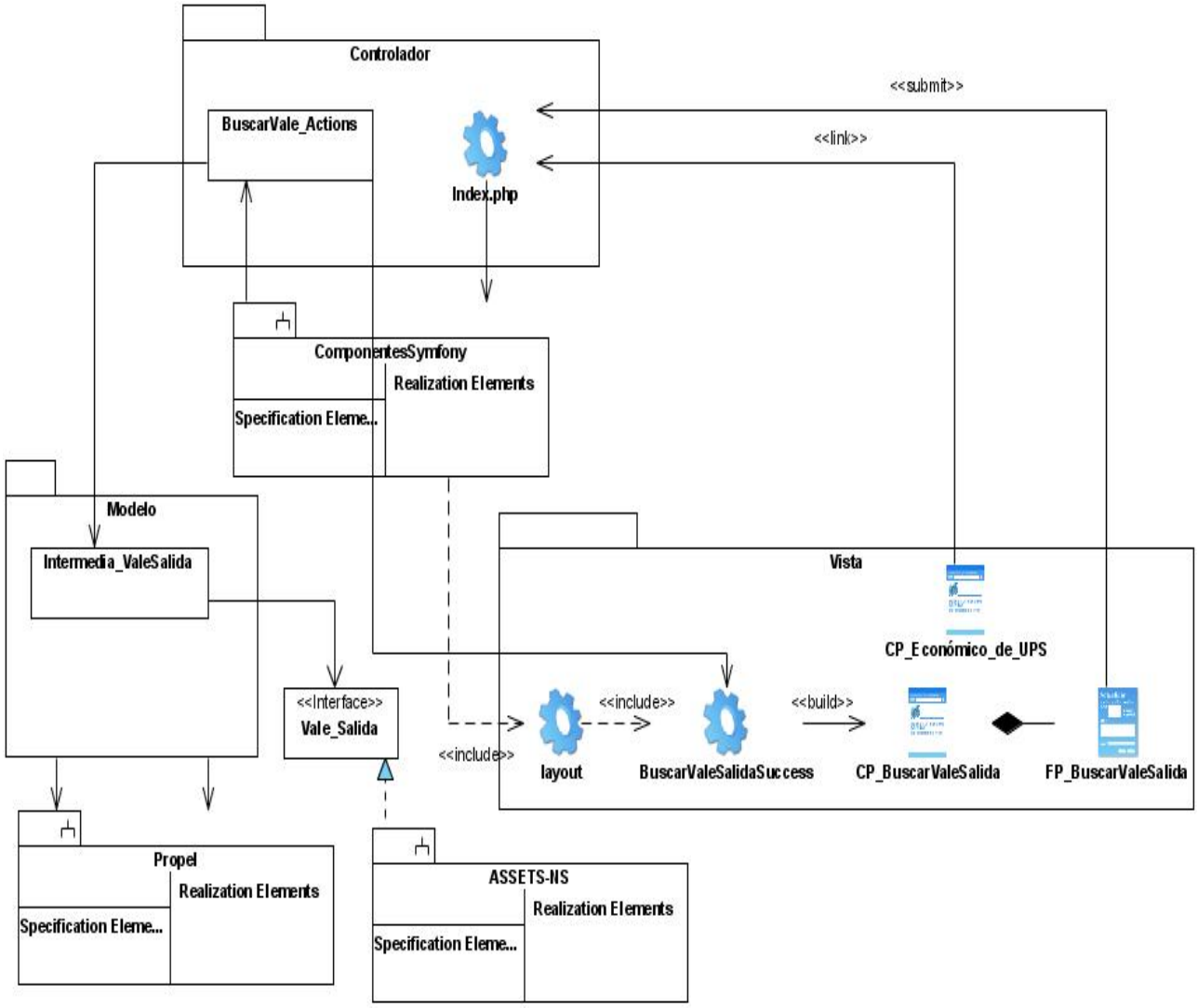

Figura 30 Diagrama de clases del diseño (Web) del CU Buscar vale de salida.

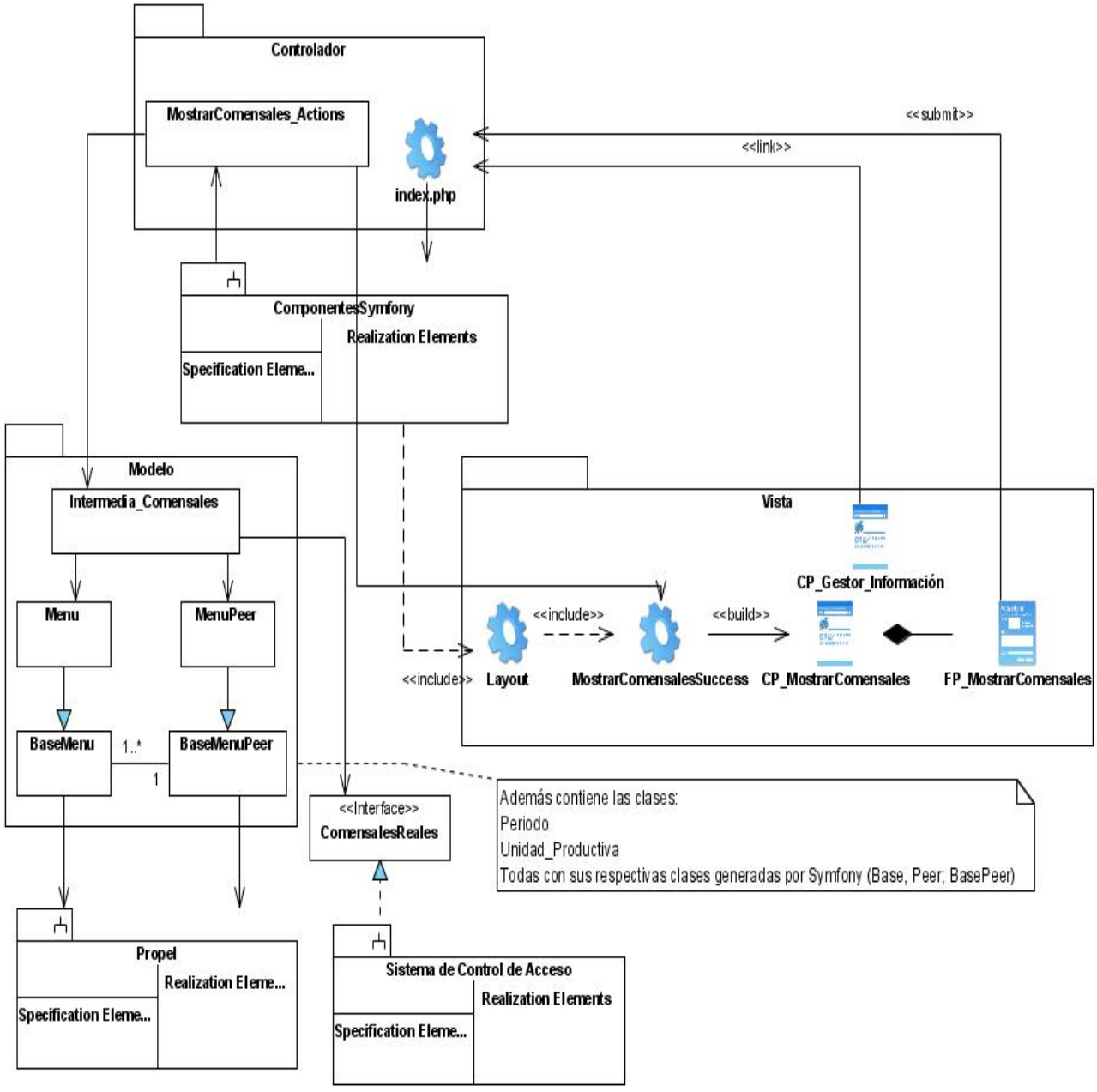

Figura 31 Diagrama de clases del diseño (Web) del CU Mostrar comensales planificados y reales.

