Universidad de las Ciencias Informáticas

Facultad 5
Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE)

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

La Automatización de la Planificación del Menú para los Servicios Alimentarios en Instituciones de Nivel Superior

Autor: Arisney Figueredo Ramos

Tutor: Ing. Yoandry Martínez Rodríguez

XILEMA SIPME Sistema de Planificacion del menú

La Habana, Cuba. Julio, 2015.

XILEMA SIPME Sistema de Planificación de menú

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco 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 _____.

Arisney Figueredo Ramos

_____
Firma de Autor

Yoandry Martínez Rodríguez

_____
Firma del Tutor

XILEMA SIPME Sistema de Planificación de menú

DATOS DE CONTACTO

TUTOR:
Yoandry Martínez Rodríguez
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
Correo: yoandry@uci.cu

AGRADECIMIENTOS

A Dios. Por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos, además de su infinita bondad y amor.

A mis madres Eloisa y Carmen. Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor incondicional.

A mi padre Juan. Por los ejemplos de perseverancia y constancia que lo caracterizan y que me ha infundado siempre, por el valor mostrado para salir adelante y por su amor.

A mi hermana Arisneivi. Por ser mi hermana y amiga, pero sobre todo porque nuestro corazón está unido en todo momento.

A mi familia en general, mis tíos Mary y Vlady, a mi prima Daniellis. A mis abuelos, tíos, tías, primas y primos de III Frente, que a pesar de la distancia, siempre se han preocupado por mis estudios.

A Roger por ser más que un amigo durante todos estos años en la universidad.

A mis super amigos: Midiala, Alejandro, Roberto, Exxon, Lisardo, Yunior, Jordanis, Osmany, Manuel Alejandro, Annareya, Claudia María, Yohana Nieves. A todos mis amigos de la secundaria, politécnico y servicio militar.

A Yaiselin y Jose, les debo la vida.

A mi brigada 5503 por ser un pilar fundamental en mi formación profesional, en especial: Erduin, Yanet, Marín, Ernesto, Orson, Mirnerys, Kiki, Luis Angel, Liset y Susana.

A todas aquellas personas que de alguna forma contribuyeron a mi formación, y ayudarme a lograr lo que soy hoy.

"La magnitud de lo que logramos no depende de lo que tenemos, sino de lo que seamos capaces de hacer."

ERNESTO
Guevara de la Serna

Guerrillero heroico

RESUMEN

El presente trabajo tiene como objetivo desarrollar un sistema informático que apoye a la Dirección General de Alimentos (DGA) de la Universidad de las Ciencias Informáticas (UCI) en el proceso de planificación del menú para el servicio de alimentación.El sistema surge debido a que los documentos carta técnica y vale de solicitud se realizan de forma manual,lo que trae como consecuencia que en ocasiones se cometan errores en la planificación de los servicios de alimentación y pérdidas económicas a la universidad.La propuesta de solución se basa en el desarrollo de un sistema de apoyo a la planificación del menú, que permita la gestión del menú diario, de los productos existentes en el almacén, de los posibles platos a elaborar, así como las solicitudes que se les realizan a los menús planificados con anterioridad y la generación de los documentos necesarios en el proceso.El sistema desarrollado permite confeccionar el menú para el servicio de alimentación con mayor precisión, evitando grandes pérdidas económicas a la universidad.

Palabras claves: carta técnica, menú, planificación, vale de solicitud.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las Tecnologías de la Información y la Comunicación (TIC) se definen como un conjunto de técnicas, recursos y procedimientos que se utilizan con el fin de procesar, almacenar y transmitir información de forma digital(1). Tienen como objetivo la mejora de la calidad de vida de las personas dentro de un entorno social. Las TIC, cuya evolución avanza a pasos agigantados día tras día, presentan una relación directa con cambios de tipo procedimental, cultural, estratégico y productivo. Es por ello que exigen de las personas y organizaciones, que evolucionen al mismo ritmo para no quedar relegados en el pasado tecnológico.Constituyen un fenómeno que ha invadido todos los sectores de la sociedad, lo cual resulta beneficioso, pues libera a las personas de cargas pesadas logrando así, que el producto final pueda ser desarrollado con mejor calidad y en menor tiempo, además de que se optimizan los recursos empleados para ello.

Con la implantación de las nuevas tecnologías, se tiene una rápida adopción de sistemasinformáticos capaces de facilitar la realización de tareas con un alto nivel de complejidad. Ejemplo de estos son las que están orientados al tratamiento y administración de datos e información, también conocidos como sistemas de información. Estos realizan cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información, con el fin de apoyar las actividades de una empresa o negocio.

Con el objetivo de lograr eficacia y eficiencia en todos los procesos y por consiguiente mayor generación de riqueza y aumento en la calidad de vida de los ciudadanos, en la actualidad se lleva a cabo en la mayoría de los países, la informatización de la sociedad. Es evidente que para los países subdesarrollados resulta un reto el logro de este propósito, ya que su problemática fundamental está en lograr una alta incidencia en la modernización y calidad de todos los sectores de la vida.

Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las TIC; y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría a la sociedad cubana acercarse más hacia el objetivo de un desarrollo sostenible. Un ejemplo de que el país puede encaminarse a la modernización informática, es la UCI, la cual tiene como misión la formación de profesionales altamente calificados en esta rama. En la universidad existen disímiles proyectos que se encuentran inmersos en el desarrollo de aplicaciones y servicios informáticos, que apoyan los distintos servicios que brinda la propia universidad, así como empresas cubanas y extranjeras.

Uno de sus centros de desarrollo, el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE), desarrolla en la actualidad un sistema informático para brindar a la comunidad universitaria un mejor servicio referente a la reservación de los alimentos. Elproceso de planificación, llevado a cabo por la DGA de la UCI, resulta muy engorroso, pues es el

resultado de una larga jornada de trabajo consultando los documentos:plan de existencia en el almacén, los partes de comensales que se emiten diariamente por cada complejo comedor, el recetario técnico y la propuesta de menú semanal. Luego, se procede a elaborar la cartatécnica y el vale de solicitudpara poder extraer los productos del almacén. La entrega de estos documentos debe realizarse en el menor tiempo posible y con la mayor precisión. Las decisiones que se toman a la hora de crear el parte de los comensales, están basadas en la experiencia y apreciación de la planificación de días anteriores, lo que trae como consecuencia que en ocasiones se cometan errores en la planificación de los servicios de alimentación,provocandopérdidas económicas a la universidad.

Por tanto, la **situación problémica** es la siguiente:

- El proceso de planificación, llevado a cabo por la DGA de la UCI, resulta muy engorroso, pues es el resultado de una larga jornada de trabajo.
- La entrega de la carta técnica y el vale de solicitud no se realizaen el menor tiempo posible ni con la mayor precisión.
- Al crearse los partes de los comensales, se cometen errores en la planificación de los servicios de alimentación.

Por la situación planteada anteriormente, se identifica como **problema de investigación**: ¿Cómo realizar la planificación del menú para brindar un mejor servicio de alimentación?

Lo anterior se enmarca en el**objeto de estudio:**Los sistemasde planificación de servicios alimentarios.

Para resolver el problema planteado se propone el siguiente **objetivo general**: Desarrollar un sistema informático para el proceso de planificación del menú que permita mejorar el servicio de alimentación partiendo de los recursos existentes.

Todo lo anterior define como **campo de acción:** Los sistemas de planificación de servicios alimentarios en las instituciones universitarias con personal interno.

Para darle cumplimiento al objetivo se proponen las siguientes **tareas de investigación**:

1. Elaboración del marco teórico de la investigación a partir del estado del arte existentesobre el tema de investigación.
2. Determinación del flujo del proceso necesario para la planificación del menú del servicio de alimentación.
3. Diseño del sistema de planificación del menú para la UCI.

4. Implementacióndel sistema de planificación del menú parala UCI.

5. Validación del desempeño del sistema mediante las pruebas de calidad.

Los **métodos científicos de investigación**son utilizados para descubrir las relaciones internas y externas de los procesos de la realidad natural y social, siendo responsables de avances que se han producido en todos los campos científicos y que influyen en la sociedad. En la presente investigación se hace uso de algunos de estos métodos, entre los que se encuentran:

Métodos teóricos:

- **Histórico-lógico**: Este método se utiliza con el objetivo de estudiar la evolución de los antecedentes históricos de los sistemas existentes de planificación del menú. Permite valorar el desarrollo alcanzado, revelando sus principales etapas de desenvolvimiento.

- **Analítico-sintético**:Se emplea enel estudio de los sistemas de planificación, así como la búsqueda de información acerca de los lenguajes, herramientas y metodologías que son usados para la implementación de estos. También para estudiar cómo se desarrollan otros sistemas de planificación en un entorno de trabajo, analizando los resultados de sus deficiencias y características generales para así obtener un resultado eficiente que cumpla con los requisitos pedidos por el cliente.

- **Modelación**:Este método se utiliza para modelarlos diagramas y prototipos de interfaz con el objetivo de entender las necesidades, definir metas, objetivos y procesos como parte del desarrollo del sistema de planificación del menú para la UCI.

Métodos empíricos:

- **Consulta de la información en todo tipo de fuentes:** Mediante el mismo se selecciona la información necesaria para construir los aspectos conceptuales y metodológicos de la investigación.

- **Observación:**Se utiliza para estudiar más de cerca y obtener información detallada acerca el proceso de planificación del menú en la UCI.

- **Pruebas de validación**: Se emplea para demostrar el desempeño del sistema elaborado.

A continuación se describen los capítulos por los que está conformada la investigación:

Capítulo 1.Fundamentación teórica del sistema.
En este capítulo se definen aquellos conceptos que son de vital importancia para el desarrollo y entendimiento del problema de la investigación, presenta un análisis acerca de sistemas de planificación. Además, contiene un análisis de la metodología de desarrollo de *software*, herramientas, tecnologías y lenguajes que se usarán para la implementación del sistema.

Capítulo 2.Análisis y diseño del sistema.
Se describe el flujo actual del proceso de planificación, así como la propuesta de solución que responde a la problemática planteada. También, se definen los requisitos funcionales y no funcionales del sistema, así como el modelo conceptual, historias de usuario, diagrama de clases y patrones de diseño empleados.

Capítulo 3. Implementación y pruebas ala solución propuesta.
En este capítulo se describen los aspectos relacionados con la implementación de la solución planteada. Unido a esto, se definen los tipos de pruebas que se realizaron con el objetivo de validar que la solución funcione de manera correcta y cumpla con las especificaciones planteadas en las descripciones funcionales.

CAPITULO 1. FUNDAMENTACIÓN TEÓRICA DEL SISTEMA

1.1. Introducción

En el presente capítulo se reflejan distintos elementos que permiten fundamentar el tema abordado en la investigación. Se hace alusión alos conceptos fundamentales asociados al objeto de estudio con el fin de lograr un mejor entendimiento del problema de investigación y los fundamentos teóricos que sustentan la investigación. Se elabora un estudio acerca del comportamiento de sistemas similares a nivel mundial analizando las principales ventajas y desventajas. Además se realiza un estudio de la metodología, tecnologías, herramientas y lenguajes que serán usados en el desarrollo de la solución.

1.2. Conceptos asociados al dominio del problema

En esta sección se ofrecen algunos conceptos para el mejor entendimiento y comprensión de la investigación por parte del lector.

1.2.1. Sistema

Según la Real Academia Española (RAE) un **sistema** es:

1. m. Conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí.

2. m. Conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto(2).

En el Escritorio de Educación Rural Secundaria de Argentina se plantea como **sistema** al conjunto donde todos sus componentes están relacionados entre sí, posibilitando su correcto funcionamiento. Cuando una relación o componente es cambiado trae consecuencias en otros, afectando así su funcionamiento general(3).

Teniendo en cuenta lo anteriormente planteado, en esta investigación, se define como **sistema** al conjunto de funciones interrelacionadas: *hardware, software* y recursos humanos, que permiten almacenar y procesar información.

1.2.2. Planificación

1. f. Acción y efecto de planificar.

2. f. Plan general, metódicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado, tal como el desarrollo armónico de una ciudad, el desarrollo económico, la investigación científica, el funcionamiento de una industria, entre otros(4).

En términos generales, por planificación se refiere a la acción o resultado de planificar alguna cuestión, tarea o actividad que lleva a cabo un ser humano. La planificación debe ser entendida básicamente como un proceso que implicará la observación de una serie de pasos que se establecerán a priori y para los cuales, quienes realizan la planificación, deberán utilizar una serie de herramientas y expresiones(5).

En esta investigación la planificación se hace asociada a un evento específico, para una fecha determinada, donde evento es, en relación con la solución propuesta, un período de tiempo en el cual se brinda un servicio como desayuno, almuerzo, comida y merienda.

1.2.3. Menú

Procede del francés *menu*, es un término con varios usos de acuerdo al contexto. Puede tratarse del conjunto de los platos que constituyen una comida (desayuno, almuerzo, merienda o cena). También puede ser la carta donde se indican las comidas, los postres y las bebidas disponibles en un restaurante o en un establecimiento similar(6).

1.2.4. Sistema de planificación

Un sistema de planificación tiene como objetivo diseñar e implementar procesos de planificación y sistemas de información para la gestión que permitan a la institución disponer de la información necesaria para apoyar la toma de decisiones respecto a los procesos y resultados de la provisión de sus bienes y servicios, y rendir cuentas de su gestión institucional(7).

1.3. Sistemas homólogos

Luego de definir y describir los principales conceptos relacionados con la investigación científica, se hace un estudio acerca de los principales sistemas homólogos, atendiendo particularmente la forma en la que realizan la planificación.

1.3.1. PEEM: Programa de Elaboración y Evaluación de Menús

La aplicación informática Programa de Elaboración y Evaluación de Menús (PEEM) responde al diseño y la planificación de los menús alimentarios y dietas terapéuticas definidas para enfermos, acompañantes y trabajadores en cualquier institución de salud. La aplicación PEEM permite la construcción, edición, visualización e impresión del menú a elaborar diariamente; y la elaboración de las correspondientes órdenes de solicitud y entrega de víveres y alimentos a confeccionar. Adicionalmente, la aplicación PEEM almacena toda la información generada, a los fines de trazabilidad de la organización, elaboración de reportes estadísticos, y la conducción de ejercicios de inspección, auditoría y control de la calidad del proceso de la prescripción dietética hospitalaria(8).

Desarrollo de la aplicación PEEM

La aplicación PEEM se desarrolló como un *software* de utilización laboral orientado a un entorno multiusuario basado en un sitio *web* utilizando la tecnología *PHP-MySQL*. La programación de la parte dinámica en el lado del servidor se realizó en *PHP* 5.2.3 mientras que la parte del cliente se programó en *JavaScript*. Se empleó *MySQL*en su versión 5.0 como gestor de las bases de datos.

Características de la aplicación PEEM:

- Control de usuarios.
- Control de ingredientes.
- Cartas técnicas.
- Menú del día.
- Menú de la semana.

1.3.2. Cedrux

Es un sistema desarrollado en la UCI utilizando plataformas libres, el cual está proyectado como un paquete de soluciones integrales de gestión para las organizaciones cubanas. El sistema se encarga de la planificación de la demanda, los materiales, la capacidad y el procesamiento de órdenes en la empresa. Tiene la limitante que se encuentra en fase de desarrollo, por lo que aún no ha sido publicado su funcionamiento interno para ser estudiado en profundidad y obtener detalles acerca del mismo(9).

1.3.3. DIAL

Programa para evaluación de dietas y gestión de datos de alimentación. Este programa de nutrición responde a una necesidad constatada en la formación de estudiantes de nutrición y en la respuesta a cuestiones formuladas por diversos grupos de población (público en general y/o profesionales interesados en temas de alimentación). Tiene importantes aplicaciones para transformar la información sobre dieta ingerida, o planificada, en datos de energía y nutrientes que permiten valorar si esa dieta es correcta o incorrecta, y establecer aquellos aspectos en los que se debe mejorar, para poder introducir modificaciones rápidas, cambiando, añadiendo o sustituyendo algún alimento con el fin de determinar qué cambios se producen en la ingesta y en la cobertura de lo recomendado.
El programa DIAL está diseñado para calcular, programar y modificar de forma fácil, sencilla y rápida cualquier tipo de dieta. Es muy versátil, ya que con él se puede trabajar sin necesidad de ser expertos. Permite conocer al instante la energía y los principales nutrientes que contienen los alimentos y preparar fácilmente cualquier tipo de dieta. Las aplicaciones de este programa son muchas y variadas, siendo de gran utilidad en la práctica diaria de la nutrición y de la

dietética, no sólo para el profesional sino también para todas aquellas personas que deseen realizar una dieta sana y nutricionalmente equilibrada.
A continuación se describen sus principales características:

- Planificación, programación, control y seguimiento de menús y dietas de individuos y grupos (sanos y enfermos).
- Preparación de dietas para comedores escolares, residencias de ancianos, restaurantes, establecimientos de catering y en centros de Dietética.
- Para comparar el valor nutricional de diferentes alimentos y poder seleccionar aquellos ricos o pobres en un determinando componente.
- Planificación de programas nutricionales de distribución de alimentos.
- En la enseñanza en general y en la educación nutricional en particular.
- En la investigación, en la valoración del estado nutricional de individuos y grupos de población(10).