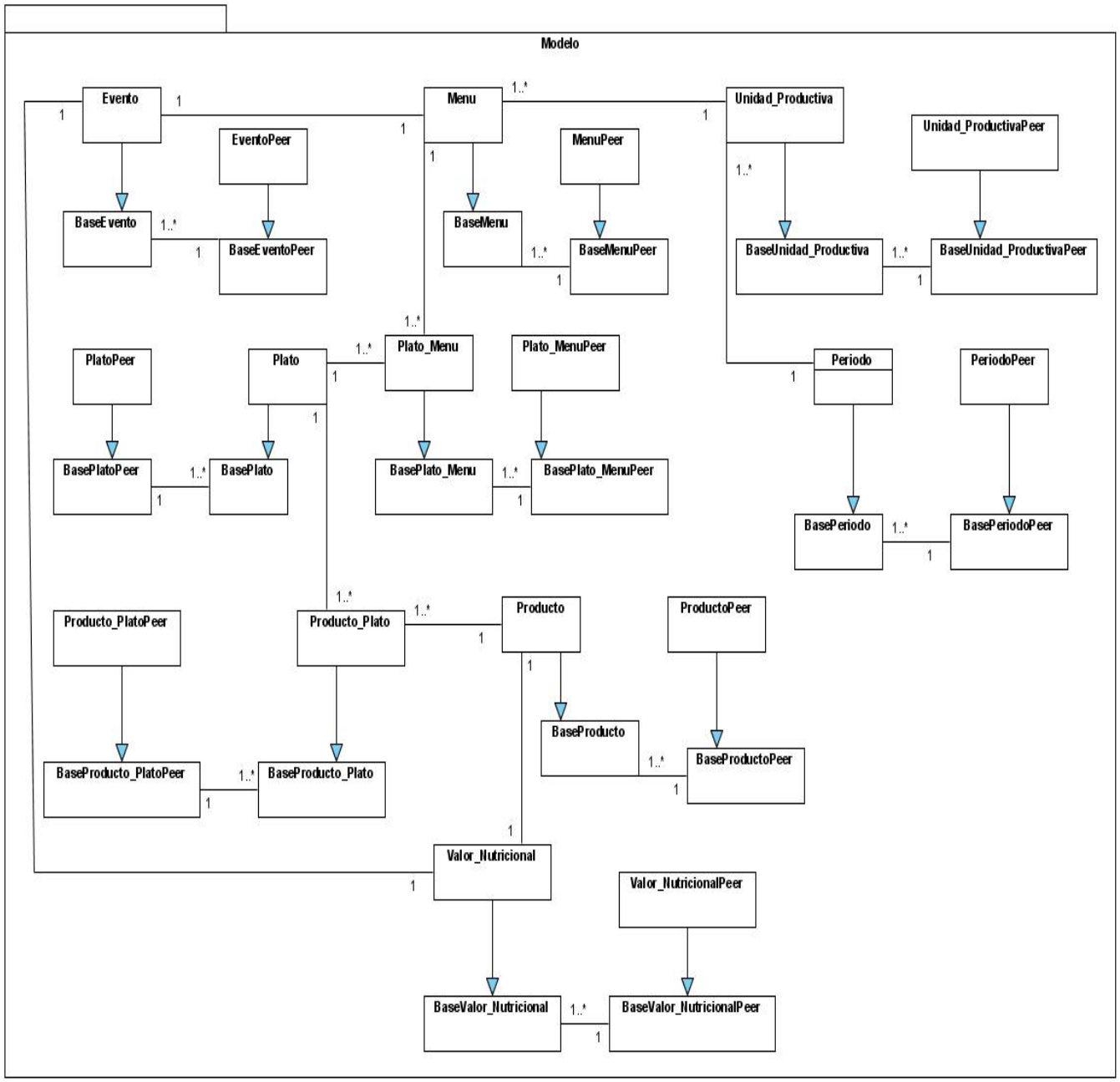

Figura 32 Paquete del modelo

3.5.4 Diagramas de Secuencia.

En los diagramas de secuencia para aplicaciones Web sin importar el enfoque empleado, es necesario considerar que hay mensajes que son usados simbólicamente para reflejar interacciones propias de la tecnología, tal es el caso de los vínculos, envíos de formulario, construcción de páginas, navegaciones y redireccionamientos (*link, submit, build navigate, redirect*). Como consecuencia de la interacción entre el actor y la Aplicación Web, muchas de esas interacciones se ejecutan entre los distintos elementos que la componen, su representación en un diagrama de secuencia contribuye a entender mejor su funcionamiento, colaborando con el objetivo de documentar la solución.

En este epígrafe se pondrán los diagramas de secuencia de los casos de uso del sistema, los cuales ilustran el funcionamiento que tendrá el sistema ante alguna petición del actor.

En el caso de Obtener orden de producción y solicitud de materiales, por ejemplo, la línea es la siguiente:

El actor solicita en la página principal, la primera que se muestra luego de haberse autenticado, obtener la orden de producción y solicitud de materiales para el menú de un día determinado, como resultado dicha página hace un link al Index.php y este construye la página cliente (CP_ObtenerOPySM) donde el actor entra la fecha para la cual desea llevar a cabo dicha acción.

Esta página cliente le hace un envío con el nombre del módulo (OrdenSolicitud) y la acción (Obtener OPySM) a la Index.php, la cual envía esos datos al sfController, componente de Symfony, el cual se encarga de verificar la seguridad del sistema, verificar el formulario y de decirle a la clase Actions cual es la acción que debe ejecutar (executeObtenerOPySM()). La acción solicitada hace una llamada al método listarMenusConfirmados() de la clase intermedia, encargada de la lógica del negocio, la misma es la responsable de buscar en la base de datos la información que se solicita, en este caso un listado con los menues confirmados que existen, le pasa este listado a la Actions, quien seguidamente le envía este listado a la ObtenerOPySMSuccess para que construya la página cliente (CP_ObtenerOPySM) y esta mostrará en un *Select* una lista de los menues confirmados para la fecha indicada.

El actor selecciona el menú en la página cliente para el cual ejecutar la acción y esta le hace un envío nuevamente al Index.php enviándole el nombre del módulo (OrdenSolicitud) y la acción (Obtener

OPySM) a ejecutar, este le envía al sfContoller los datos, el cual se encarga de indicarle a la clase Actions la acción que debe ejecutar (executeObtenerOPySM()). Esta acción hace una llamada al método obtenerOPySM() de la clase intermedia la cual busca en la base de datos la información necesaria, en este caso los datos para elaborar los documentos orden de producción y solicitud de materiales, le manda estos datos a la clase Actions, quien posteriormente le envía todos estos datos a la página ObtenerOPySMSuccess la cual se encargará de construir la página cliente con el resultado final de la acción solicitada.

Esta interacción tiene lugar cada vez que se hace una solicitud al sistema, es decir, luego de haberse seleccionado la opción que se quiera realizar, que el controlador frontal recibe la petición, se verifica la seguridad y si el usuario tiene los permisos para acceder a la información solicitada se ejecutan una serie de acciones, en las cuales la diferencia de los mensajes se debe las distintas peticiones con las que se debe crear la página cliente, cuando es necesario buscar en la base de datos y cuando no se requieren datos de la misma. Es por ello que se plantea que una vez que se identifica el mecanismo a emplear, se debe documentar una sola vez y de forma general, pues repetir lo mismo en cada diagrama resultaría redundante y cargaría el diagrama con mensajes y objetos que siempre serán los mismos.

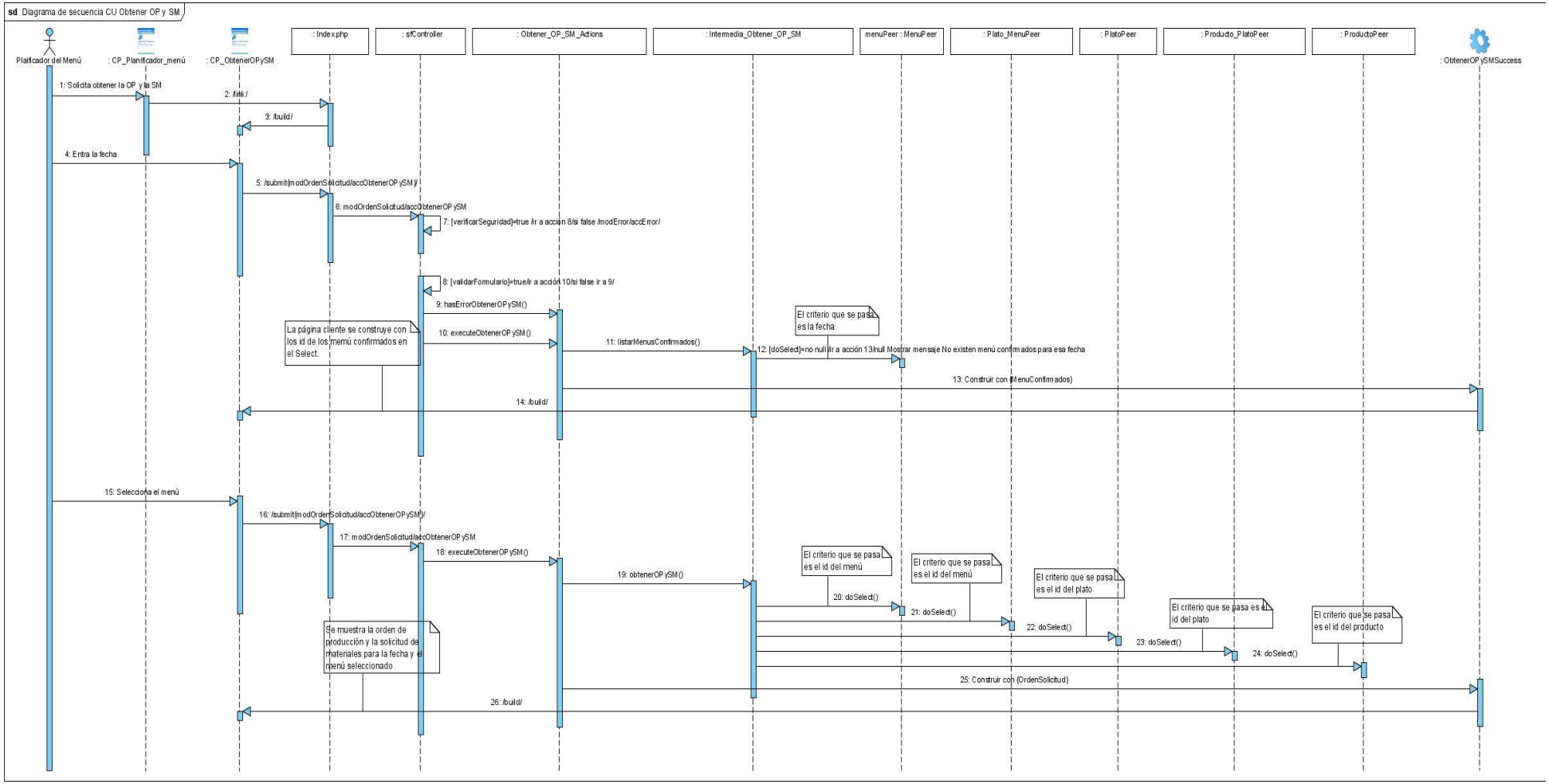

Figura 33 Diagrama de secuencia del CU Obtener Orden de Producción y Solicitud de Materiales.

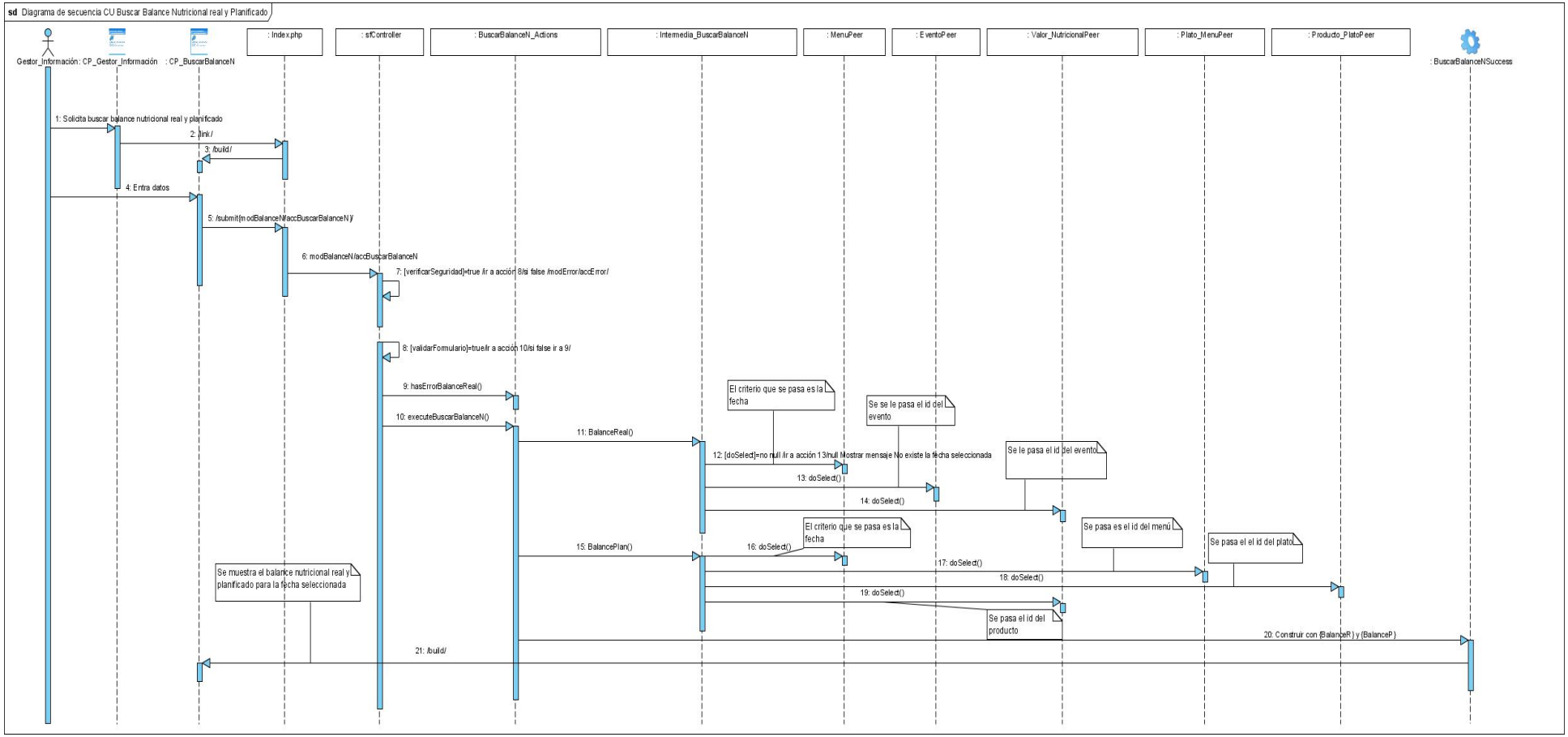

Figura 34 Diagrama de secuencia del CU Buscar Balance Nutricional real y planificado.

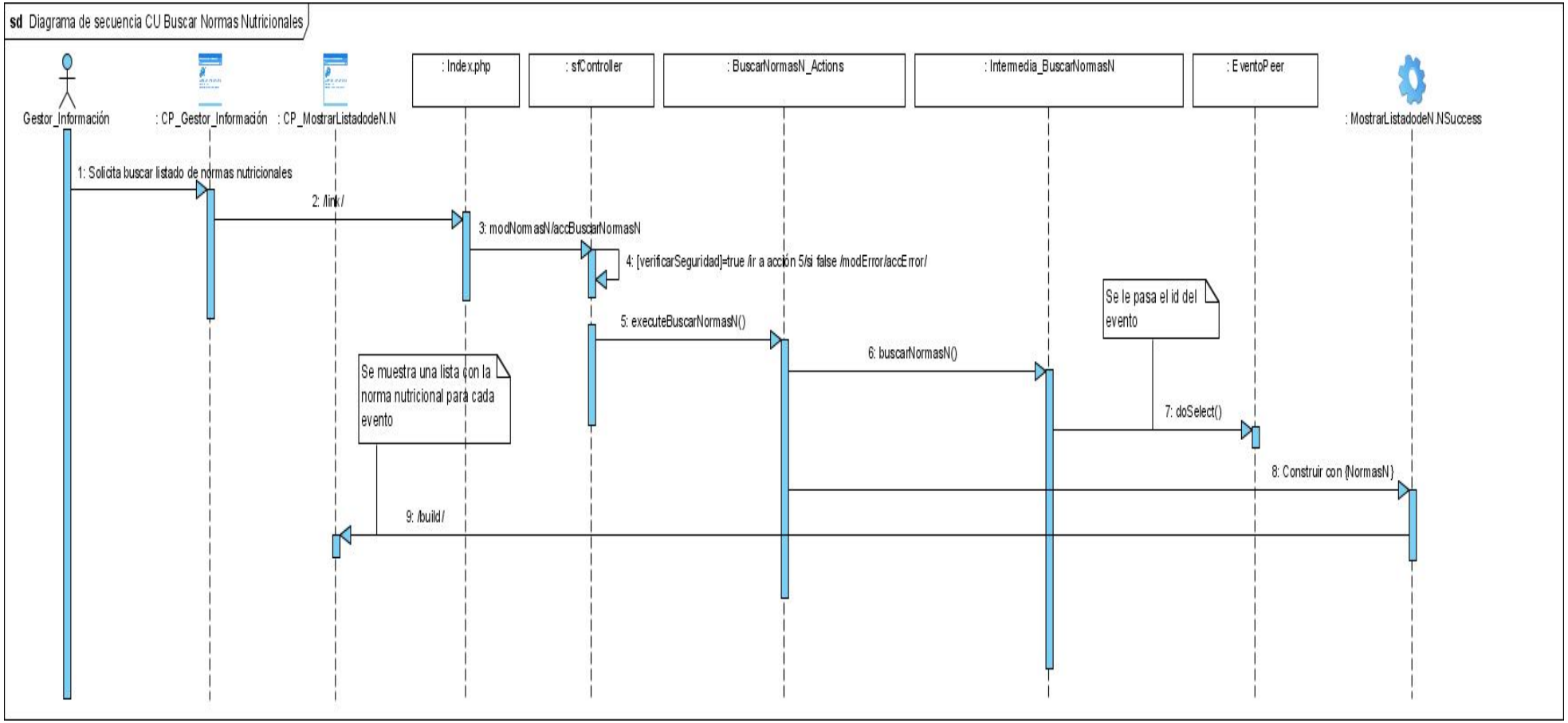

Figura 35 Diagrama de secuencia del CU Buscar Normas Nutricionales.