1.3.4. DieTools

Software de elaboración, evaluación y prescripción de dietas destinado al diseño, planificación, gestión y control de los menús alimentarios servidos en las instituciones hospitalarias. Es una herramienta capaz de gestionar todas las necesidades alimenticias de un centro hospitalario, desde la compra de materias primas hasta que la dieta es servida al paciente en función de sus necesidades. También permite controlar las dietas de los acompañantes, así como las comidas del personal.
DieTools, responde en todos los circuitos y departamentos involucrados en la dieta del paciente hospitalizado, ofrece menús interactivos para la captación del dato necesario; permite la emisión de los reportes requeridos por la gestión administrativa; y ofrece resúmenes estadísticos diarios y acumulados a los fines de evaluación del proceso. La aplicación hace posible la automatización del trabajo técnico-administrativo del Departamento de Dietética y Cocina y con ello, la reducción de errores durante el proceso de diseño y planificación de menús alimentarios.
La implementación de *DieTools* conlleva un aumento de la productividad laboral, seguridad en la alimentación del paciente, un ahorro importante en las compras y la optimización de los recursos humanos en el ámbito hospitalario. Las actividades que forman parte del proceso de la prescripción dietética implica el diseño del menú diario de alimentos, la planificación de los insumos necesarios para la elaboración de tal menú diario, y la emisión de órdenes y solicitudes a los actores involucrados para que procedan a cumplir las actividades que les son competentes dentro del proceso general.
DieTools responde al diseño y la planificación de los menús alimentarios y dietoterapéuticos que se sirven a enfermos, acompañantes y trabajadores en cualquier institución de salud. Además permite la construcción, edición, visualización, impresión del menú a elaborar diariamente, la elaboración de las correspondientes órdenes de solicitud y entrega de víveres y alimentos a

confeccionar. Adicionalmente, *DieTools* posibilita el almacenamiento de toda la información generada, a los fines de trazabilidad, elaboración de reportes estadísticos, gestión de costes y la administración de inspección, auditoría y control de la calidad del proceso de la prescripción dietética hospitalaria. *DieTools* incorpora las herramientas de cálculo necesarias para la conversión del menú alimentario en una orden de solicitud de víveres para el almacén hospitalario. La aplicación *DieTools* hace posible la edición de estas herramientas de cálculo en caso que se requiera, y de acuerdo a las revisiones hechas por los dietistas del centro indicará la materia prima necesaria para la elaboración de todos los menús.
Toda la información generada por la *DieTools* queda almacenada en tablas de datos apropiadas y puede ser recuperada y manipulada a voluntad del usuario para la construcción de reportes estadísticos diarios/acumulados, todos los informes se exportan a *Excel* para un mejor tratamiento(11).

1.3.5. Resultados

Los sistemas anteriores no se adecúan a las características específicas que presenta la DGA.En el caso del sistema PEEM es un sistema privativo. El país se está encaminando hacia la independencia tecnológica, por lo que representa un costo elevado utilizar este sistema. Los sistemas*Cedrux, DIAL y DieTools*, no muestran la forma en la que realizan la planificación, solamente mencionan que son elaborados con ese fin pero no hacen alusión al código fuente.

1.4. Herramientas, metodología y tecnologías utilizadas

Para el desarrollo del *software* es necesario el uso de diferentes herramientasy tecnologías, así como la metodología, a continuación se describen las principales características de las utilizadas para el desarrollo de la solución que se propone, las cuales fueron establecidas por la dirección del proyecto.

1.4.1. Lenguajes de programación

Un lenguaje de programación puede definirse como una notación[1] para escribir instrucciones u órdenes útiles para el ordenador y necesarias para la realización de determinado proceso(12).

Java *e*s un lenguaje de alto nivel que aplica un modelo de programación orientada a objetos, que se puede utilizar para crear aplicaciones completas que pueden ejecutarse en un único ordenador o ser distribuidos entre servidores y clientes en una red(13).

Es uno de los lenguajes de programación más usado en el desarrollo de aplicaciones, debido a sus características principales tales como:fácil de usar, multiplataforma, sumamente flexible, permite la creación de programas modulares y de códigos reutilizables. Dicho lenguaje además

[1] Por notación se entiende el sistema de signos convencionales adoptado para expresar ciertos conceptos matemáticos.

es compilado y permite el desarrollo de programas seguros y de alto rendimiento.

Proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir *socket*, además establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Java soporta la sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Es muy versátil ya que utiliza *byte-codes* que es un formato intermedio que sirve para transportar el código eficientemente. Por ser indiferente a la arquitectura sobre la cual está trabajando, esto hace que su portabilidad sea muy eficiente, sus programas son iguales en cualquiera de las plataformas, ya que *Java* especifica tamaños básicos, esto se conoce como la máquina virtual de java.

JavaScriptes un lenguaje de programación que se puede utilizar para construir sitios *web* y para hacerlos más interactivos. Además, es *Open Source*[2], por lo que cualquier persona puede utilizarlo sin comprar una licencia(14).Es un lenguaje interpretado orientado a las páginas *web*, con una sintaxis semejante a la del lenguaje *Java*. Presenta la tecnología *AJAX*[3] que es un concepto de programación basado en múltiples tecnologías *web* como *JavaScript* y *XML*[4] – de ahí el nombre de *AJAX*. La idea de *AJAX* es conectar a una página *web* con un servidor *web* sin que se recargue la página. Es por eso que se utiliza *AJAX*, porque será el responsable de establecer la conexión entre la página *web* y el servidor(15).

1.4.2. Marcos de trabajo

Algunos autores describen un *framework* (marco de trabajo) como una estructura de soporte definida en la cual otro proyecto puede ser organizado y desarrollado. *Marc Clifton*, en su artículo "*What is A Framework*" menciona varios elementos categóricos de los marcos de trabajo. Los mismos facilitan el trabajo con complejas tecnologías, unifican componentes/objetos haciéndolos más útiles, promueven la implementación consistente y flexible de aplicaciones. Además, pueden ser fácilmente depurados y probados (16).

Bootstrap

Bootstrap, nació por el año 2011, como una iniciativa para acelerar la creación de páginas *web*, en la actualidad se ha consolidado como un *framework* ineludible para los desarrolladores que se dedican a la creación y al diseño *web*. Hoy día, es uno de los proyectos abiertos más populares que existen en *Github*[5], con más de 14.000 colaboradores y tiene una gran receptividad al tener una alta compatibilidad con los navegadores más importantes, esto quiere decir que los desarrolladores tienen la tranquilidad de que sus códigos son interpretados

[2] De código abierto, término con el cual se conoce al *software* distribuido y desarrollado libremente. Esto significa que cuando los programadores pueden leer, modificar y redistribuir el código fuente de un programa, éste evoluciona y mejora.
[3] Asynchronous Java y XML.
[4] Lenguaje de Marcas Extensible, del inglés *eXtensible Markup Language*.
[5] Plataforma de desarrollo colaborativo.

correctamente y los usuarios visualizan lo que quieren expresar a través de los mismos. Básicamente lo que hace es pre-configurar clases (que son definiciones de estilos asociados a las diferentes etiquetas que utilizamos en el lenguaje *HTML*[6]) y con las mismas simplificar de forma muy considerable el código que había que escribir antes para conseguir el mismo resultado(17).

jQuery 1.11.1

jQuery es un *framework* de *JavaScript*, un conjunto de funciones que ya fueron desarrolladas y probadas. Estas funciones están listas para utilizarlas de una manera muy simplificada y se pueden lograr los mismos resultados en menos tiempo, sin necesidad de programar una funcionalidad completamente.

Permite agregar efectos y funcionalidades complejas a las aplicaciones *web*, como por ejemplo: validaciones de formularios, entre otras. Otra ventaja es la posibilidad que brinda de trabajar con *AJAX*, sin preocuparse de los detalles complejos de la programación. Además cuenta con la posibilidad de agregar *plugins*, facilitando más el trabajo (18).

Hibernate Framework4.3.1

Hibernate parte de una filosofía de mapear objetos Java, también conocidos como *POJOs*[7]. Una de las principales características de *Hibernate* es su flexibilidad, envolviéndolo todo bajo un *framework* común. Es una capa de persistencia objeto/relacional que permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. Es un *software* libre y está bajo licencia *LGPL*[8](19).

Spring Framework 4.0.1

Spring está compuesto por un conjunto de módulos, de los cuales se pueden tomar aquellos que faciliten la implementación del trabajo en cuestión. Ideado principalmente por *Rob Johnson*, es libre y de código abierto desde febrero del 2003. El centro de *Spring* está basado en el principio de Inyección de Dependencias. Esta técnica hace externa la creación y el manejo de las dependencias de las clases, con lo que se logra una mayor limpieza y claridad en el código.

Es un *framework* que ha venido a revolucionar la manera de programar aplicaciones *Java*,debido a la facilidad de crear componentes reutilizables. Se adapta fácilmente a otros *frameworks* y ofrece una autonomía a los desarrolladores, además de soluciones bien documentadas. Permite desarrollar un *software* seguro y robusto, mediante el uso de las prácticas comunes en la industria de *software*(20).

[6]Lenguaje de Marcas de Hipertexto del inglés HyperText Markup Language.
[7] Plain Old Java Objects
[8] Lesser GNU Public License

Spring contiene una gran variedad de características las cuales seorganizan en los siguientes módulos:

- ***Spring Core*:** "Núcleo", es la parte fundamental del *framework* ya que provee toda la funcionalidad de Inyección de Dependencias permitiéndose administrar la funcionalidad del contenedor de *beans*.
- ***Spring Context*:** "Contexto", provee de herramientas para acceder a los *beans* de una manera elegante. El paquete de contexto hereda sus características del paquete de *beans* y añade soporte para mensajería de texto.
- ***Spring* DAO:** Provee una capa de abstracción de *JDBC*que elimina la necesidad de teclear código tedioso y redundante. Permite que la programación del acceso a datos sea un poco más limpia y entendible. Permite administrar transacciones tanto declarativas como programáticas.
- ***Spring* ORM:** Provee capas de integración para API de mapeo objeto-relacional, incluyéndose *Hibernate* e *iBatis*. El mapeo de ORMse puede usar en conjunto con otras características de *Spring*, como la administración de transacciones mencionada con anterioridad.
- ***Spring AOP*:** Provee una implementación de programación orientada a aspectos. Sus funcionalidades permiten desacoplar el código de una manera limpia para implementar funcionalidades que por lógica y claridad debería estar separada.
- ***Spring Web*:** Provee características básicas de integración orientado a la *web* y se encarga de crear el contexto para las aplicaciones *web*. Incluye soporte para una variedad de tareas como la subida de archivos y la vinculación de parámetros de las peticiones a objetos del negocio.
- ***Spring Web MVC*:** Provee una implementación Modelo-Vista-Controlador para las aplicaciones *web*. La implementación de *Spring MVC* permite una separación entre código de modelo de dominio y las formas *web*.

1.4.3. Entorno integrado de desarrollo (*IDE*)

Un *IDE* es una aplicación compuesta por un conjunto de herramientas útiles para un desarrollador, que brinda ayudas visuales en la sintaxis, plantillas, *plugins* y sencillas opciones para probar y hacer un *debug*. Puede ser exclusivo para un lenguaje de programación o utilizarse para varios. Generalmente trae incluido un editor de código, un compilador, un depurador y en algunos casos un constructor de interfaz gráfica (21).

Eclipse Luna 4.4.1

Eclipse contiene un área de trabajo de base y un sistema de *plugin* extensible para personalizar el entorno. El IDE puede usarse para desarrollar aplicaciones principalmente en *Java*. Por medio de varios *plugins* de Eclipse también puede ser utilizado para desarrollar aplicaciones en

otros lenguajes de programación: Ada, ABAP, C, C + +, COBOL, *Fortran*, *Haskell*, *JavaScript*, *Lasso*, *Natural*, *Perl*, *PHP*, *Prolog*, *Python*,*Ruby* (incluyendo *Ruby* en *Rails*),*Scala*, *Clojure*, *Groovy*, *Scheme* y *Erlang*. Los entornos de desarrollo incluyen las herramientas de Eclipse *Java* de desarrollo (JDT) para *Java* y *Scala*, Eclipse CDT para C / C+ + y Eclipse PDT para *PHP*, entre otros(22).

1.4.4. Gestor de base de datos

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de datos (BD), asegurando su integridad, confidencialidad y seguridad(23).

PostgreSQL 9.1

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia de Distribución de *SoftwareBerkeLey* (BSD) y con su código fuente disponible libremente. *PostgreSQL* utiliza un modelo cliente-servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando(24).Entre las ventajas que brinda este gestor de base de datos se encuentran:

- Puede ser utilizado en los principales sistemas operativos: *Linux*, *Unix*, *Windows*, entre otros.
- Incorpora una estructura de datos "*array*".
- Permite la declaración de funciones propias.
- Soporta el uso de índices, reglas y vistas.
- Aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multi-hilos.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

Puede ser utilizado, modificado y distribuido gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos.

1.4.5. Contenedor de *servlets*

Apache Tomcat7.0.47

ApacheTomcat se desarrolla en un entorno abierto y participativo, publicado bajo la Licencia Apache versión 2. La herramienta está destinada a ser una colaboración de los mejores

desarrolladores de su clase en todo el mundo. El *Tomcat* en su versión 7.0.47 implementa el *Servlet* 3.0 y *JavaServerPages* (JSP) 2.2, especificaciones del *JavaCommunityProcess*, e incluye muchas características adicionales que lo convierten en una plataforma útil para el desarrollo y despliegue de aplicaciones y servicios web(25).

1.4.6. Lenguaje unificado de modelado

UML 2.0

UML[9] es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el ciclo de desarrollo de un producto de *software*. UML ofrece un estándar para describir modelos, incluyendo aspectos conceptuales tales como procesos de negocio y funciones de sistema, además de aspectos concretos como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reutilizables. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos, además prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que las notaciones y métodos usados para el diseño orientado a objetos han evolucionado, los modeladores solo tienen que aprender una única notación. UML se puede usar para modelar distintos tipos de sistemas, tanto de *software* como de *hardware*, además de permitir la organización del mundo real(26).

1.4.7. Herramienta CASE

Visual Paradigm 8 for UML

Visual Paradigm para UML (VP-UML) es una herramienta de diseño UML y herramienta CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Computadora) para UML diseñado para ayudar al desarrollo de *software*. Es compatible con los equipos de desarrollo de *software* en la captura de requisitos de planificación, código de la ingeniería, modelado de clases, modelado de datos, entre otros(27).

Entre las características más significativas por las cuales se escogió esta herramienta se encuentran la gestión de los flujos de trabajo.Además, está disponible para distintos sistemas operativos y en varios idiomas. Es fácil de instalar y actualizar. Presenta capacidades para gestionar ingeniería directa e inversa, y permite la generación de código *Java* a partir de modelos o diagramas.

1.4.8. Metodología de desarrollo de *software*

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo *software*. Puede seguir uno o

[9]Lenguaje de Modelado Unificado, del inglés *Unified Modeling Language*.

varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo(28).

Las metodologías están encaminadas a estructurar, planear y controlar el proceso de desarrollo en sistemas de información. Toda metodología debe estar adecuada a las necesidades del *software* y cuánto pueda abarcar el mismo. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo *software*, pero los requisitos de un *software* a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del mismo.

Para guiar el desarrollo de la solución planteada se empleará la metodología *AUP*[10]adaptada al ciclo de vida definido para la actividad productiva de la UCI, debido a que es la metodología establecida por la universidad para los proyectos en desarrollo.

Proceso Unificado Ágil o *Agile Unified Process*

El Proceso Unificado Ágil o *Agile Unified Process* (*AUP)*en ingléses una versión simplificada deRUP. Esta describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega una fase de Cierre. Para una mayor comprensión se muestra la siguiente tabla.

Tabla 1 Fases de AUP adaptada a la actividad productiva de la UCI

Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el

[10] Versión simplificada de la metodología tradicional Rational Unified Process (RUP).

	negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del *software*.
Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Se consideran las siguientes disciplinas:

Modelado de negocio (opcional): Disciplina destinada a comprender los procesos de negocio de una organización, con el objetivo de asegurar queel *software* desarrollado va a cumplir su propósito.

Requisitos: Empleada para desarrollar un modelo del sistema a construir. Además, comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.

Análisis y diseño: En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema. También, se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos. Los modelos desarrollados son más formales y específicos.

Implementación:A partir de los resultados del Análisis y Diseño se construye el sistema.

Pruebas internas: Se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.

Pruebas de liberación: Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

Pruebas de aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el *software* está listo y que puede ser empleado por usuarios finales para ejecutar aquellas funciones y tareas por las cuales se construyó el *software*.

Despliegue (opcional): Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente (29).

Entre las técnicas ágiles que utiliza AUP se encuentra el Modelado ágil que permite encapsular los requisitos funcionales en Historias de Usuarios(HU) o en Descripción de requisitos por procesos. La otra forma de encapsular los requisitos es por Casos de Uso. Para la solución planteada se hará uso de la técnica de Historias de Usuarios para encapsular los requisitos funcionales.

1.4.9. Gestión y construcción de proyecto

ApacheMaven3.0.5

Apache *Maven* es una herramienta de gestión y construcción de proyectos *Java,* que utiliza un modelo de objeto de proyecto o *Project Object Model* (POM) para describir el proyecto de *software* a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos.

El objetivo principal de *Maven* es permitir al desarrollador comprender el estado completo de un esfuerzo de desarrollo en el menor período de tiempo. Para alcanzar este objetivo hay varias áreas de preocupación que *Maven* intenta tratar como:

- Hacer fácil el proceso de construcción.
- Proporcionar un sistema de construcción uniforme.
- Proporcionar información sobre los proyectos de calidad.
- Proporcionar directrices para las mejores prácticas de desarrollo.
- Permitir la migración transparente a nuevas características (30).

Las partes del **ciclo de vida** principal del proyecto *Maven* son:

1. ***Compile***: Genera los ficheros .class compilando los fuentes .java
2. ***Test***: Ejecuta los test automáticos de *JUnit* existentes, abortando el proceso si alguno de ellos falla.
3. ***Package***: Genera el fichero .jar con los .class compilados
4. ***Install***: Copia el fichero .jar a un directorio del ordenador donde *Maven*guarda todos los .jar. De esta forma esos .jar pueden utilizarse en otros proyectos *Maven* en el mismo ordenador.
5. ***Deploy***: Copia el fichero .jar a un servidor remoto, poniéndolo disponible para cualquier proyecto *Maven* con acceso a ese servidor remoto.

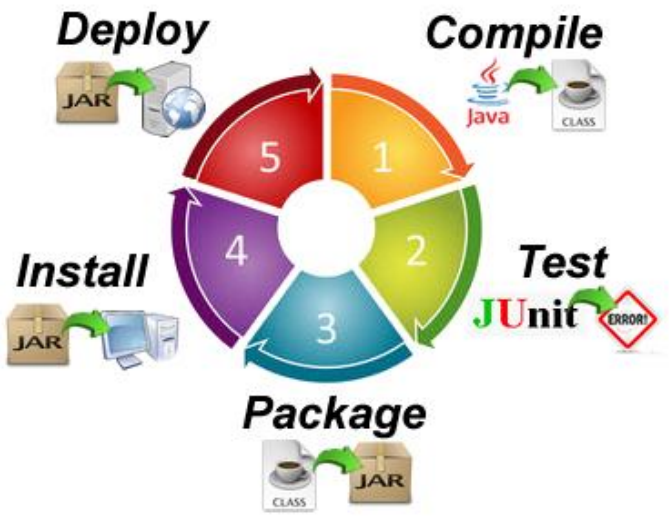


Figura 1. Ciclo de vida de un proyecto Maven. Elaboración propia

1.5. Conclusiones parciales

Una vez finalizado este capítulo se puede llegar a las siguientes conclusiones:

- El estudio realizado en el presente capítulo permitió determinar las principales características de la solución a implementar.
- Como guía en el desarrollo del *software* se escogió la metodología AUP.
- Se seleccionaron como lenguajes, herramientas y tecnologías para el desarrollo dela solución propuesta, la herramienta *CASE Visual Paradigm*, *UML*como lenguaje de modelado, IDE de desarrollo el Eclipse Luna y lenguaje de programación *Java*para lograr el objetivo planteado al inicio de la investigación.

CAPITULO 2. ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción

En el presente capítulo se procede a realizar la descripción de la propuesta de solución y el flujo actual del proceso de negocio relacionado con el objeto de estudio que lo soporta. Se realizan una serie de tareas que son imprescindibles, para lograr un desarrollo exitoso del sistema. Se especifican además los requisitos principales del sistema mediante los requerimientos funcionales y no funcionales.

2.2 Estado actual del proceso: Planificación del menú

La DGA en la UCI es la que lleva a cabo la planificación y control de una serie de actividades asociadas al servicio de los complejos comedores. Todos los viernes, se reúne con el objetivo de confeccionar una plantilla con el menú semanal que contiene, por cada evento (Desayuno, Almuerzo, Comida y Merienda), los distintos platos a elaborar. Dicha plantilla es enviada a los complejos comedores donde se tiene en cuenta para decidir la cantidad de comensales a cocinar según el plato definido en dicho menú, luego se envía a la DGA un parte de comensales.

Hoy día estos procesos se realizan de forma manual, con solo el apoyo de herramientas ofimáticas como es el caso de *Microsoft Office*, específicamente *Excel*. El almacenamiento de toda la información generada por las constantes solicitudes no se registra en un solo lugar.Al contrario, se basa en las copias físicas de cada una de las solicitudes y aunque se almacenan las copias digitales por separadas de cada solicitud así como los vales, la consulta o modificación de estos puede tornarse engorroso tras algún error cometido.Para un mejor entendimiento del proceso, Ver la Figura 2.

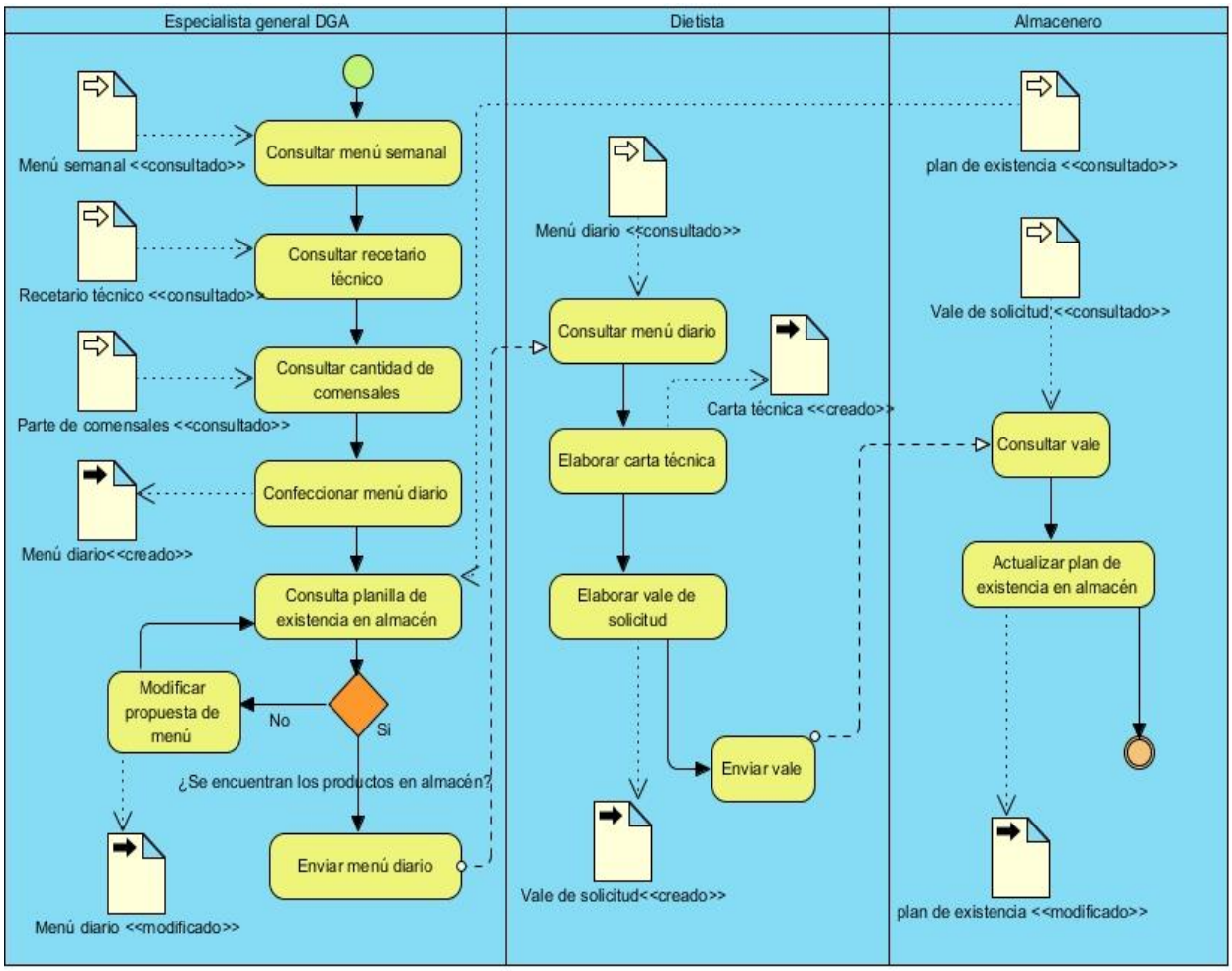


Figura 2. Diagrama de proceso de negocio. Elaboración propia

2.3 Propuesta de solución

Para darle solución a la problemática planteada se decide la implementación de un Sistema de Apoyo a la Planificación del Menú en los servicios alimentarios que permita a los trabajadores de la DGA la elaboración del menú para los comensales, así como la posibilidad de generar los documentos carta técnica y vale de solicitud. A continuación se enuncian algunas de las especificaciones con que cuenta la propuesta de solución:

- No debe incluir un algoritmo para planificar de manera automática el menú, sino que sirva para automatizar un proceso que hoy día se realiza de manera manual.

- Sistema basado en autenticación por roles definidos en la BD para los usuarios con acceso al sistemausando además las credenciales presentes en el servidor de direcciones LDAP UCI.

- Con el usuario autenticado el sistema debe permitir la gestión de los productos en

existencia, lo que permitirá contar en BD con los insumos necesarios para la confección de los platos.

- Debe permitir la gestión de los platos a partir de los productos presentes en la BD, lo que permite la creación de los menús, en cualquiera de sus eventos (Desayuno, Almuerzo, Comida y Meriendas).

- Teniendo ya almacenado todos los datos referentes a los productos, platos y menús, debe permitir la creación delasolicitud del evento referente a un menú, una vez en persistencia los datos.

- Debe garantizar la generación de los vales de las solicitudes para realizar las peticiones a los almacenescon el objetivo de extraer los productos en existencia para la cocción de los alimentos del menú.

2.4 Modelo de dominio

Un modelo de dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de ejecución, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de *software* sino de la propia realidad física.

Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de *software* o de otro tipo. Similares a los mapas mentales utilizados en el aprendizaje, el modelo de dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir(31).A continuación, en la Figura 3. Modelo de dominio, se especifican aquellos conceptos asociados al domino del problema.

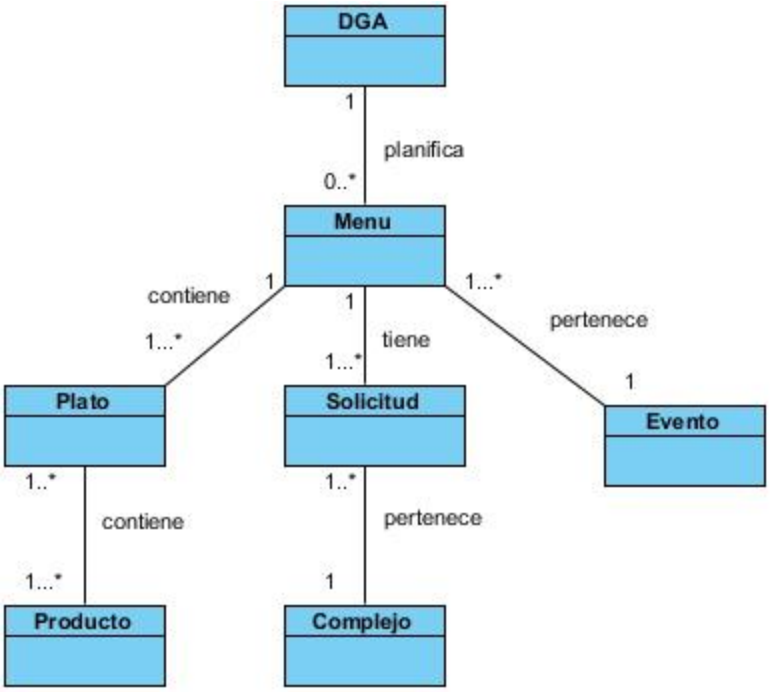


Figura 3. Modelo de dominio. Elaboración propia.

A continuación se realiza una breve descripción de los principales conceptos utilizados al modelo del dominio:

- **DGA**:Representa la dirección encargada de realizar la planificación de los menús diarios.
- **Menú**:Concepto encargado de representar el conjunto de platos asociados a un evento del día.
- **Plato**:Contiene el conjunto de productos necesarios para la elaboración de un plato.
- **Producto**:Representa la materia prima utilizada para elaborar un plato.
- **Evento**:Define para qué momento del día se planifica un menú, ya sea Desayuno, Almuerzo, ComidaoMerienda.
- **Solicitud**:Concepto que se utiliza para definir la demanda necesaria de comensales para un menú determinado.
- **Complejo**:Concepto encargado de representar el lugar en el que se desarrolla el evento.

2.5 Especificación de los requisitos del *software*

El análisis de requisitos es una de las etapas más importantes en el ciclo de desarrollo de un *software*. La captura, análisis y especificación de requisitos es crucial, pues de esta etapa depende en gran medida el éxito de los objetivos planteados. Para la captura de los mismos se aplicó la técnica de levantamiento de requisitos:Tormenta de ideas, en la cual el cliente expuso sus necesidades y criterios, logrando una mayor comprensión. Luego de capturados se enumeran a través de requisitos funcionales y no funcionales, todas las acciones que el sistema

deberá ser capaz de realizar.

2.5.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares(32).Los requisitos funcionales con los que cuenta el sistema se muestran en la siguiente tabla:

Tabla 2 Requisitos funcionales del sistema

Nº	Funcionalidad	Descripción	Prioridad	Complejidad
RF1	Listar producto.	El sistema debe mostrar un listado con los productos existentes.	Baja	Media
RF2	Adicionar producto.	El sistema debe crear un nuevo producto en el listado de productos existentes una vez sean introducidos los datos del producto, tales como: - Nombre del producto. - Cantidad para el almacén 12, 14 y 17. - UM de servicio. - UM de almacén. - Equivalencia. - Precio del producto. - Tipo de producto.	Alta	Alta
RF3	Mostrar producto.	El sistema debe mostrar los detalles de un producto existente.	Media	Media
RF4	Modificar producto.	El sistema debe modificar los datos de un producto existente a partir de los datos de este.	Alta	Alta
RF5	Filtrar producto.	El sistema debe buscar un producto en el listado de productos existentes a partir de un criterio de búsqueda.	Media	Media
RF6	Asignar rol.	El sistema debe permitir el establecimiento de un rol determinado a un usuario autenticado.	Alta	Alta
RF7	Listar plato.	El sistema debe mostrar un listado con los platos creados con anterioridad.	Baja	Media
RF8	Adicionar plato.	El sistema debe crear un nuevo plato	Alta	Alta

		en el listado de platos existentes, una vez introducidos los datos para el mismo como: - Nombre. - Norma. - UM de la norma. - Kilocalorías.		
RF9	Mostrar plato.	El sistema debe mostrar los detalles de un plato existente.	Media	Media
RF10	Modificar plato.	El sistema debe modificar los datos de un plato en el listado de platos existentes.	Alta	Alta
RF11	Filtrar plato.	El sistema debe buscar un plato en el listado de platos existentes.	Media	Media
RF12	Modificar roles asignados.	El sistema debe permitir la modificación de los roles asignados a un usuario.	Alta	Alta
RF13	Listar menú.	El sistema debe mostrar un listado con los menús creados con anterioridad.	Media	Media
RF14	Adicionar menú.	El sistema debe adicionar un menú en el listado de los menús existentes una vez sean introducidos los datos de este: - Evento. - Fecha. - Platos asociados a este. - Solicitudes.	Alta	Alta
RF15	Mostrar menú.	El sistema debe mostrar los detalles de un menú existente.	Media	Media
RF16	Modificar menú.	El sistema debe modificar los datos de un menú existente.	Alta	Alta
RF17	Filtrar menú.	El sistema debe buscar un menú, según un criterio específico.	Media	Media
RF18	Listar solicitud.	El sistema debe listar las solicitudes realizadas, así como agregar o modificar las solicitudes, además mostrar detalles de una solicitud.	Media	Media
RF19	Ingresar solicitud.	El sistema debe ingresar al sistema la	Alta	Alta

		cantidad de comensales para los cuales se cocinará, una vez introducidos los datos de esta: - Complejo Comedor. - Cantidad de comensales.		
RF20	Modificar solicitud.	El sistema debe modificar la cantidad de comensales para los que se solicita un menú.	Alta	Alta
RF21	Mostrar solicitud.	El sistema debe mostrar los detalles de una solicitud.	Media	Media
RF22	Generar vale de solicitud.	El sistema debe permitir la generación de un vale de solicitud por complejo comedor.	Alta	Alta
RF23	Generar carta técnica.	El sistema debe permitir la generación de una carta técnica por complejo comedor.	Alta	Alta

2.5.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares (32).En las siguientes tablas se muestranlos requisitos no funcionales con los que cuenta el sistema en sus diferentes clasificaciones:

Tabla3 Usabilidad

Requisitos no funcionales	Descripción
RNF1.Navegación intuitiva.	El sistema deberá poseer una interfaz intuitiva, con un menú de funcionalidades en la parte superior y un área de contenido debajo de esta para optimizar el área de visualización.
RNF2. Usuario informado.	Mantener al usuario informado de los resultados de las operaciones realizadas por él a partir de mensajes de confirmación e información tras las acciones realizadas.