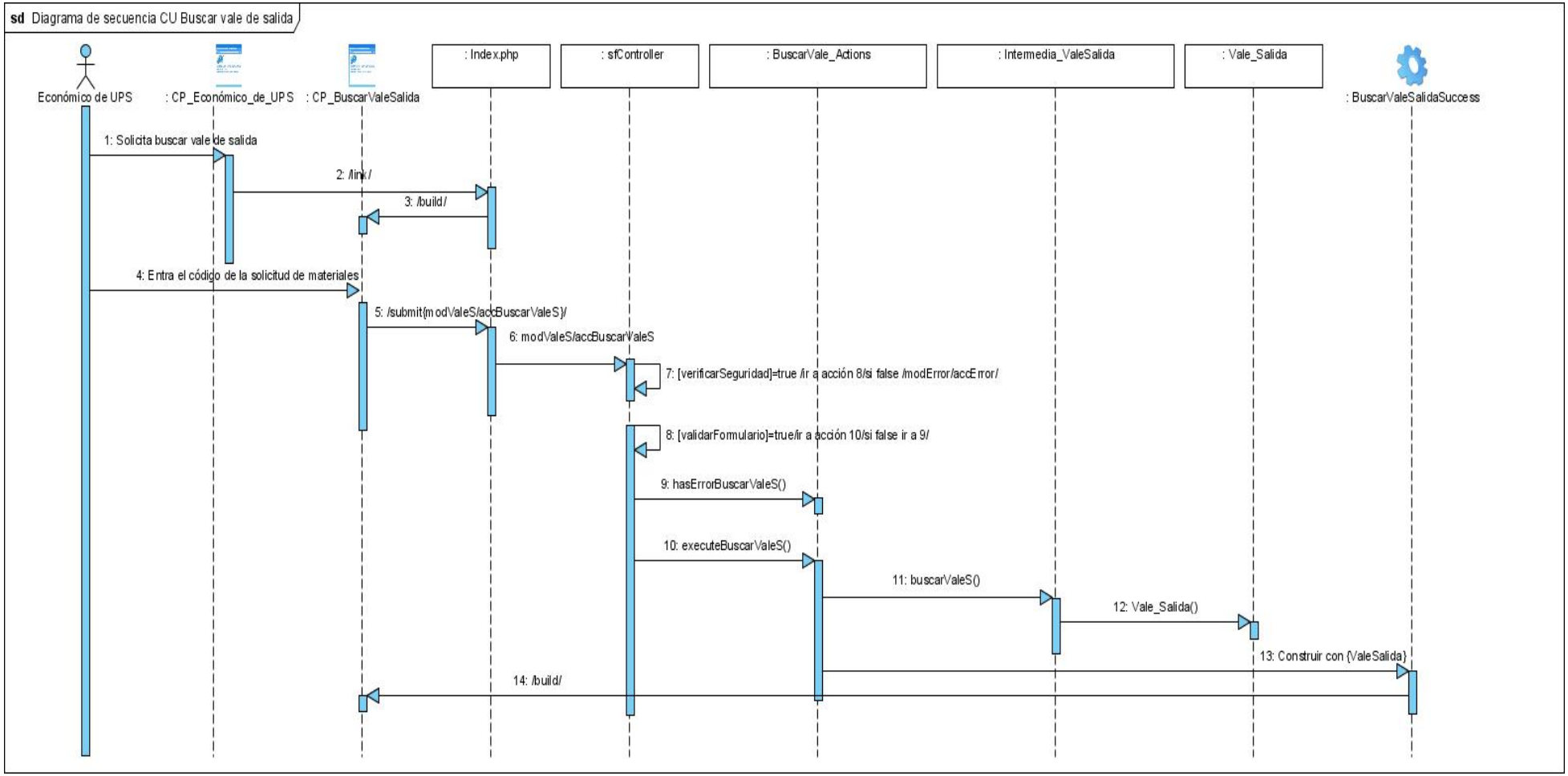

Figura 36 Diagrama de secuencia del CU Buscar vale de salida.

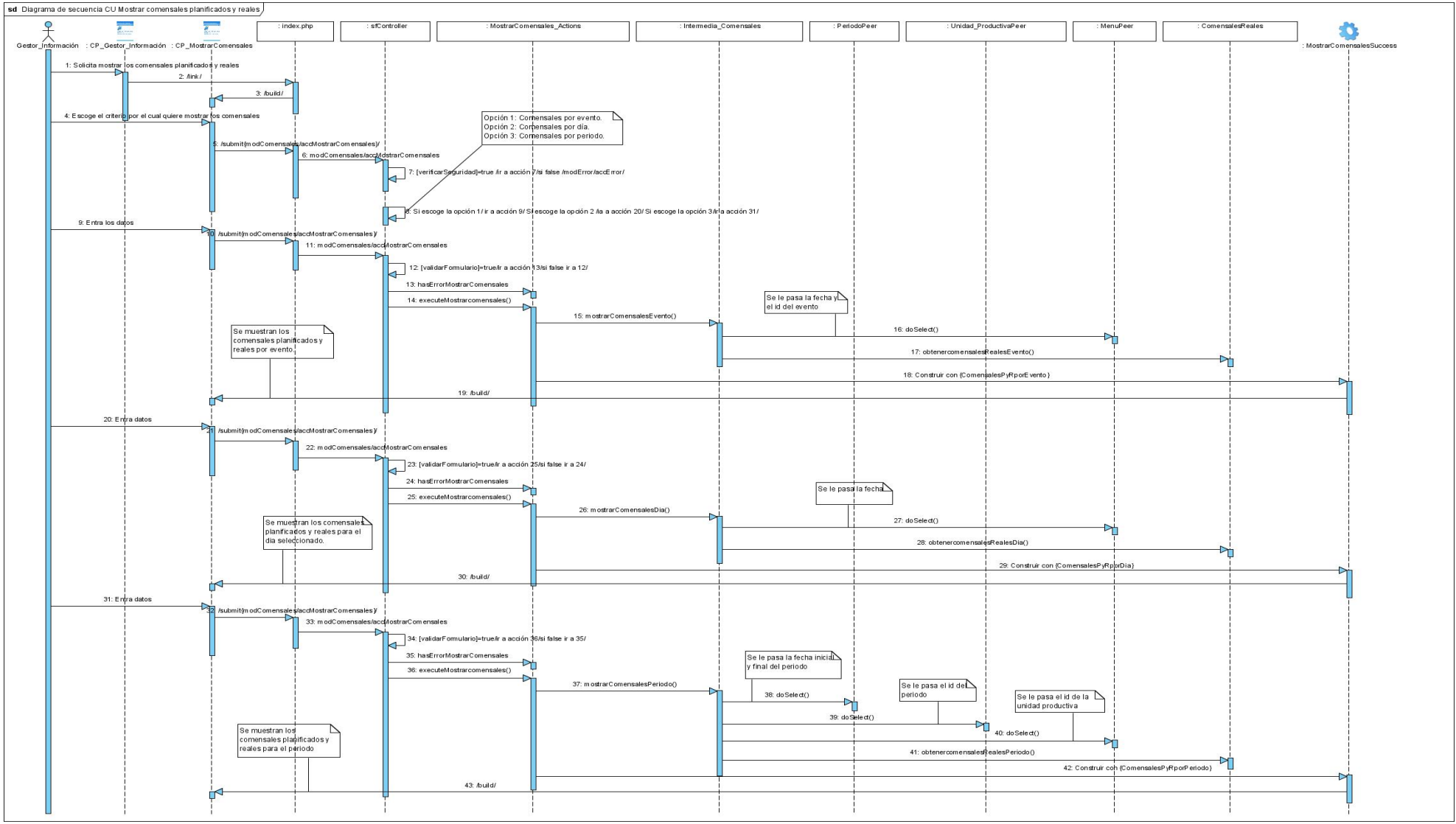

Figura 37 Diagrama de secuencia del CU Mostrar comensales planificados y reales.