Tabla 4 Portabilidad

Requisitos no funcionales	Descripción
RNF3. Sistema multiplataforma.	El sistema debe ser multiplataforma, siendo compatible con cualquier sistema operativo (SO GNU/Linux o SO *MicrosoftWindows*).

Tabla 5 Seguridad

Requisitos no funcionales	Descripción
RNF4. Autenticación mediante directorio LDAP.	El sistemadeberá permitir la autenticación mediante el directorio LDAP que proporciona la UCI.
RNF5. Permitir el acceso al sistema basado en roles.	El sistema debepermitir la autorización a usuarios, necesaria para que cada usuario tenga los permisos que le han sido asignados según su rol.

Tabla 6 Requisitos de hardware

Requisitos no funcionales	Descripción
RNF6. Memoria RAM.	La memoria RAM mínima deberá ser de 512MB o más.
RNF7. Espacio en disco duro.	Se requiere al menos 1024MB de espacio libre en el disco duro.
RNF8. Procesador Intel Pentium.	El sistema necesita al menos un procesador Intel Pentium IV, con una frecuencia de 2.0GHz o superior.

Tabla 7 Requisitos de software

Requisitos no funcionales	Descripción
RNF9. Estación de trabajo.	Deberá tener instalado el navegador: Mozilla Firefox en su versión 13 o superior, Internet Explorer 9 o superior. Apache Tomcat 7.0.47 o superior, PostgreSQL en su versión 9.1 o superior.

Tabla 8 Fiabilidad

Requisitos no funcionales	Descripción
RNF10. Disponibilidad.	El sistema deberá estar disponible para que los usuarios accedan cuando lo deseen, 24 horas los 7 días de la semana.

Tabla 9 Interfaz de usuario

Requisitos no funcionales	Descripción
RNF11.Interfaces específicas para cada elemento.	El sistema deberá mostrar una interfaz específica para la gestión de cada uno de sus elementos.
RNF12. Ambiente agradable.	El sistema deberá tener un ambiente agradable y sencillo, combinado con colores serios y una estructura intuitiva que permita que el usuario se adapte con facilidad.

2.6 Historias de usuarios

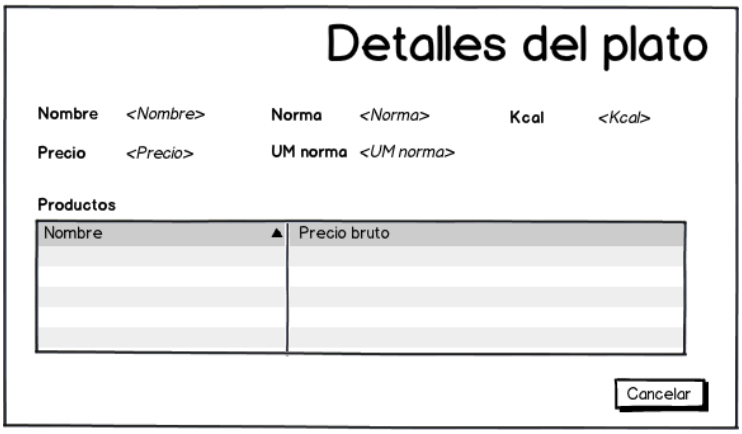
Una HU es una representación de un requisito de *software* escrito en una o dos frases utilizando el lenguaje común del usuario. Las Historias de Usuarios son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos. Las Historias de Usuarios son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las Historias de Usuarios permiten responder rápidamente a los requisitos cambiantes(33).

Durante el diseño de la propuesta de solución se identificaron 23 historias de usuario que responden a las diferentes funcionalidades solicitadas por el cliente y presentan una descripción para que el desarrollador conozca su posterior implementación. A continuación se describen las Historias de Usuariosreferentes alos requisitos funcionales “Listar producto” y “Mostrarplato”, el resto de las historias se encuentran en los anexos del 3 al 23.

Tabla 10 Historia de usuario “Listar producto”

Historia de usuario	
Número 1	**Nombre de la Historia de Usuario:** Listar producto
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 15 días
Riesgo en desarrollo: Alto	**Puntos reales:**120 horas
Descripción: El sistema debe permitir que al hacer clic en la opción “Gestionar producto”, se muestre un listado con los productos existentes. Para cada producto mostrado se habilitarán las opciones “Mostrar detalles del producto” y “Modificar producto”.	
Observaciones:NA	
Prototipos de interfaces: 	

Tabla 11 Historia de usuario "Mostrarplato"

Historia de usuario	
Número 9	**Nombre de la Historia de Usuario:** Mostrar plato
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 15 días
Riesgo en desarrollo: Alto	**Puntos reales:**120 horas
Descripción: El sistema permitirá que al hacer clic en la opción "Detalles del plato" correspondiente a un plato determinado, se muestre un formulario con los datos del plato ya registrado, de forma no editable, para la consulta de los mismos. Se mostrará la opción "Cerrar".	
Observaciones: NA	
Prototipos de interfaces: 	

2.7 Arquitectura del sistema

La arquitectura es la parte de la Ingeniería de *Software* que se ocupa de la descripción y el tratamiento de la estructura de un sistema como una serie de componentes, con el fin de organizaradecuadamente los distintos subsistemas, y permitir la integración de diferentes grupos de desarrollo en el mismo proyecto. Habitualmente se vincula con la fase de ejecución, aunque este detalle no es estrictamente necesario.Su principal objetivo es hacer énfasis en la importancia de la descripción estructural de los sistemas *software*, un aspecto bien conocido pero poco tratado(34).

La definición oficial que es utilizada es la provista por el documento lEEE STD 1471- 2000, que expresa: "La arquitectura del *software* es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución".

Para la implementación de la propuesta de solución se definió una arquitectura en 3 capas. A continuación se describen cada una de las capas:

1. Capa de presentación.

Presenta el sistema al usuario con el cual se comunica, además de capturar y mostrar información. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser amigable (entendible y fácil de usar) para el usuario. Dentro de esta capa se aplica el patrón arquitectónico MVC en las vistas con seguridad de *Spring*. Esta capa está contenida dentro delpaquete *views*.

2. Capa de negocio.

Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos para solicitar al gestor de base de datos con el objetivo de almacenar o recuperar datos. Esta capa está contenida en los paquetes *domain* y *service*, las cuales mantienen comunicación con las otras capas del sistema.

3. Capa de datos.

Es donde residen los datos, además de ser la encargada de acceder a los mismos. Está formada por uno o más SGBD que realizan todo el almacenamiento de estos, y reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. La utilización de *Hibernate* junto al patrón DAO brinda la posibilidad de la no dependencia de un SGBD único, sino que es flexible a un cambio en dicha tecnología. Esta capa en el sistema estará contenida en el paquete dao.

Ventajas de la arquitectura 3 capas:

- Soporta un diseño basado en niveles de abstracción crecientes, lo que permite a los desarrolladores la partición de un problema complejo en una secuencia de pasos incrementales.
- Admite naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.
- El módulo de la capa intermedia puede ser reutilizado por múltiples aplicaciones si

está diseñado en forma modular.

- La interfaz del cliente no es requerida para comunicarse con el receptor de datos, por lo tanto esa estructura de datos puede ser modificada sin cambiar la interfaz del usuario.

- Las llamadas de la interfaz del usuario en la estación de trabajo, al servidor dela capa intermedia, son más flexibles que en el diseño de dos capas, ya que la estación sólo necesita transferir parámetros a la capa intermedia(35).

A continuación se muestra la Figura 4. Diagrama de Arquitectura 3 capas, correspondiente a la arquitectura de la solución planteada.

Figura 4. Diagrama de Arquitectura 3 capas.

2.7.1 Patrones

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de *software*. Proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos.

Los patrones de *software* son mecanismos con el objetivo de dar solución a problemas que ocurren repetidamente dentro de un contexto muy bien definido. Las soluciones propuestas a través de estos, involucran algunas clases de estructuras que permiten contemplar los requisitos no funcionales. Estos requisitos atraviesan diferentes niveles de abstracción y etapas del ciclo de vida, partiendo del análisis del dominio, pasando por la arquitectura de *software* y llegando hasta el nivel de los lenguajes de programación(36).

2.7.2 Patrones arquitectónicos

Para el desarrollo de *software* existen buenas prácticas en el diseño arquitectural, especialmente en cuanto a la arquitectura lógica a gran escala, estas prácticas se han escrito

en forma de patrones. Un patrón de arquitectura de *software*, es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes, sus responsabilidades, relaciones, y las formas en que colaboran.

Los patrones arquitectónicos son los que definen la estructura de un sistema, los cuales a su vez se integran de subsistemas con sus responsabilidades. Tienen una serie de directivas para organizar los componentes del sistema, con el objetivo de facilitar la tarea del diseño (37).

A continuación se enuncian las **principalescaracterísticas**:

- Sintetizan las soluciones arquitectónicas y estructurales dentro del tipo de problema que modelan.
- Permiten identificar y comprender la arquitectura del sistema.
- Permiten la reutilización de soluciones arquitectónicas de calidad.
- Son de gran ayuda para controlar la complejidad de un diseño.
- Facilitan la documentación de diseños arquitectónicos.
- Proporcionan un vocabulario común que mejora la comunicación entre diseñadores.

El *framework* Spring utiliza en su implementación patrones que clasifican y describen formas de solucionar problemas específicos y comunes a la hora de desarrollar aplicaciones. En la propuesta solución se utilizará el siguiente:

Modelo-Vista-Controlador (MVC).
Divide una aplicación interactiva en tres componentes. 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 permita 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. El patrón de arquitectura MVC ayuda a reducir la complejidad en el diseño arquitectural e incrementar la flexibilidad y la usabilidad(38).

2.7.3 Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. Cada patrón permite que algunos aspectos de la estructura del sistema puedan cambiar independientemente de otros aspectos. Facilitan la reusabilidad, extensibilidad y mantenimiento.
Cuando se habla de estos patrones no se da una definición como tal, pero se considera que son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la Programación Orientada a Objetos (POO). Un patrón de diseño puede considerarse como un documento que define una estructura de clases que aborda una situación

particular, permitiendo la reutilización de cada patrón las veces que sean necesarias, simplificando así el tiempo en el desarrollo del sistema (37).

Patrones de diseño GRASP

Los Patrones Generales para Asignar Responsabilidades: GRASP (en inglés: *GeneralResponsibility Asignment Software Patterns*), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Su nombre se debe a la importancia de captar estos principios, para diseñar eficazmente el *software* orientado a objetos (39).

GRASPestá compuesto por los siguientes:

- **Patrón Experto:** Este patrón consiste en asignar una responsabilidad al experto en información, es decir, a la clase que cuenta con la información necesaria para cumplir con la responsabilidad. Se refuerza el encapsulamiento y se favorece el bajo acoplamiento. En la solución, las clases del dominio cumplen con este patrón, debido a que cada una de estas se encargan de manejar sus propias acciones con la información que contienen. En este caso puede ser evidenciado en la clase ProductoController que recibe todas las peticiones provenientes de un producto.
- **Patrón Creador:** Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Su propósito principal es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Brinda soporte al bajo acoplamiento. En este caso puede ser evidenciado en las clases *controller* que crean las instancias propias a almacenar en la BD.
- **Patrón Bajo Acoplamiento:** Pretende asignar una responsabilidad para mantener el bajo acoplamiento, es decir, el diseño de clases más independientes, que no se relacionen con muchas otras, que reduzcan el impacto de los cambios, que sean más reutilizables y acrecienten la oportunidad de una mayor productividad. En este caso puede ser evidenciado en las clases *controller* MenuController y PlatoController, las que dependen en todo momento de las clases PlatoController y ProductoController respectivamente.
- **Patrón Alta Cohesión:** Su objetivo es asignar una responsabilidad, de modo que la cohesión siga siendo alta. Las clases con alta cohesión se caracterizan por tener responsabilidades estrechamente relacionadas y no realizar un trabajo enorme. Una clase con alta cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño.

- **Patrón Controlador:** Consiste en asignar la responsabilidad a una clase para manejar los mensajes correspondientes a eventos de un sistema. El controlador es un intermediario entre la interfaz de usuario y el núcleo de las clases donde reside la lógica de la aplicación. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación para aumentar la reutilización del código y a la vez tener un mayor control. Un ejemplo de esta es la *PlatoController*.

Patrones de diseño GoF

Por sus siglas en inglés Gang of Four. Llamados así debido a que fueron cuatro autores los que escribieron el libro “Design Patterns” que ilustra sus funciones(39).
A continuación se describen:

- **Patrón Fachada**: Patrón estructural que trata de simplificar la interfaz entre dos sistemas o componentes de *software*. Oculta un sistema complejo detrás de una clase que funciona como pantalla o fachada. La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo definen, de forma que solo se ofrezca un punto de entrada al sistema cubierto por la fachada. Su uso aísla los posibles cambios que se puedan producir en alguna de las partes. Se puede encontrar la utilización de este patrón en la entidad ServiceFacade para acceder a los objetosDAO.

- **Patrón*Singleton***: Patrón creacional diseñado para restringir la creación de objetos pertenecientes a una clase. Su objetivo es garantizar que una determinada clase solo tenga una instancia y proporcionar un punto de acceso global a esta. Es un patrón muy sencillo de diseñar y a menudo es implementado por otros patrones. En este caso puede ser evidenciado en la clase ServiceFacade.

Patrón Data Access Object (DAO)

Este patrón maneja la conexión con la fuente de datos para obtener y almacenar información. Su uso ofrece varios beneficios para la persistencia de datos como son la separación del acceso a datos de la lógica de negocio, lo cual suele ser altamente recomendable en sistemas medianos o grandes, o que manejen lógica de negocio compleja. Permite el encapsulamiento de la fuente de datos, lo cual es especialmente beneficioso en sistemas con acceso a múltiples entradas. Posibilita el ocultamiento de la API con la que se accede a los datos, lo que viabiliza que se pueda cambiar el negocio del manejo de los datos persistentes sin que afecte el resto. Por ejemplo cambiar de *framework*de acceso a datos, centralizando todo el acceso a datos en una capa independiente(40). En este caso puede ser evidenciado en las clases DAO que actúan.

2.8 Diagrama de clases del diseño

El diagrama de clases del diseño no es más que una representación de clases y asociaciones. Los tipos de clases que se utilizan describen la relación que tienen al momento de la ejecución de la aplicación y cómo se realiza la comunicación entre ellas(41). En la Figura 5 se muestra el diagrama de clases de la solución propuesta.

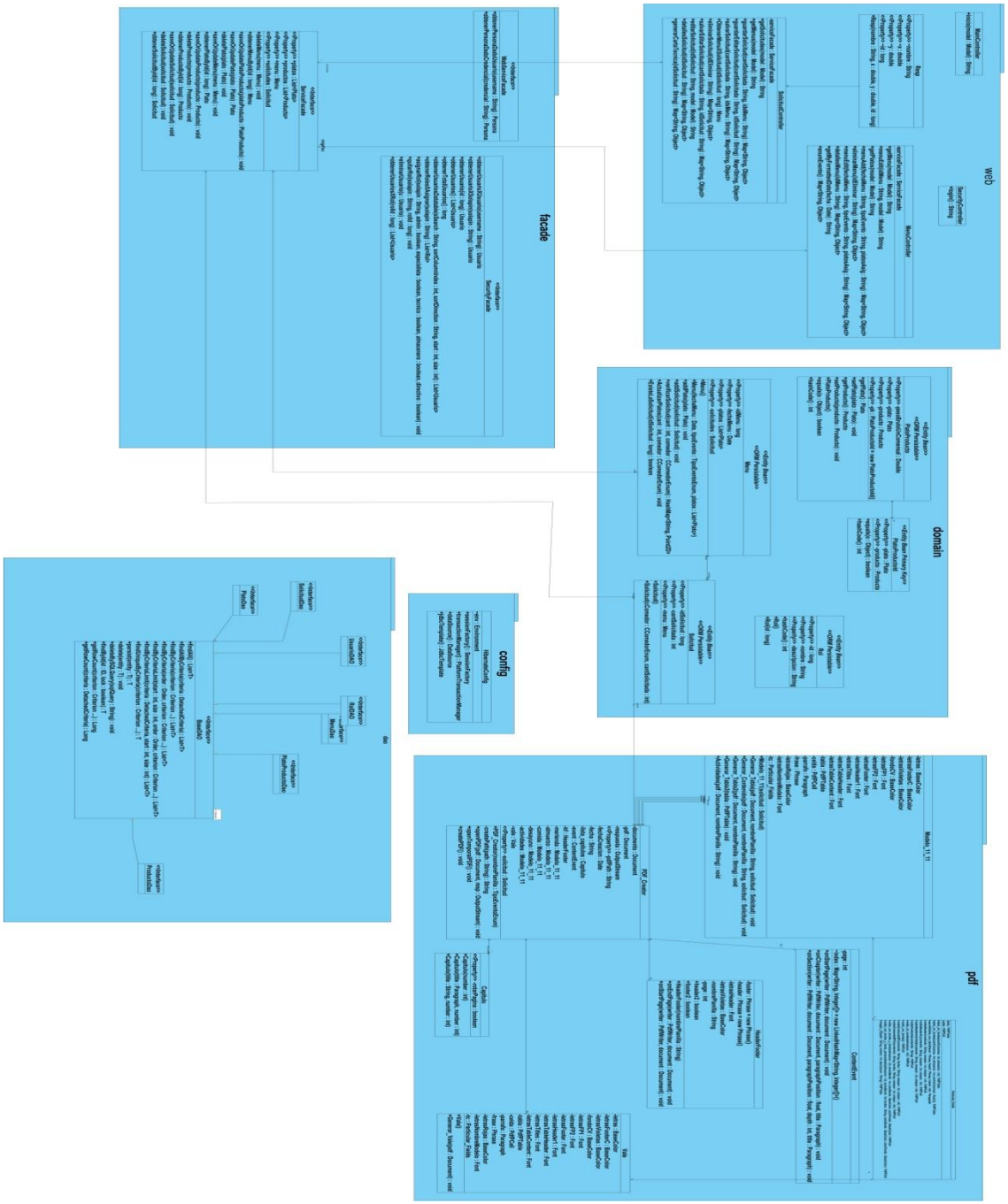


Figura 5. Diagrama de clases Elaboración propia.

2.8.1 Descripción de las clases

A continuación se realiza una descripción de las principales clases correspondientes al sistema:

Tabla 12 Descripción de la clase Producto

Descripción de la Clase *Producto*	
Nombre: Producto	
Tipo de clase: Modelo	
Atributos	**Tipos**
idProducto	*private*
nombreProducto	*private*
tipoProducto	*private*
equivalencia	*private*
almacenUM	*private*
servicioUM	*private*
cantAlm12	*private*
cantAlm14	*private*
cantAlm17	*private*
precioCompraProducto	*private*
Responsabilidades	
Nombre:	getIdProducto () : *long*
Descripción:	Obtiene el identificador de un producto.
Nombre:	getNombreProducto () : *String*
Descripción:	Obtiene el nombre de un producto.
Nombre:	getTipoProducto () : TipoProductoEnum
Descripción:	Obtiene el tipo de un producto.
Nombre:	getEquivalencia () : *Double*
Descripción:	Obtiene el valor en unidad de medida de un producto en el almacén.
Nombre:	getAlmacenUM () : UnidadMedidaEnum
Descripción:	Obtiene la unidad de medida de un producto en el almacén.
Nombre:	getServicioUM () : UnidadMedidaEnum
Descripción:	Obtiene la unidad de medida de un producto en un plato.
Nombre:	getCantAlm12 () : *Double*
Descripción:	Obtiene la cantidad existente de un producto en el almacén del complejo comedor 1.
Nombre:	getCantAlm14 () : *Double*
Descripción:	Obtiene la cantidad existente de un producto en el almacén del complejo comedor 2.
Nombre:	getCantAlm17 () : *Double*

Descripción:	Obtiene la cantidad existente de un producto en el almacén del complejo comedor 3.
Nombre:	getPrecioCompraProducto () : *Float*
Descripción:	Obtiene el precio de compra en el mercado de un producto.
Nombre:	setIdProducto (idProducto : *long*) : *Void*
Descripción:	Modifica el identificador de un producto por el especificado por parámetro.
Nombre:	setNombreProducto (nombreProducto : *String*) : *Void*
Descripción:	Modifica el nombre de un producto por el especificado por parámetro.
Nombre:	setTipoProducto (tipoProducto : TipoProductoEnum) : *Void*
Descripción:	Modifica el tipo de producto por el especificado por parámetro.
Nombre:	setEquivalencia (equivalencia : Double) : *Void*
Descripción:	Modifica el valor en unidad de medida de un producto en el almacén por el especificado por parámetro.
Nombre:	setAlmacenUM (almacenUM : UnidadMedidaEnum) : *Void*
Descripción:	Modifica la unidad de medida de un producto del almacén por el especificado por parámetro.
Nombre:	setServicioUM (servicioUM: UnidadMedidaEnum) : *void*
Descripción:	Modifica la unidad de medida de un producto de un plato por el especificado por parámetro.
Nombre:	setCantAlm12 (cantAlm12 : *Double*) : *Void*
Descripción:	Modifica la cantidad existente de un producto en el complejo comedor 1 por el especificado por parámetro.
Nombre:	setCantAlm14 (cantAlm14 : *Double*) : *Void*
Descripción:	Modifica la cantidad existente de un producto en el complejo comedor 2 por el especificado por parámetro.
Nombre:	setCantAlm17 (cantAlm17 : *Double*) : *Void*
Descripción:	Modifica la cantidad existente de un producto en el complejo comedor 3 por el especificado por parámetro.
Nombre:	setPrecioCompraProducto (precioCompraProducto : *Float*) : V*oid*
Descripción:	Modifica el precio de compra de un producto en el mercado por el especificado por parámetro.

Tabla 13 Descripción de la clase Plato

Descripción de la Clase *Plato*	
Nombre: Plato	
Tipo de clase: Modelo	
Atributos	**Tipos**
idPlato	*private*

nombPlato	*private*
norma	*private*
normaUM	*private*
kilocalorias	*private*
productos	*private*
Responsabilidades	
Nombre:	getIdPlato () : *long*
Descripción:	Obtiene el identificador de un plato.
Nombre:	setIdPlato (idPlato: *long*): *Void*
Descripción:	Modifica el identificador de un plato por el que se especifica por parámetro.
Nombre:	getNombPlato () : String
Descripción:	Obtiene el nombre de un plato.
Nombre:	setNombPlato (nombPlato: String): void
Descripción:	Modifica el nombre de un plato por el que se especifica por parámetro.
Nombre:	getNorma () : int
Descripción:	Obtiene el valor de la norma de un plato.
Nombre:	setNorma (norma: int): void
Descripción:	Modifica la norma de un plato por el que se especifica por parámetro.
Nombre:	getNormaUM (): UnidadMedidaEnum
Descripción:	Obtiene la unidad de medida de la norma a servir en un plato.
Nombre:	setNormaUM (precioPlato: Double) : void
Descripción:	Modifica la norma de un plato por el que se especifica por parámetro.
Nombre:	getKilocalorias () : Double
Descripción:	Obtiene las kilocalorías que aporta el plato.
Nombre:	setKilocalorias (kilocalorías: Double) void
Descripción:	Modifica las kilocalorías de un plato por el que se especifica por parámetro.
Nombre:	getProductos (): List<PlatoProducto>
Descripción:	Obtiene el listado de productos asignados al plato
Nombre:	setProductos (productos: List<PlatoProducto>): void
Descripción:	Modifica los productos asignados a un plato por el que se especifica por parámetro.
Nombre:	addProducto (producto: PlatoProducto): void
Descripción:	Adiciona un producto en el listado de productos asignados al plato.

2.9 Modelo de datos

Sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos o podríamos decir que es un conjunto de concepto que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos (42). A continuación, en la

Figura 6, se muestra el modelo de datos correspondiente a la solución propuesta.

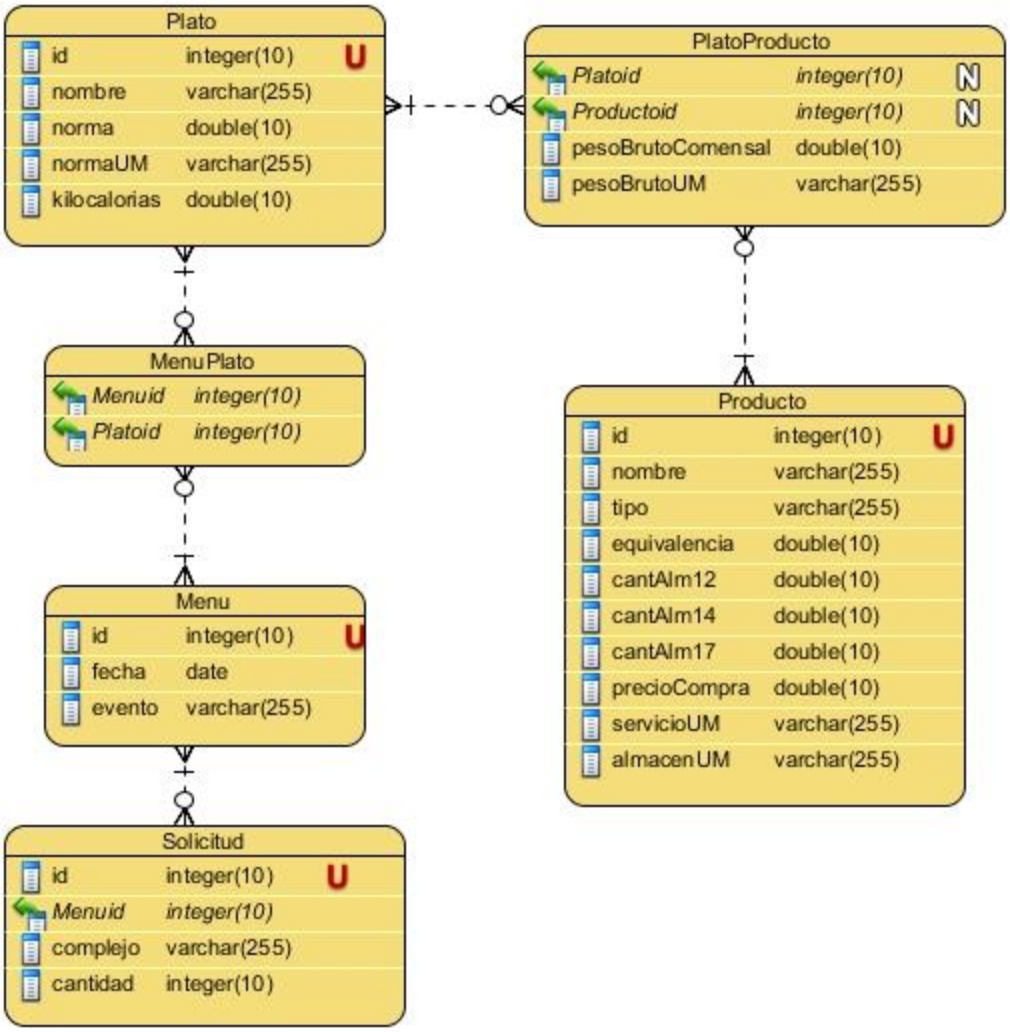


Figura 6. Modelo de datos. Elaboración propia.

2.10 Conclusiones parciales

Tras la definición de las características técnicas del sistema se arrojan las siguientes conclusiones:

- Se generaron artefactos acorde a la metodología de desarrollo de *software* (Modelo de procesos de negocios, Modelo de dominio, Diagrama de arquitectura, Diagrama de clases y el Modelo de datos) para formar bases para la implementación de la propuesta de solución.

- Se generaron las Historias de Usuarios para los requisitos funcionales, lo que permitirá construir un sistema que brindará solución a la problemática planteada.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

Laconcepción de la propuesta del sistema presentada en el capítulo anterior, ayuda al programador a entender las funcionalidades que busca el cliente y, por tanto, ser capaz de llevarlas a código interpretable por la computadora, o lo que es lo mismo implementar el sistema. Durante la fase de implementación y después de esta se desarrollan un conjunto de pruebas con el objetivo de asegurar la calidad de la aplicación y que esta ofrezca solución a los problemas planteados por el cliente.

3.2 Implementación

La implementación es un proceso de varias fases que comienza cuando se crea una aplicación en el equipo de un desarrollador y termina cuando está instalada y lista para ejecutarse en el equipo de un usuario. Describe cómo los elementos del modelo de diseño se implementan en términos de componentes, en otras palabras, toma el resultado del modelo de diseño para generar el código final del sistema. Dicho código está determinado por el lenguaje de programación y tiene como objetivo llevar a cabo la implementación de cada una de las clases significativas del diseño.

3.2.1 Estándares de codificación

Los estándares de codificación comprenden todos los aspectos de la generación de código. Estos son definidos por el equipo de desarrollo para lograr una estandarización en la programación del *software*. Engloban todos los aspectos relacionados con la generación de código, de tal manera que sea prudente, práctico y entendible para todos los programadores. Un código fuente en su totalidad debe reflejar un estilo armonioso, de forma tal que aparente ser realizado por un único programador. Si se logra la mantenibilidad del código, el sistema puede modificarse para añadir nuevas características, modificar las existentes, depurar errores, o mejorar el rendimiento(43). Un ejemplo de esto se puede apreciar en la Figura 7. Ejemplo de codificación del sistema. (Elaboración propia).

```
@RequestMapping(value = "/detallesPlato", method = RequestMethod.POST, produces = "application/json")
@ResponseBody
public Map<String, Object> detallesPlato(@RequestParam String idPlato) {

    Map<String, Object> response = new HashMap<String, Object>();
    Plato plato = null;
    plato= serviceFacade.obtenerPlatoById(Long.valueOf(idPlato));
    Prueba prueba=null;
    List<Prueba> pruebas = new ArrayList<Prueba>();
    List<PlatoProducto> productos=plato.getProductos();

    for (int i=0; i<productos.size(); i++) {
        prueba=new Prueba();
        prueba.setIdProductoo(productos.get(i).getProducto().getIdProducto());
        prueba.setNombreProducto(productos.get(i).getProducto().getNombProducto());
        prueba.setPesoBruto_UM(productos.get(i).getPesoBruto_UM());
        prueba.setPesoBrutoUnComensal(productos.get(i).getPesoBrutoUnComensal());

        pruebas.add(i,prueba);
    }

    if (plato != null){
        response.put("result", true);
        response.put("plato", plato);
        response.put("detalles",pruebas);
    }else{
        response.put("result", false);
        response.put("message", "No existe ese plato");
    }
    return response;
}
```

Figura 7 Ejemplo de codificación del sistema. Elaboración propia

3.2.2 Convenciones de nomenclatura

En la implementación del sistema se maneja la notación PascalCase para la nomenclatura de las clases e interfaces y la notación CamelCase para nombrar las variables y métodos presentes en dichas clases.

PascalCase: Plantea que los identificadores, nombres de variables, métodos y funciones que están compuestos por múltiples palabras juntas, iniciarán cada palabra con letra mayúscula. Ejemplo: ProductoController.java, ProductoDAO.java.

CamelCase: Plantea que los identificadores, nombres de variables, métodos y funciones que están compuestos por múltiples palabras juntas, iniciarán cada palabra con letra mayúscula, excepto la primera palabra. Ejemplo: setIdProducto(int idProducto), getNombProducto().