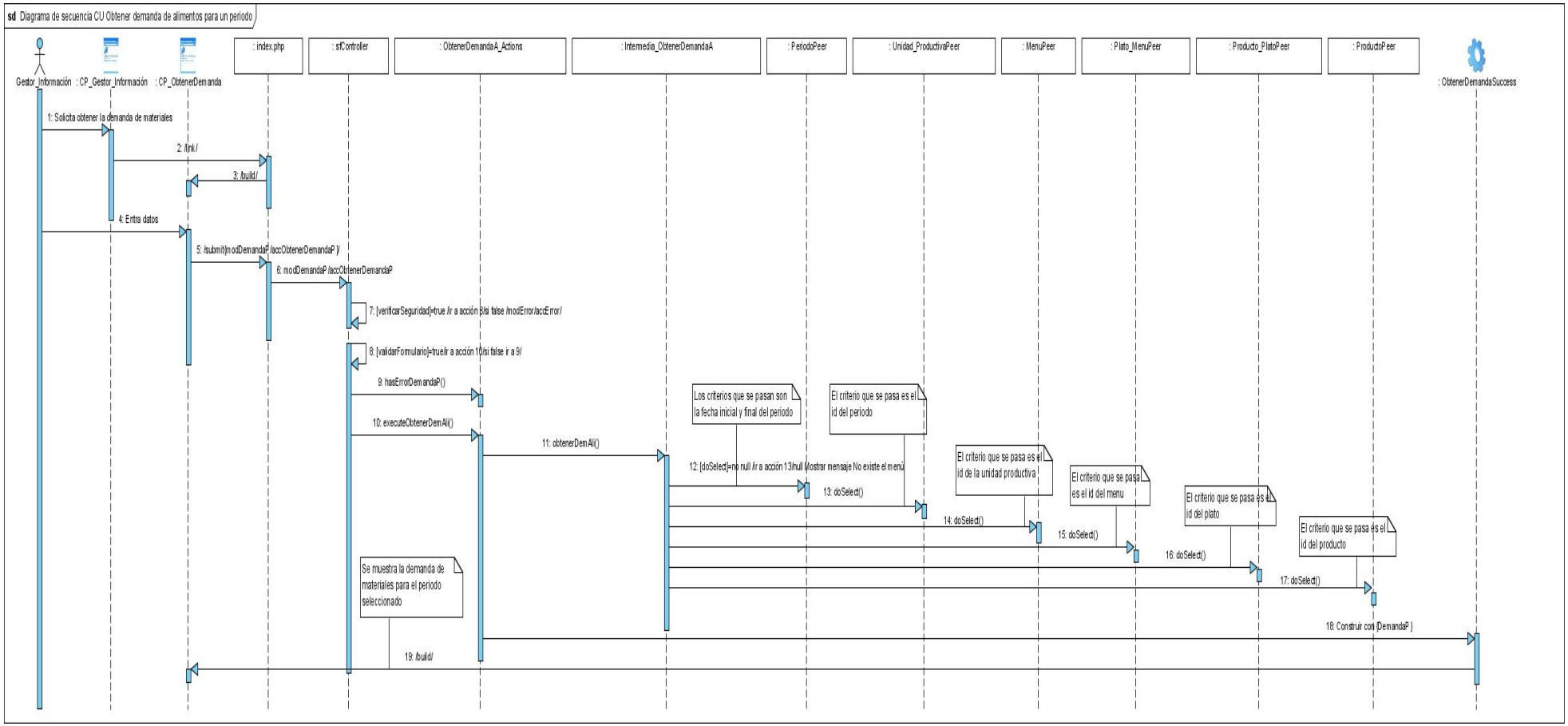

Figura 38 Diagrama de secuencia del CU Obtener demanda de alimentos para un periodo.

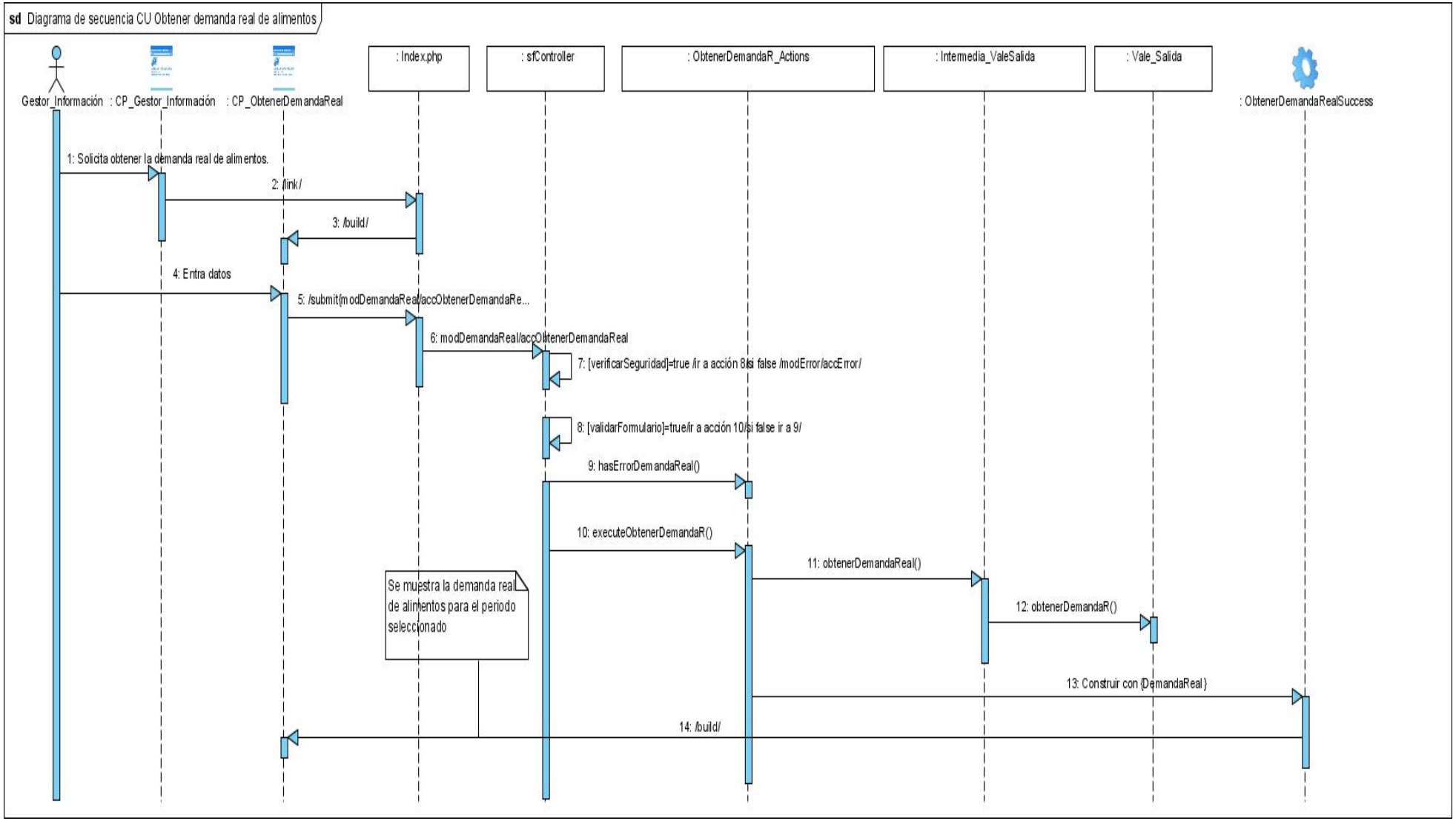

Figura 39 Diagrama de secuencia del CU Obtener demanda real de alimentos

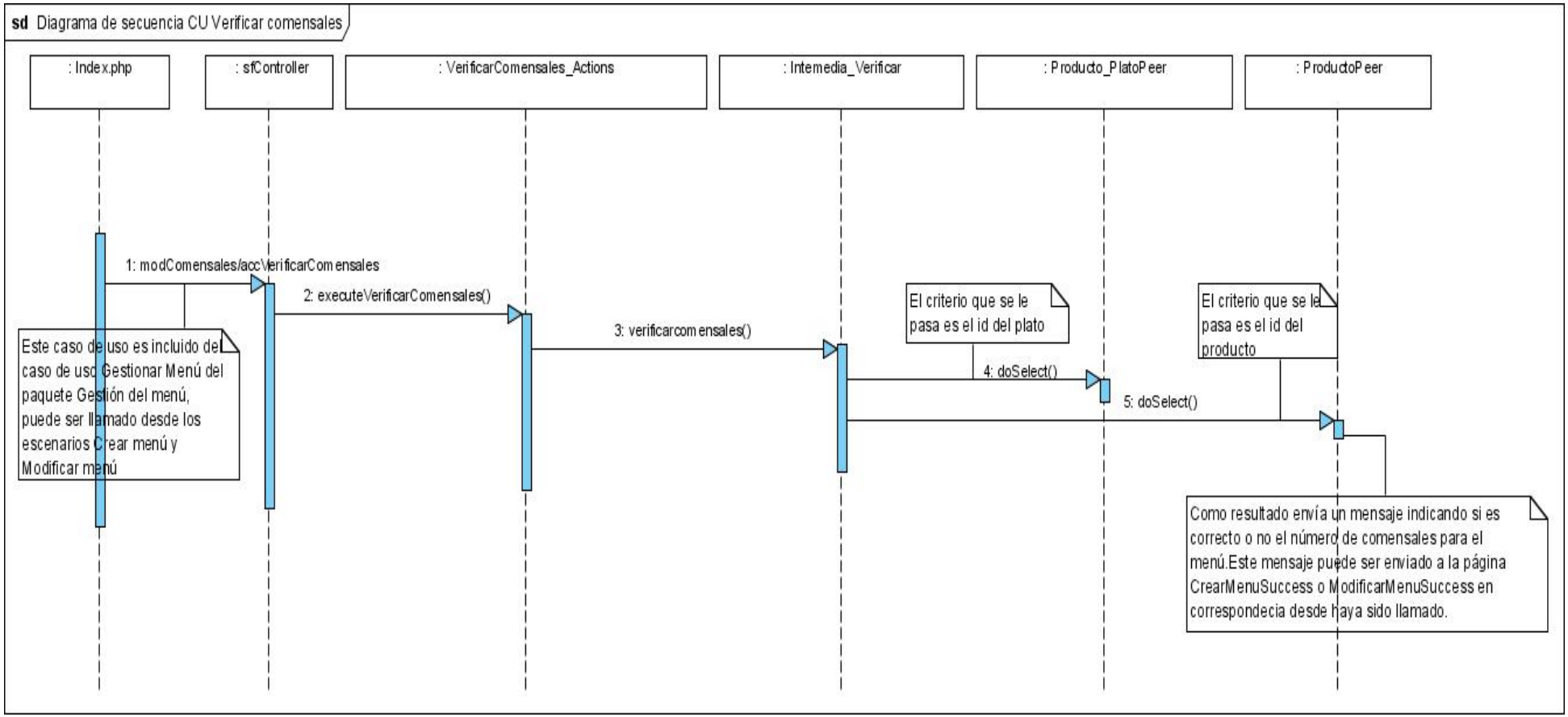

Figura 40 Diagrama de secuencia del CU Verificar comensales

Para ver los restantes diagramas de secuencia remitirse al expediente de proyecto.

3.6 Validación del diseño realizado

Existen varias técnicas para validar el diseño realizado, entre ellas tenemos:

- **Evaluación de la calidad de las clases diseñadas a través de la generación del código con la herramienta CASE de modelado:** este método no puede ser utilizado debido a las características que posee el diagrama de clases del diseño realizado, pues en este se tiene un paquete genérico (Modelo) para evidenciar las relaciones entre las diferentes clases del mismo, por lo que el código que se generaría no sería de mucha utilidad para los programadores del proyecto.

- **Realizar una comparación entre lo implementado y lo modelado**: este procedimiento no puede ser aplicado debido a que en proyecto aún no se ha comenzado con la implementación.

- **Realizar encuestas y/o entrevistas a los programadores con el objetivo de conocer el grado de satisfacción que los mismos adquirieron como resultado de un análisis del diseño realizado**: Este método es uno de los más conocidos y utilizados, siendo este precisamente el utilizado por los analistas del proyecto Alimentos UCI.

  La encuesta aplicada a los cuatro programadores consta de 7 aspectos, arrojando los siguientes resultados:

  Los prototipos no funcionales abarcan todas las funcionalidades identificadas para la aplicación, de igual manera existe correspondencia con los requerimientos no funcionales al ser las interfaces sencillas, con colores claros, sin sobrecarga de imágenes, además cada usuario que se autentifique en el sistema tendrá acceso solamente a la página que le corresponde.

  Todos los programadores han estudiado los diagramas de clases del diseño y secuencia realizados, entendiendo el flujo de mensajes que se representa en los mismos, considerándolos además legibles y apropiados para comenzar con la implementación del sistema.

3.7 Consideraciones del capítulo.

En este capítulo se definieron las clases que contendrán las capas de presentación, negocio y datos, las primeras se encargarán de la comunicación cliente-sistema, las de negocio que manipularán la información y las de la capa de datos que brindan la información requerida. Se realizaron los diagramas de clases correspondientes a los procesos fundamentales del sistema y los diagramas de interacción donde se modelaron los aspectos dinámicos de la aplicación.

Conclusiones

Como resultado de esta investigación se puede concluir que:

Se analizaron los procesos que se llevan a cabo en la gestión de comensales, llegando a un mayor grado de comprensión de las actividades que se desarrollan y que serían posible objeto de automatización dando paso a obtener los artefactos correspondientes al modelado del negocio, los cuales contribuyeron a lograr una mejor comunicación con el cliente.

Se llevó a cabo el estudio de las funcionalidades que debe cumplir el módulo Gestión de Comensales identificando los requisitos funcionales y no funcionales del sistema, efectuando también la descripción detallada de los casos de uso.

Se logró que los artefactos obtenidos en la etapa de análisis y diseño se ajustaran a los principios y patrones del diseño orientado a objetos, proporcionando además la comunicación entre los analistas y el equipo de desarrollo.

Se validó el diseño de las clases necesarias para la implementación a través de una encuesta realizada a los programadores del proyecto.

Recomendaciones

- Refinar el diseño propuesto para la implementación.
- Llevar a cabo la implementación del módulo Gestión de comensales, así como su integración al módulo Gestión del Menú, debido a la gran relación existente entre ellos ya que la mayoría de los procesos que se llevan a cabo en la gestión de comensales dependen de los que se realizan en la gestión del menú.

Referencias bibliográficas

(1). **Zavala.** angelfire. *angelfire.* [En línea] 08 de 09 de 2002. [Citado el: 23 de 12 de 2007.] http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html.

(2). ***Sanchez, María A. Mendoza*.** Informatizate. *Informatizate.* [En línea] 7 de 06 de 2004. [Citado el: 07 de 12 de 2007.] (Mendoza, 2004)

(3). Teleformación. *Entorno Virtual de Aprendizaje.* [En línea] 16 de 09 de 2007 [Citado el: 07 de 12 de 2007.] http://teleformacion.uci.cu/mod/resource/view.php?id=6653

(4). Free Download Manager - Sitio de descargas de software [En Línea] [Citado el 10 de 12 de 2007] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/

(5) **ATOS, I. T**. (04 de 03 de 2008). *INES.* [Citado el 31 de 03 de 2008.] de INES: http://www.ines.org.es/vulcano/wp-content/uploads/2007/09/d6-estudio-de-herramientas-de-certificacion-tid.pdf

(6). Sparx Systems*. Sparx Systems*. [En línea] [Citado el: 01 de 04 del 2008.] http://www.sparxsystems.com.ar/products/ea.html.

(7) *Sistemas Phoenix S*. *de R.L.* [En línea] [Citado el: 01 de 04 de 2008.] http://www.campusoft.com.mx/Studio%20MX%202004.htm

(8). *VitaminaWEB.com*. [En línea] [Citado el: 01 de 04 de 2008.] http://www.vitaminaweb.com/html/articulos/articulo02.php

(9). Oktaba, Hanna. Introducción a Patrones. *Introducción a Patrones.* [En línea] [Citado el: 24 de 04 de 2008.] http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html.