3.3 Diagrama de componentes

Un Diagrama de componentes muestra las dependencias lógicas entre componentes *software*, sean estos últimos fuentes, binarios o ejecutables, los cuales ilustran las piezas del *software*. Los Diagramas de componentes prevalecen en el campo de la arquitectura de *software* pero pueden ser usados para modelar y documentar cualquier arquitectura del sistema, es decir,

para describir la vista de implementación estática de un sistema.

Los Diagramas de componentes se relacionan con los diagramas de clases, ya que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones, pero un Diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clases. Usualmente un componente se implementa por una o varias clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema.(44) A continuación, en la Figura 8 se muestra un diagrama de componentes:

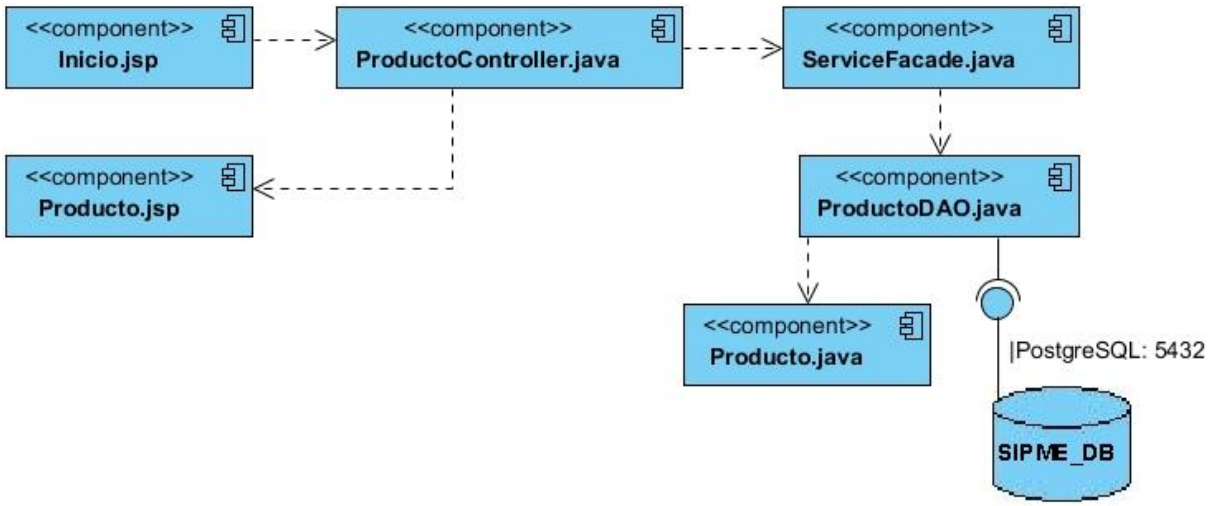


Figura 8. Diagrama de componentes. Elaboración propia.

3.4 Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar la disposición física de los componentes de hardware utilizados en la implementación del sistema y la relación entre cada uno de ellos (45).

En este diagrama se representan los elementos del sistema en tiempo de ejecución por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos de *software*. Los elementos de *software* podrán ejecutarse cumpliendo con las restricciones que aparecen en el diagrama de la Figura 9.

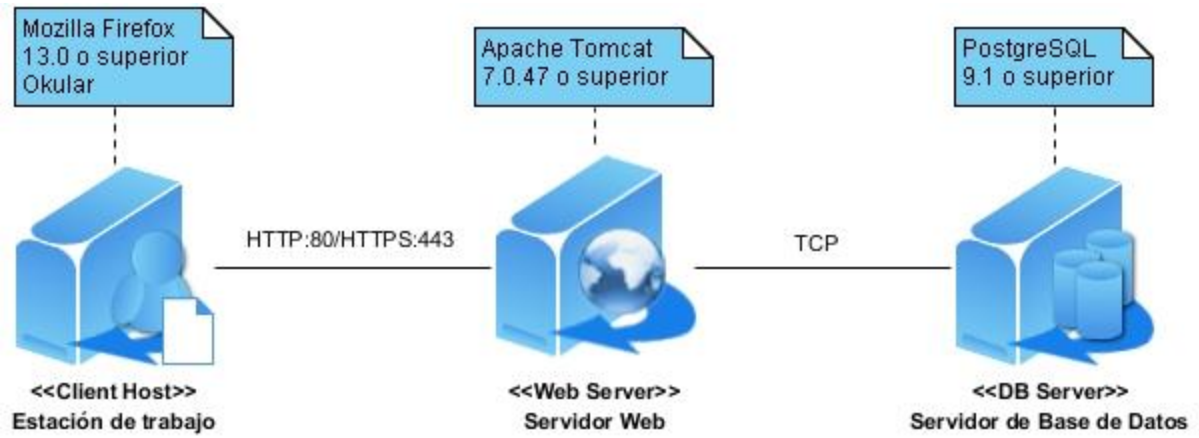


Figura 9. Diagrama de despliegue. Elaboración propia.

3.5 Pruebas al sistema

Las pruebas de software consisten en una serie de acciones en las que el sistema es ejecutado bajo condiciones y requisitos específicos. Los resultados son observados y registrados realizando una evaluación de la calidad del producto para ser entregado(46).

Luego de generado el código fuente, es necesario realizarle pruebas para detectar y corregir la mayor cantidad de errores posibles con el objetivo de brindar una solución con la calidad requerida. Existen diferentes tipos de pruebas, las cuales tienen como objetivosla detectar alguna anomalía en los resultados. En su mayoría están orientadas a comprobar las funcionalidades y la lógica del negocio, verificando que el sistema se comporte a la altura de las especificaciones que pide el cliente.

3.5.1 Pruebas de Caja negra

Para realizar las pruebas al sistema desarrollado se empleó el método de caja negra, método que se emplea a la interfaz del *software*. Las pruebas de caja negra se limitan a que el *tester*[11]pruebe con “datos” de entrada y estudie como salen, sin preocuparse de lo que ocurre en el interior (47). En general este método se centra en los requisitos funcionales del *software*. Dentro de los métodos de pruebas de caja negra se encuentran partición equivalente, el cual se utiliza para validar las funcionalidades del producto.

Partición Equivalente

Se define como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores,

[11]Persona encargada de probar el sistema.

reduciendo así el número total de casos de prueba que hay que desarrollar. Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (48).

Pruebas de rendimiento

Se define como un tipo de prueba para realizar una comparación de rendimiento entre un parámetro desconocido y uno conocido para establecer una línea comparativa. Cuenta con varios métodos de prueba para realizar las comparaciones como pueden ser: Benchmark, Contención, Carga, Performance profile; todas con enfoques diferentes pero encaminadas al mismo fin.

3.5.2 Pruebas de aceptación

El método de pruebas de aceptación se refiere a las pruebas llevadas a cabo por parte del cliente, labor fundamental para que valide el sistema, como último paso, previo a la puesta en explotación. Se debe insistir, principalmente, en los criterios de aceptación del sistema que sirven de base para asegurar que satisface los requisitos exigidos (49).

El ciclo de vida de las pruebas de aceptación es el siguiente:

- Diseñar casos de prueba de aceptación basados en los requerimientos que presenta el cliente.
- Preparar datos de las pruebas de aceptación.
- Validar datos de los casos de prueba de aceptación.
- Procedimiento de la prueba.
- Ejecutar las pruebas de aceptación para validar el análisis de requerimientos del cliente.
- Comparar los resultados de las pruebas con los casos de prueba iniciales (50).

A continuación se muestran los casos de pruebas realizados para las historias de usuario que fueron mostradas en el capítulo anterior, mostrando las restantes en los anexos del 24 al 42.

Tabla 14 Prueba de aceptación sobre la historia de usuario Listar producto

Caso de prueba	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Listar producto.	
Descripción: Prueba para la funcionalidad de gestionar los productos.	
Condiciones de ejecución: • El usuario debe estar autenticado.	

Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Gestionar producto".
Resultado esperado: El sistema debe mostrar un listado con los productos existentes en el almacén, además de las opciones correspondientes a cada producto. En caso de no existir productos almacenados, debe mostrar el mensaje "No hay productos para mostrar".
Evaluación de la Prueba: Satisfecho.

Tabla 15 Prueba de aceptación sobre la Historia de usuario Mostrar plato

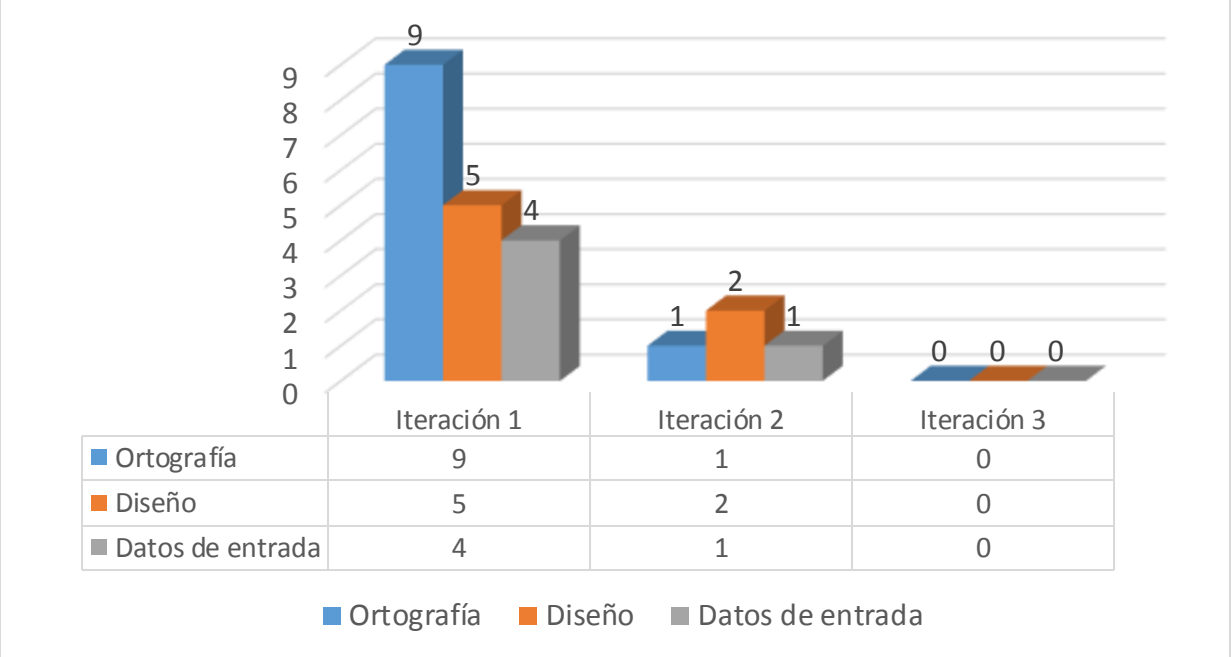
Caso de prueba	
Código: HU9_P9	**Historia de usuario: 9**
Nombre: Mostrar plato.	
Descripción: Prueba para la funcionalidad de mostrar plato.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un plato en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Mostrar detalles del plato".	
Resultado esperado: El sistema debe mostrar un formulario con los datos del plato ya registrado, los cuales no deben ser modificados por el usuario.	
Evaluación de la Prueba: Satisfecho.	

3.6 Resultados obtenidos

Como parte de las pruebas de aceptación realizadas en 3 iteraciones, fueron detectadas un total de 23 no conformidades, las cuales fueron agrupadas en 3 tipos: ortografía, diseño, datos de entrada, las que se muestran en la Figura 10. Una vez terminadas las pruebas de aceptación, las cuales arrojaron resultados satisfactorios, se verificó que el sistema desarrollado Sistema de Planificación de Menú cumple con las especificaciones del cliente. El sistema permite gestionar los productos existentes en el almacén de la universidad, cada uno de los platos que se pueden elaborar, las solicitudes de los comensales, así como la gestión de los posibles menús.

Así mismo y como resultado de las pruebas de rendimiento realizadas a la aplicación se definió un ambiente de pruebas estándar para un cliente que realice el uso de un navegador web Mozilla Firefox en una versión superior a la 13, sobre un Sistema Operativo (SO) que puede ser Windows o Linux, con un mínimo de 1GB de RAM. En correspondencia a esto depende el rendimiento del sistema, debido a la diferencia de carga que supone el uso de Linux como SO. Cabe resaltar que en un ambiente real las pruebas a realizar serían para un máximo de 9

usuarios, los que serían los directores de complejos comedores (3), los almaceneros (3), el especialista general de la DGA, el técnico y el Director General de la DGA como Administrador del sistema. Teniendo esto como base se decide realizar las pruebas de rendimiento para un número superior de usuarios, en específico 15 usuarios, para realizar la comparación con un ambiente en estrés, arrojando este como resultado un tiempo de respuesta crítico de 4.2800 milisegundos lo que representa un tiempo de respuesta de 4.3 segundos para una petición de un usuario, lo que representa una velocidad considerablemente rápida para 15 operaciones concurrentes como se muestra en la Figura 11.



	Iteración 1	Iteración 2	Iteración 3
Ortografía	9	1	0
Diseño	5	2	0
Datos de entrada	4	1	0

Figura 10. Relación de no conformidades detectadas en las pruebas.

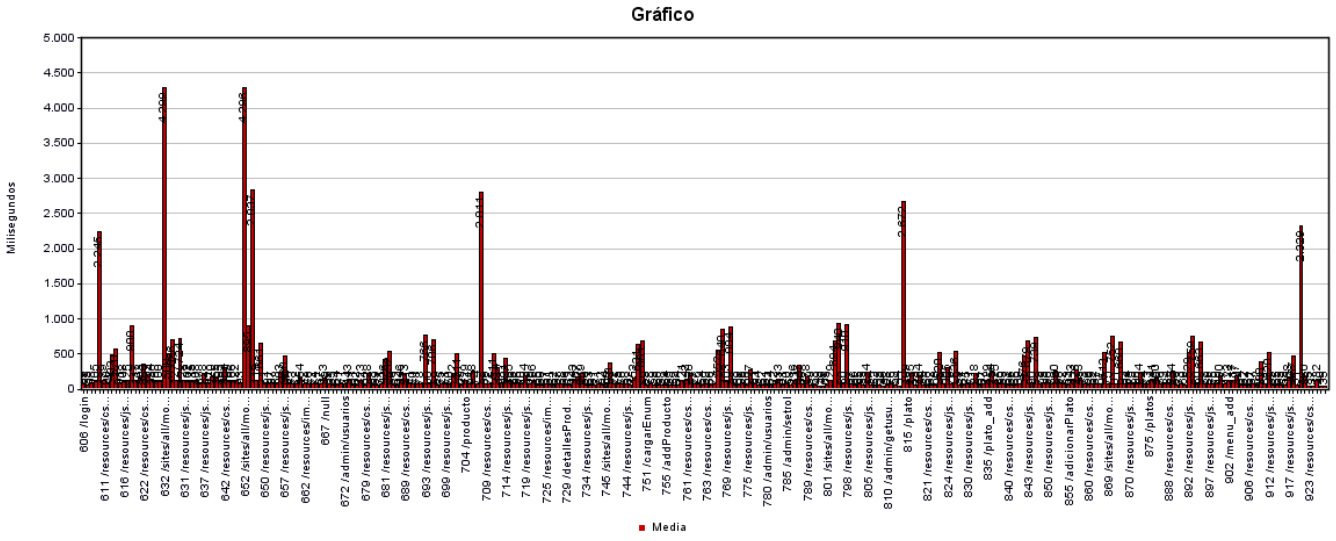


Figura 11. Tiempo de respuesta para 15 usuarios concurrentes.

3.7 Conclusiones parciales

Tras definir las características de implementación y pruebas del sistema propuesto como solución, se arrojan las siguientes conclusiones:

- Se realizaron los diagramas que propone la metodología de desarrollo a utilizar, para el modelado de la estructura del sistema.
- Se realizaron casos de pruebas para la rectificación de los errores posibles, para darle solución a estos en aras de liberar un producto con una mayor calidad.

CONCLUSIONES GENERALES

Una vez finalizado el trabajo se puede arribar a las siguientes conclusiones:

- Tras el estudio de las herramientas de planificación de menú existentes fue posible la definición de las funcionalidades necesarias a modelar por parte de la propuesta de solución.
- Las herramientas y tecnologías puestas en práctica permitieron el desarrollo claro y fluido de un sistema construido sobre bases sólidas y un entorno bien definido.
- Se desarrolló un sistema que sirve de apoyo a los trabajadores de la Dirección General de Alimentos de la UCI en la confecciónde l menú para el servicio de alimentación que brinda la universidad.

REFERENCIAS BIBLIOGRÁFICAS

1. MALBERNAT, L. R. *Tecnologías educativas e innovación de la Universidad.* 26 de diciembre de 2010, 2010, Disponible en: http://www.lacapitalmdp.com/noticias/La-Ciudad/2010/12/27/168009.htm.

2. DRAE. *Sistema.* 22 ed. 2001, Disponible en: http://lema.rae.es/drae/.

3. *Definición de sistema.* Disponible en: http://escritoriorural.educ.ar/actividades/definicion-de-sistema/.

4. ---. *Planificación.* 22 ed. 2001, Disponible en: http://lema.rae.es/drae/?val=planificación.

5. ABC, D. *Definición de planificación* Disponible en: http://www.definicionabc.com/general/planificacion.php.

6. DEFINICION.DE. 2015, nº Disponible en: http://definicion.de/menu-2/.

7. DIPRES. Sistema de Planificación y Control de Gestión. 2009, nº

8. ALEXANDER GONZÁLEZ DOMÍNGUEZ, G. C. M. *Diseño y desarrollo de una aplicación informática para la elaboración y evaluación de menús hospitalarios.* Perfil de Nutrición y Dietética. Facultad de Tecnología de la Salud. . Universidad de Ciencias Médicas de La Habana, 2009.

9. DANIA PÉREZ-ARMAYOR, E. O. L.-A., ARIEL RACET-VALDÉS, JOSÉ ANTONIO DÍAZ-BATISTA. Funcionalidades de Sistemas de Planificación de Recursos Empresariales para cadenas de Suministro. 2013, nº Disponible en: http://scielo.sld.cu/scielo.php?pid=S1815-59362013000200005&script=sci_arttext.

10. INGENIERÍA, A. Programa para la evaluación de dietas y gestión de datos de alimentación. 2008, nº Disponible en: https://www.alceingenieria.net/nutricion/dial.pdf.

11. Herramientas Tecnológicas para la atención. 2013, nº Disponible en: http://ammfen.org/memorias/28CongresoMazatlan%28XXVIII%29/documentos/ponencias/05.pdf.

12. ROQUE, E. G.-C. *Principios básicos de informática.* 2007. ISBN 978-84-9849-098-5.

13. *What is java?* [Consultado el: 3 de enero de 2015]. Disponible en: http://searchsoa.techtarget.com/definition/Java.

14. DISCLAIMER. *¿Qué es JavaScript?* Disponible en: http://www.efectosjavascript.com/javascript.html.

15. OLSON, S. D. *Ajax on Java.* revisada ed. 2007. ISBN 0596553641, 9780596553647.

16. CLIFTON, M. *What is A Franework?*Code Project: Disponible en: http://www.codeproject.com/Articles/5381/What-Is-A-Framework.

17. *Bootstrap la herramienta que facilita el desarrollo de una página web*. [Consultado el: 23

de enero de 2015]. Disponible en: http://www.ticweb.es/bootstrap-la-herramienta-que-facilita-el-desarrollo-de-una-pagina-web/.

18. MURPHEY, R. *Fundamentos de JQuery* Abril de 2013, [Consultado el: 20 de diciembre de 2014]. Disponible en: http://librojquery.com/.

19. ANDALUCÍA, J. D. *Marco de desarrollo de la Junta de Andalucía.* [Consultado el: 14 de enero de 2015]. Hibernate. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/97.

20. COMMUNITY., S. S. *SpringSource.org.* [Consultado el: 15 de enero de 2015]. Disponible en: http://www.springsource.org/.

21. MENTORING, G. *¿Qué es un IDE?* Disponible en: http://globalmentoring.com.mx/cursos-java/java-fundamentos/que-es-un-ide/.

22. FUNDACIÓN, T. E. *Eclipse. In.* 2014. vol. 2014,

23. RIVERA, H. C. *Sistemas gestores de base de datos (sgbd).* 15 de junio de 2012, Disponible en: http://es.slideshare.net/hcumbicusr/sistemas-gestores-de-base-de-datos-sgbd-13333545.

24. TOOLS, P. P. *PdAdmin PostgresSQL Tools* [Consultado el: 5 de diciembre de 2014]. Disponible en: http://www.pgadmin.org/.

25. FOUNDATION, A. S. *Apache Tomcat.* 2015.

26. BOOCH, G., RUMBAUGH, J Y JACOBSON, I. *El Lenguaje Unificado de Modelado.* México: Addison Wesley Iberoamericana ed. 1999, 1999.

27. PARADIGM, V. *UML herramienta CASE para el desarrollo de software* [Consultado el: 10 de enero de 2015]. Disponible en: http://www.visual-paradigm.com/product/vpuml/.

28. ESMERALDA VILLEGAS ZAMUDIO, A. V. *Metodologías de desarrollo de software.* Instituto superior de Apatzingán, 2010.

29. SÁNCHEZ, T. R. *Programa de Mejora. Metodología de desarrollo para la Actividad productiva en la UCI.* 12 de noviembre de 2014,

30. FOUNDATION, A. S. *Apache Maven Project* Disponible en: http://maven.apache.org/what-is-maven.html.

31. GARCERANT, I. *Modelo de dominio* 10 de julio de 2008, [Consultado el: 15 de marzo de 2015]. Disponible en: https://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

32. SOMMERVILLE, I. G., M.I.A. *Ingeniería del software.*Pearson Educación, 2005. ISBN 9788478290741.

33. COHN, M. *User Stories Applied.*2004. ISBN 0-321-20568-5.

34. QUINTERO, C. E. C. *Arquitectura de software dinámica basada en*

*reflexión.*Departamento de Informática. Universidad de Valladolid, 2002.

35. *Arquitectura de 3 capas.* Disponible en: http://www.buenastareas.com/ensayos/Arquitectura-De-3-Capas/3936241.html.

36. FIGUEROA, P. *Patrones* Disponible en: http://agamenon.uniandes.edu.co/~pfiguero/soo/Magister_Patrones/intropatrones.html.

37. LAGO, R. *Scribd* Arquitectura de software. Disponible en: http://www.proactivacalidad.com/java/patrones/index.html.

38. MESTRAS, J. P. *Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC). Programación Orientada a Objetos.* [Consultado el: 20 de febrero de 2015]. Disponible en: http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.mvc.pdf

39. LARMAN, C. *UML y Patrones.* 2da ed. 2004,

40. ORACLE. *Oracle Technology Network for Java Developers* [Consultado el: 20 de febrero de 2015]. Disponible en: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.

41. IBIBLIO.ORG. *Modelado de Sistemas con UML* Disponible en: http://www.ibiblio.org/pub/Linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.

42. BLOG, T. S. *Definición de un modelo de datos* 28 de Agosto de 2010, Disponible en: https://tombasededatos.wordpress.com/2010/08/28/2-1-definicion-de-un-modelo-de-datos/.

43. MICROSOFT. *Revisiones de código y estándares de codificación* Disponible en: https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

44. RAMÍREZ, E. A. Artefacto: Diagrama de componentes. 2009, nº p. 15.

45. PRESSMAN, R. *Ingeniería de software. Un enfoque práctico.* Editado por: Mcgraw-Hill. 2006. ISBN 970-10-5473-3.

46. IVAR JACOBSON GB, J. R. *El proceso Unificado de Desarrollo de Software.* 2000.

47. ADÁN, V. G. *Pruebas de caja negra.* 14 de Agosto de 2012, 2012, Disponible en: http://www.globetesting.com/2012/08/pruebas-de-caja-negra/.

48. UCI. *Ingeniería de Software II Conferencia 4: Flujo de trabajo de Prueba*. En 2011.

49. SYNEXIA. *Definición de las pruebas de aceptación del sistema* Disponible en: http://www.synexia.net/tecnologia/cvs/descriptivo/478/5101.

50. S, C. *Plan de Pruebas de Aceptación.* 2006, Disponible en: http://www.acis.org.co/fileadmin/Curso_Memorias/Curso_CMMI_Sept06/Modulo%205%20-%20Corporate%20Maturity%20/Ejemplos/Ejemplo%20Plan%20de%20Pruebas.doc.

ANEXOS

Anexo 1 Vale de solicitud

Universidad de las ciencias informaticas				SOLICITUD DE MATERIALES			D	M	A
Direccion UCI			Codigo:				1	4	15
Almacen al que solicitan: 12				Codigo:			Proceso: 515		
Destino	Orden de produccion No:		Lote:	Centro de costo: 4139			Evento		Comensales
				Area: Complejo # 1			Desayuno		1200
	Orden de trabajo:		Producto código:		Otros:				

Codigo	Descripcion:	U/M	Cantidad
	Leche en polvo	Kg	27.6
	Azúcar refino	Kg	29.52
	Sal	Kg	0.36
	Café	Kg	11.16
	Queso frescal	Kg	48

Solicitado:				Autorizado:				Recibido:				Folio.
Nombre: Aylenis Herrera				Nombre: Nilberto Chávez				Nombre:				
	D	M	A		D	M	A		D	M	A	
Firma	1	4	15	Firma	1	4	15	Firma	1	4	15	

Anexo 2 Carta técnica

MODELO 11-11

Dirección General De Alimentos

Complejo: 1 — Eventos: Desayuno

No de Comensales: 1200 rac

Para el día 1 de Abril del 2015

A servir	Menú	Alimentos e Ingredientes	Norma (g)	Rac.	Cantidad	U	A solicitar	Real
232 mL	Leche	Leche en polvo	23	1200	27.60	Kg	Leche en polvo	27.6
		Azúcar refino	21	1200	25.20	Kg	Azúcar refino	29.52
		Sal	0.3	1200	0.36	Kg	Sal	0.36
		Agua	190	1200	228.00	lt	Café	11.16
							Queso frescal	48
20 ml	Café	Café	9.3	1200	11.16	kg		
		Azúcar refino	3.6	1200	4.32	kg		
		Agua	32	1200	38.40	lt		
1 u	Pan c/ Queso	Pan (80 g)	1	1200	1200	U		
		Queso frescal	40	1200	48.00	Kg		

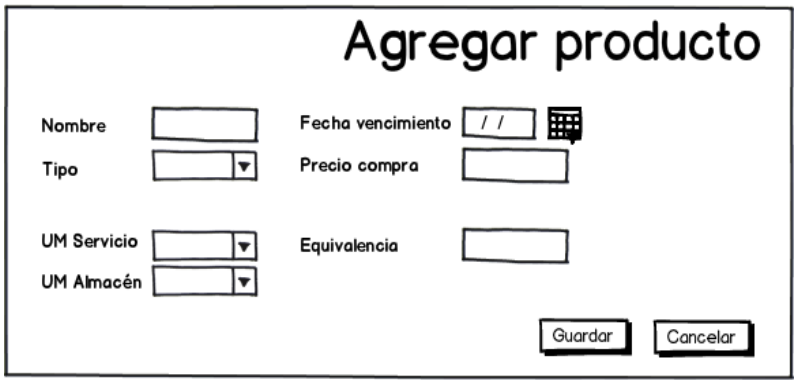
Confeccionado por: Aylenis Herrera — Aprobado por: Nilberto C. Chavez — Folio

Revisado por: — J. de turno: — Técnica en alimentos:

Transportador: — Reciben: Lunch....

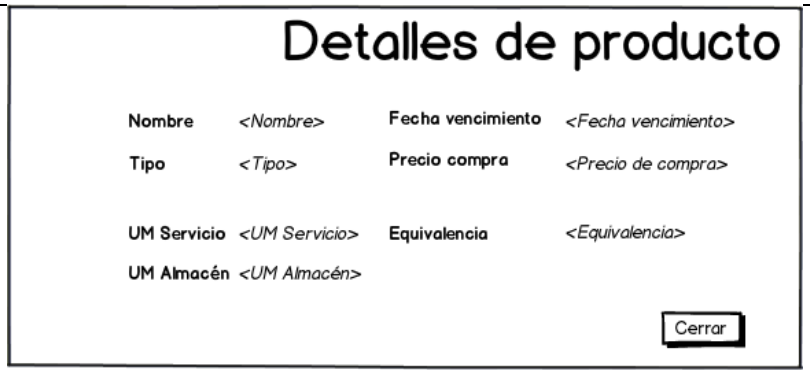
Anexo 3 Historia de usuario "Adicionar producto"

Historia de usuario	
Número 2	**Nombre de la Historia de Usuario:** Adicionar producto
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Agregar producto", se muestre un formulario en el que se introducirán los datos necesarios para la creación de un nuevo producto. Se mostrarán las opciones "Guardar" y "Cancelar".	
Observaciones:NA	
Prototipos de interfaces:	



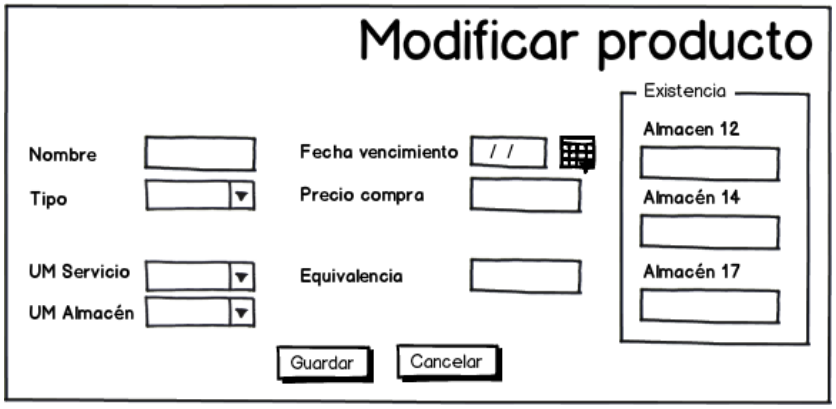
Anexo 4 Historia de usuario "Mostrar producto"

Historia de usuario	
Número 3	**Nombre de la Historia de Usuario:** Mostrar producto
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Mostrar detalles del producto" correspondiente a un producto determinado, se muestre un formulario con los datos del producto ya registrado, de forma no editable, para la consulta de los mismos. Se mostrará la opción "Cerrar".	
Observaciones:NA	
Prototipos de interfaces:	



Anexo 5 Historia de usuario "Modificar producto"

Historia de usuario	
Número 4	**Nombre de la Historia de Usuario:** Modificar producto
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Modificar producto" correspondiente a un producto determinado, se muestre un formulario con los datos del producto ya registrado de forma editable para la modificación de los mismos. Se mostrarán las opciones "Guardar" y "Cancelar".	
Observaciones:NA	
Prototipos de interfaces:	



Anexo 6 Historia de usuario "Filtrar producto"

Historia de usuario	
Número 5	**Nombre de la Historia de Usuario:** Filtrar producto
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema debe permitir filtrar los productos obtenidos a partir del criterio de búsqueda introducido.	
Observaciones: El criterio de búsqueda será verificado en cada uno de los atributos del producto y el filtrado se realizará a medida que se introduzca el criterio de búsqueda.	
Prototipos de interfaces:	

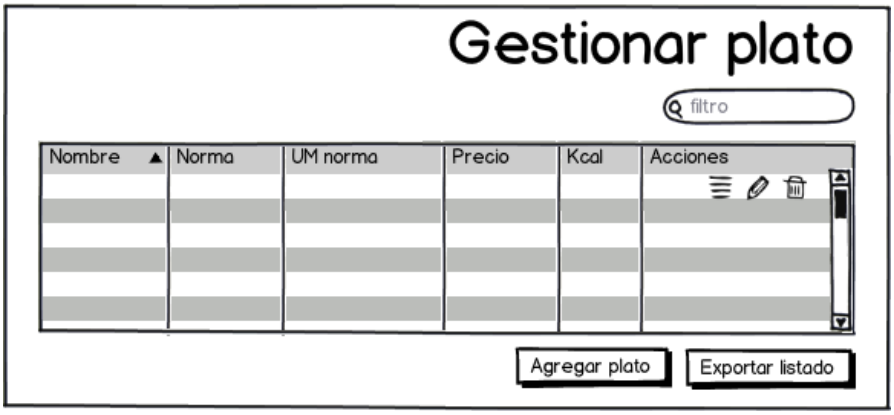
Anexo 7 Historia de usuario "Asignar rol"

Historia de usuario	
Número 6	**Nombre de la Historia de Usuario:** Asignar rol
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada: 1**
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema debe permitir que al hacer clic en la opción "Roles", se muestra un listado con todos los usuarios en la base de datos con roles especificados, además muestra un enlace que permite agregar usuarios desde el servidor de direcciones LDAP de la UCI, para incluir este como un nuevo usuario con roles en el sistema.	
Observaciones: Será asignado el rol al usuario adicionado.	
Prototipos de interfaces:	

Anexo 8 Historia de usuario "Listar plato"