(10). Zaninotto, Fabien Potencier y François*. Symphony la guía definitiva*. s.l. : Apress (ISBN-13: 978-1590597866), 2005.

(11). Larman, C. *UML y Patrones*, Tomo I, Capítulo 18.

(12). *Librosweb.es*[En línea] [Citado el: 20 de 04 de 2008] http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html

(13). SVS NEWSLETTER. *Sociedad de Validación de Sistemas S.L*. [En línea] SVS S.L, 05 de 2006. [Citado el: 17 de 04 de 2008.] http://www.svshome.com/newsletter/boletin03.html.

(14). Teleformación. *Entorno Virtual de Aprendizaje*. [En línea] (Teleformación, 2008) 28 del 02 del 2008. [Citado el: 26 de 05 de 2008.] http://teleformacion.uci.cu/mod/resource/view.php?id=21363

(15). **Torres, Sandry Hernández.** *Presentación en el forum Sandry, SISC UCI.*

Bibliografía Consultada

- Larman, *C. UML y Patrones*. En C. Larman, UML y Patrones (págs. Tomo I, Capitulo 18).
- Teleformación. *Entorno Virtual de Aprendizaje*. [En línea] 16 de 09 de 2007.
- Zaninotto, *F. P. (2005). Symfony la guía definitiva. s1*.: Apress (ISBN-13:978-1590597866).
- Febles, *Y. E. A. C. y Y. R. (2005).* Sistema Informático para la Gestión Integral de Comedores. Ciudad Habana, CUJAE. (Zavala, 2002)
- Sparx Systems. *Sparx Systems*. [En línea] [Recuperado el: 01 de 04 de 2008.] http://www.sparxsystems.com.ar/products/ea.html
- Sistemas Phoenix S. de R.L. [En línea] http://www.campusoft.com.mx/Studio%20MX%202004.htm
- *VitaminaWEB.com*. [En línea] http://www.vitaminaweb.com/html/articulos/articulo02.php
- Alvarez, Miguel Angel. desarrolloweb. *desarrolloweb.* [En línea] http://www.desarrolloweb.com/articulos/332.php.

Anexos

Anexo 1

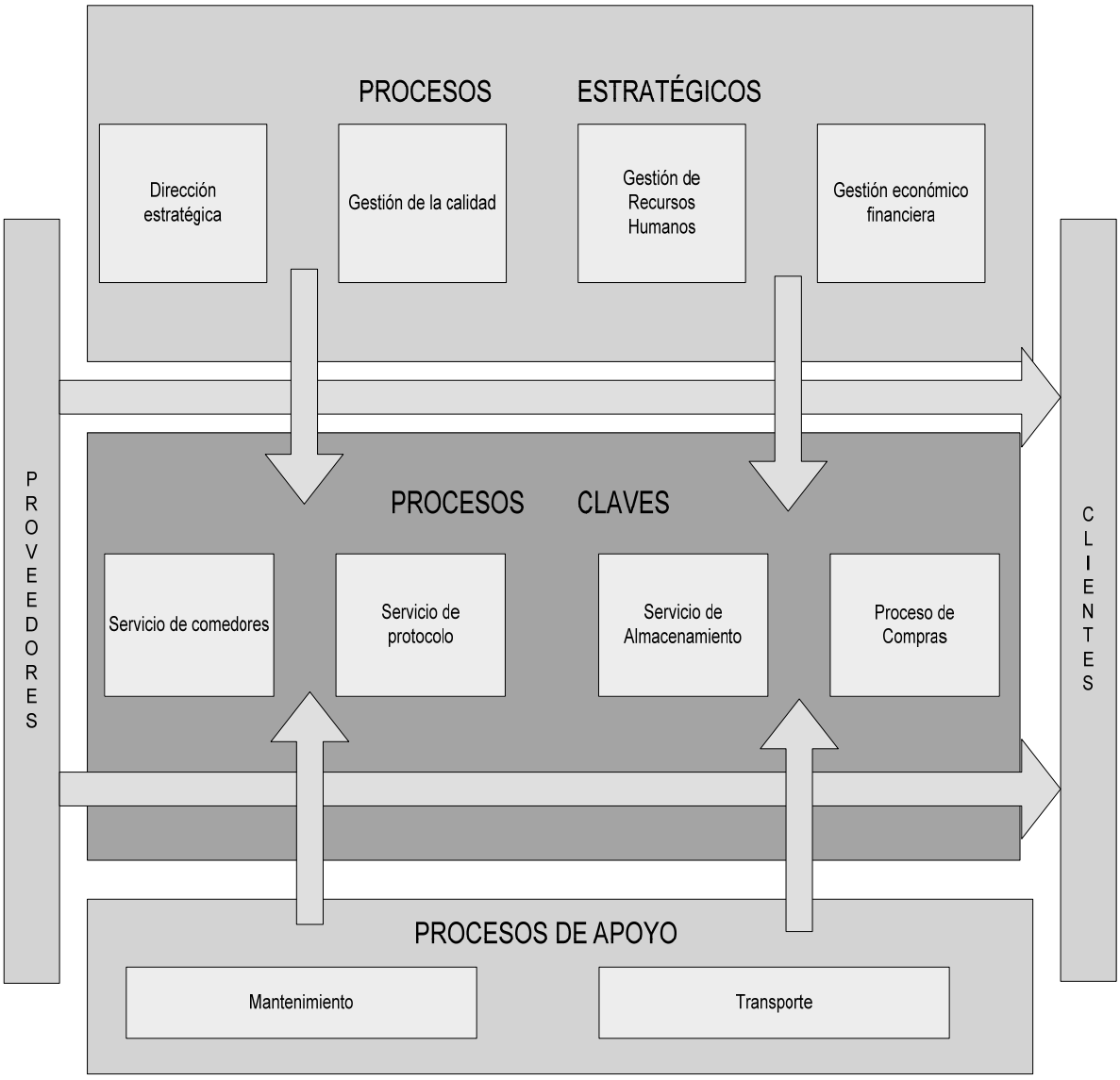

Figura 41 Mapa de procesos logísticos (15).

Anexo 2

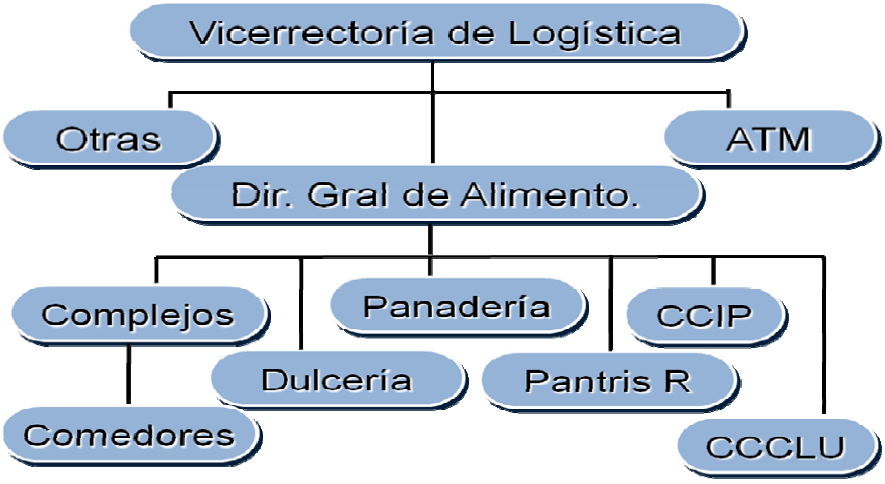

Figura 42 Organigrama de la Vicerrectoría de Logística

Anexo 3

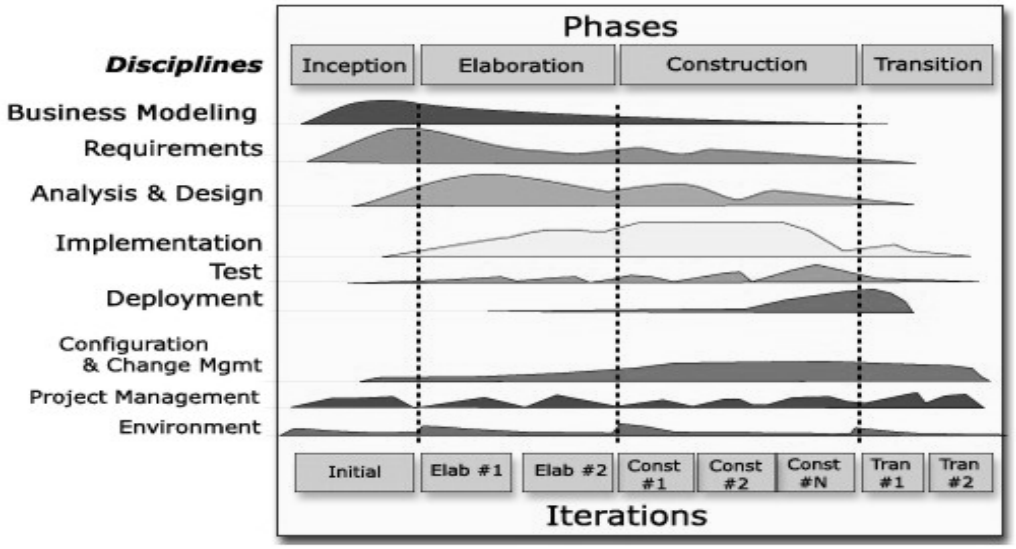

Figura 43 Gráfica de RUP

Anexo 4

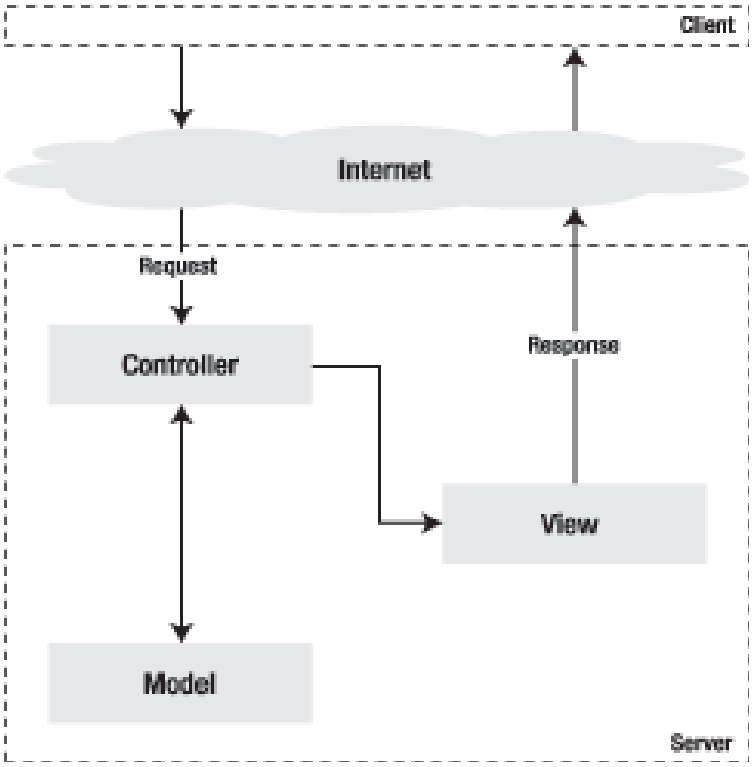

Figura 44 Patrón Modelo Vista Controlador

Glosario de términos

A

Arborescencia: ilustración de la web planeada donde la unidad básica de información responde a cada página que el usuario encontrará en su recorrido.

ASP: *Avtive Server Pages*.

ATM: Abastecimiento Técnico Material.

B

Balance Nutricional: Permite conocer el comportamiento de los indicadores nutricionales para el comedor en un período, se muestra el porciento de cumplimiento del balance respecto a las normas establecidas. Esta información, se especifica por los indicadores nutricionales establecidos (carbohidratos, energía, grasas y proteínas).

C

CASE: *Computer Aided Software Engineering*.

C++: lenguaje de programación orientado a objetos, derivado de C, creado por **Bjarne Stroustrup.**

CSS: lenguaje de hojas de estilo, la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

CVS: Sistema Concurrente de Versiones**.**

Comensal: Representa a una persona que recibe algún servicio de alimentación del comedor

Comedor: Representa una instalación donde se ofertan los servicios de alimentación.

D

Decorator: decorador, patrón de diseño, agrega funcionalidad de adorno a un objeto.

Delphi: Borland Delphi, producido comercialmete por la empresa *Borland Software Corporation*, es un lenguaje de programación y un entorno de desarrollo rápido de software diseñado para la programación de propósito general con énfasis en la programación.

Demanda de alimentos: Es la cantidad de alimentos que se necesitan para realizar un servicio determinado, en un comedor, en un período y según el criterio determinado (planificación o real).

E

Energía: energía que posen los alimentos, se mide en términos de caloría o kilocaloría.

Evento: Etapa del día, entiéndase desayuno, almuerzo, comida y merienda, en la que se oferta una determinada bandeja a comensales.
Existencias: Acto de existir. Mercancías destinadas a la venta, guardadas en un almacén o tienda.

F

Framework: simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes.

G

GPL: *General Public License*.
Grasa: constituye el nutriente energético por excelencia, suministran los ácidos grasos esenciales y proporcionan al organismo las vitaminas liposolubles, aportan energía.
GRASP: *General Responsibility Asignment Software Patterns*.

H

HTML: *Hyper Text Markup Language* (Lenguaje para marcado de hipertexto).

I

Internet Explorer: navegador o visualizador de paginas web producido por *Microsoft* para *Windows,* usado para navegar por Internet y recorres diferentes sitios de la WWW (*World Wide Web*).

J

Java: lenguaje de programación orientado a objetos creado por **James Gosling**, que permite el diseño de aplicaciones para ser ejecutadas en las páginas Web.
JBuilder: es un entorno integrado de desarrollo que permite escribir, compilar, ejecutar y depurar programas en Java.

K

KDE: *K Desktop Environment*.

L

Linux: sistema operativo que se encuentra distribuido bajo Licencia Pública General de GNU o GPL, convirtiéndolo en software libre. Desarrollado por *Linus Torvalds* y con aportes de programadores de todo el mundo.

M

Mac OS X: sistema operativo basado en Unix.

Menú: Conjunto de platos que constituyen una comida

Microsoft: compañía liderada por Bill Gates que se ha convertido en la diseñadora más grande de programas para microcomputadoras, estableciéndose como la pionera de la industria informática. La compañía ha desarrollado estándares como Windows y se transformó en una líder de la innovación tecnológica.

N

NetBeans IDE: herramienta enfocada para el desarrollo usando el lenguaje Java y un entorno de desarrollo integrado (IDE).

O

Oracle JDeveloper: te permite crear diagramas UML para definir el proyecto y pasar a implementarlo.

Orden de producción: Documento que representa la aprobación para realizar la producción de un menú planificado. Con este documento y la solicitud de materiales se procede a la elaboración de los platos en las cantidades especificadas.

P

Pascal: lenguaje de programación estructurado y de alto nível, creado en la década de los 70, por Niklaus Wirth, es un lenguaje excelente para aprender, soporta la recursividad y brinda la posibilidad de trabajar con punteros.

Pentium III: microprocesador de arquitectura i686 fabricado por Intel

Perl: lenguaje de programación creado por Larry Wall, el cual no establece una filosofia concreta de programación aunque reúne varias, muy utilizado para la construcción de aplicaciones Web, es práctico para extraer la información de archivos de texto.

Periodo: Enmarca un rango de tiempo, se caracteriza por una fecha inicial y final.

PHP: es un lenguaje de programación creado por **Rasmus Lerdorf,** diseñado para realizar páginas dinámicas.

Plato: Conjunto de elementos que forman parte de una bandeja.

PostgreSQL: servidor de base de datos relacional orientada a objetos de software libre, avanzado y de código abierto, presenta licencia BSD (*Berkeley Software Distribution*).

Python: lenguaje de programación creado por Guido van Rossum, que permite la creación de un código elegante y limpio, ahorra tiempo en el desarrollo de programas, actualmente se desarrolla como un proyecto de código abierto.

R

RAM: *Random Access Memory.*

Recabar: recoger, guardar, recaudar.

S

Solicitud de materiales: Documento utilizado para realizar los pedidos de materiales a los almacenes correspondientes, con el fin de lograr tener los productos alimenticios disponibles en la despensa para ser elaborados posteriormente

SQL: *Structured Query Language* (Lenguaje de consultas estructurado).

Symfony: completo *framework* diseñado para optimizar el desarrollo de las aplicaciones web.

T

Tecnologías: Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico.

Tendencias: Propensión o inclinación en los hombres y en las cosas hacia determinados fines.

V

VPTS: *Visual Photo Time Stamp.*

W

Windows: entorno gráfico creado y diseñado por Microsoft, utilizado por la mayoría de los usuarios a nivel mundial ya que cuenta con una interfaz que hace que el trabajo sea fácil.

X

XMI: *Extensible Markup Language* (Lenguaje de Etiquetado Extensible).