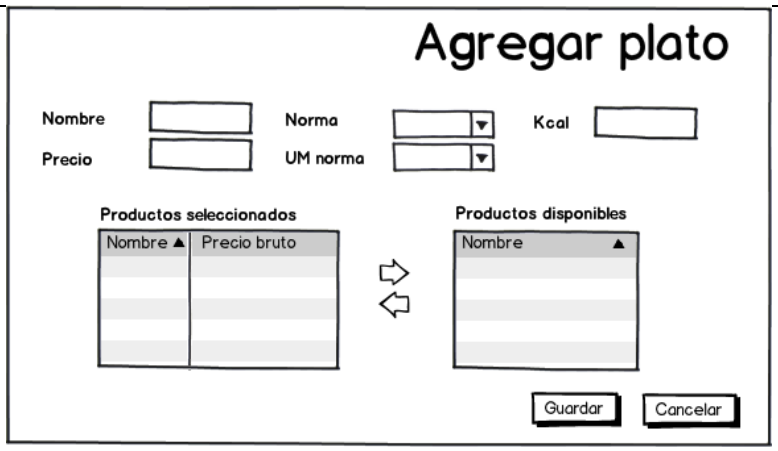
Historia de usuario	
Número 7	**Nombre de la Historia de Usuario:** Listar plato
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema debe permitir que al hacer clic en la opción "Gestionar plato", se muestre un listado con los platos existentes. Para cada plato mostrado se habilitarán las opciones "Detalles del plato" y "Modificar plato". Adicionalmente se muestran las opciones	

"Agregar plato" y "Exportar listado de platos". De igual forma se mostrará un campo que permitirá introducir un criterio de búsqueda para filtrar los platos listados.
Observaciones:NA
Prototipos de interfaces:

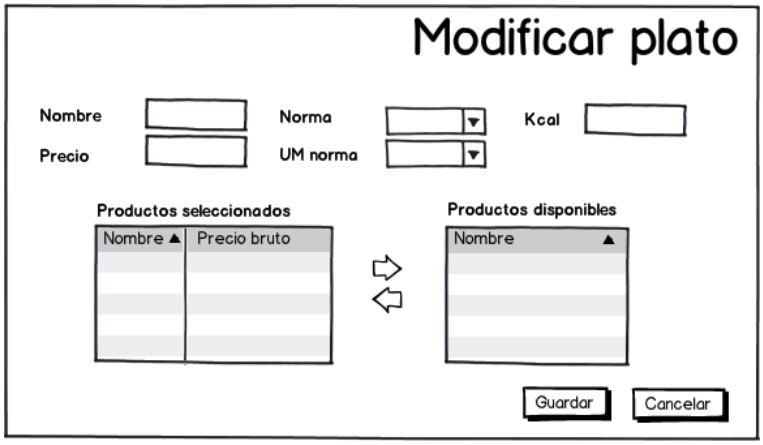


Anexo 9 Historia de usuario "Adicionar plato"

Historia de usuario	
Número 8	**Nombre de la Historia de Usuario:** Adicionarplato
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Agregar plato", se muestre un formulario en el que se introducirán los datos necesarios para la creación de un nuevo plato. Se mostrarán las opciones "Guardar" y "Cancelar".	
Observaciones: Los productos que conformarán el plato, serán identificados del listado de productos mostrado y una vez seleccionados, se indicará el peso bruto de los mismos para un comensal. El precio del plato se determinará automáticamente a partir de la suma de los precios de cada producto. El precio de un producto se determinará a partir de la siguiente fórmula: (precio de compra de un producto * 100) / (1000* peso bruto del producto).	
Prototipos de interfaces:	


Agregar plato
Nombre
Norma
Kcal
Precio
UM norma
Productos seleccionados
Nombre
Precio bruto
Productos disponibles
Nombre
Guardar
Cancelar

Anexo 10 Historia de usuario "Modificar plato"

Historia de usuario	
Número 10	**Nombre de la Historia de Usuario:** Modificar plato
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Modificar plato" correspondiente a un plato determinado, se muestre un formulario con los datos del plato ya registrado de forma editable para la modificación de los mismos. Se mostrarán las opciones "Guardar" y "Cancelar".	
Observaciones: Los productos que conformarán el plato, serán identificados del listado de productos mostrado y una vez seleccionados, se indicará el peso bruto de los mismos para un comensal. El precio del plato se determinará automáticamente a partir de la suma de los precios de cada producto. El precio de un producto se determinará a partir de la siguiente fórmula: (precio de compra de un producto * 100) / (1000* peso bruto del producto).	
Prototipos de interfaces: 	

Anexo 11 Historia de usuario "Filtrar plato"

Historia de usuario	
Número 11	**Nombre de la Historia de Usuario:** Filtrar plato
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema debe permitir filtrar los platos obtenidos a partir del criterio de búsqueda introducido.	

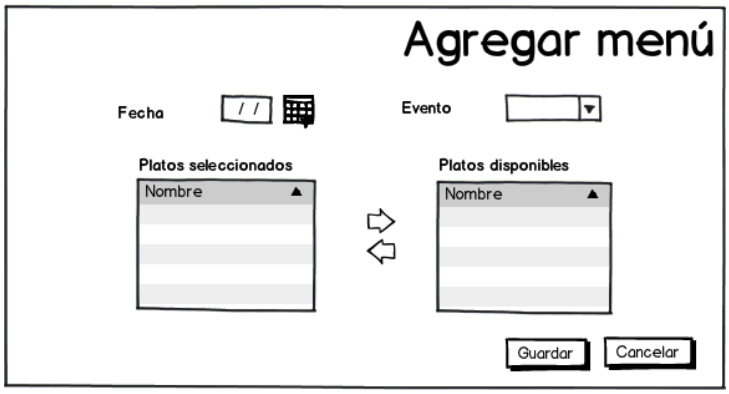
Observaciones: El criterio de búsqueda será verificado en cada uno de los atributos del plato y el filtrado se realizará a medida que se introduzca el criterio de búsqueda.
Prototipos de interfaces:

Anexo 12 Historia de usuario "Listar menú"

Historia de usuario	
Número 12	**Nombre de la Historia de Usuario:** Listar menú
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema debe permitir que al hacer clic en la opción "Gestionar menú", se muestre un listado con los menús existentes. Para cada menú mostrado se habilitará la opción "Detalles del menú". Adicionalmente se muestran las opciones "Agregar menú" y "Exportar listado de menús". De igual forma se mostrará un campo que permitirá introducir un criterio de búsqueda para filtrar los menús listados.	
Observaciones: Se habilitará una opción "Modificar menú" para aquellos menús cuya fecha de planificación sea mayor a la fecha actual.	
Prototipos de interfaces: 	

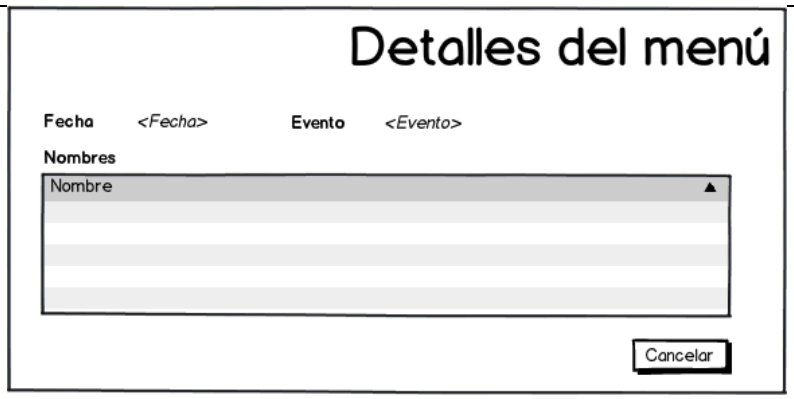
Anexo 13 Historia de usuario "Adicionar menú"

Historia de usuario	
Número 13	**Nombre de la Historia de Usuario:** Adicionarmenú
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Agregar menú", se muestre	

un formulario en el que se introducirán los datos necesarios para la creación de un nuevo menú. Se mostrarán las opciones “Guardar” y “Cancelar”.
Observaciones: Los platos que conformarán el menú, serán identificados del listado de platos disponibles. Las posibles fechas de planificación serán los días de lunes a domingo de la próxima semana. Los eventos que podrán ser planificados dependerán de la fecha que se seleccione, ya que para cada fecha solo será posible planificar un único menú para los eventos desayuno, almuerzo y comida; el evento merienda siempre estará disponible.
Prototipos de interfaces: 

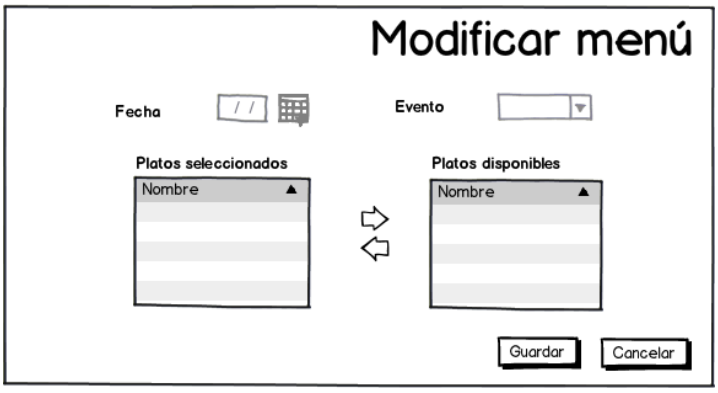
Anexo 14 Historia de usuario “Mostrar menú”

Historia de usuario	
Número 14	**Nombre de la Historia de Usuario:** Mostrar menú
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción “Mostrar detalles del menú” correspondiente a un menú determinado, se muestre un formulario con los datos del menú ya registrado, de forma no editable, para la consulta de los mismos. Se mostrará la opción “Cerrar”.	
Observaciones: NA	
Prototipos de interfaces:	



Anexo 15 Historia de usuario "Modificar menú"

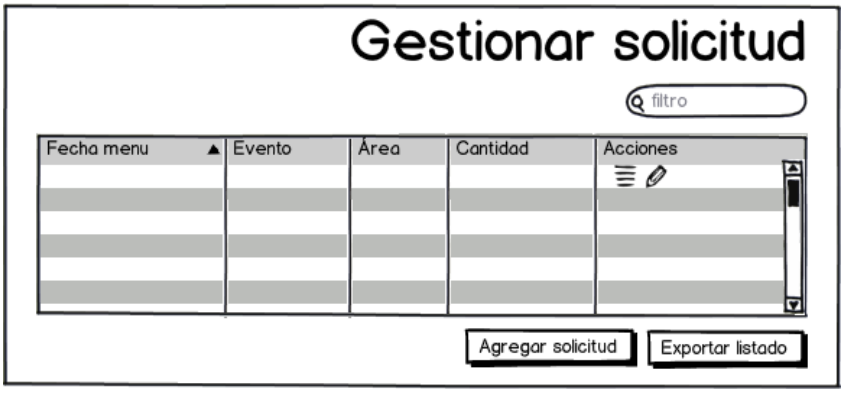
Historia de usuario	
Número 15	**Nombre de la Historia de Usuario:** Modificarmenú
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Modificar menú" correspondiente a un menú determinado, se muestre un formulario con los datos del menú ya registrado de forma editable para la modificación de los mismos. Se mostrarán las opciones "Guardar" y "Cancelar".	
Observaciones: Los campos fecha y evento no serán editables, solo podrá ser modificado el listado de platos asignados al menú. Los nuevos platos que conformarán el menú, serán identificados del listado de platos disponibles.	
Prototipos de interfaces:	



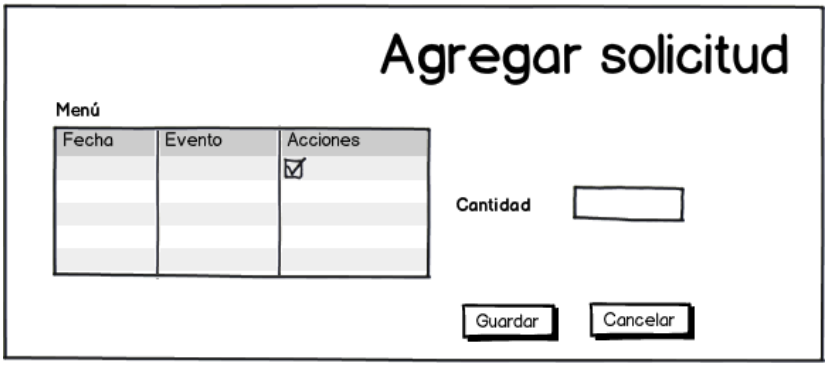
Anexo 16 Historia de usuario "Filtrar menú"

Historia de usuario	
Número 16	**Nombre de la Historia de Usuario:** Filtrar menú
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Media	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema debe permitir filtrar los menús obtenidos a partir del criterio de búsqueda introducido.	
Observaciones: El criterio de búsqueda será verificado en cada uno de los atributos del menú y el filtrado se realizará a medida que se introduzca el criterio de búsqueda.	
Prototipos de interfaces:	

Anexo 17 Historia de usuario "Listar solicitud"

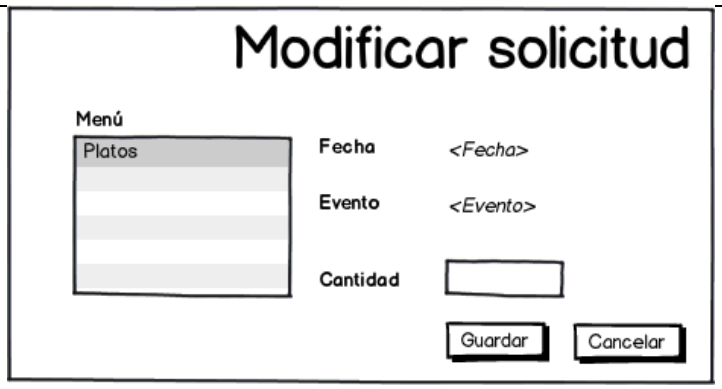
Historia de usuario	
Número 17	**Nombre de la Historia de Usuario:** Listar solicitud
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Baja	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema debe permitir que al hacer clic en la opción "Gestionar solicitud", se muestre un listado con las solicitudes realizadas. Para cada solicitud se muestran las opciones "Mostrar detalles de la solicitud", "Generar carta técnica" y "Generar vale de solicitud". Adicionalmente se muestran la opción "Ingresar solicitud".	
Observaciones: Las solicitudes que se listarán serán aquellas correspondientes a las áreas que son asignadas al usuario autenticado. Se habilitará una opción "Modificar solicitud" para aquellas solicitudes cuya fecha del menú sea mayor a la fecha actual.	
Prototipos de interfaces: 	

Anexo 18 Historia de usuario "Ingresar solicitud"

Historia de usuario	
Número 18	**Nombre de la Historia de Usuario:** Ingresarsolicitud
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Arisney Figueredo Ramos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Ingresar solicitud", se muestre un formulario en el que se introducirán los datos necesarios para la creación de una nueva solicitud. Se mostrarán las opciones "Guardar" y "Cancelar".	
Observaciones: El menú para el cual se realiza la solicitud será identificado del listado de menús sin solicitudes, una vez seleccionado se mostrará el listado de platos correspondientes al mismo. El campo área contendrá solo las áreas a las que fue asignado el usuario autenticado.	
Prototipos de interfaces: 	

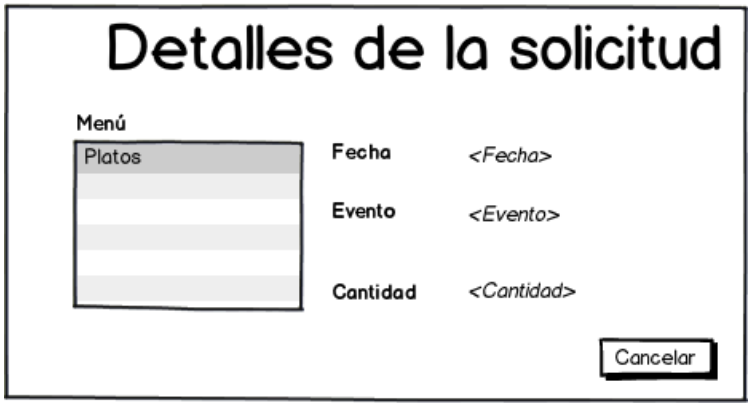
Anexo 19 Historia de usuario "Modificar solicitud"

Historia de usuario	
Número 19	**Nombre de la Historia de Usuario:** Modificarsolicitud
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Modificar solicitud" correspondiente a un menú determinado, se muestren en un formulario los datos del menú de forma no editable y de los datos de la solicitudsolo se mostrará de forma editable el campo cantidad. Se mostrarán las opciones "Guardar" y "Cancelar".	
Observaciones: NA	
Prototipos de interfaces:	



Anexo 20 Historia de usuario "Mostrar solicitud"

Historia de usuario	
Número 20	**Nombre de la Historia de Usuario:** Mostrar solicitud
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción "Detalles de la solicitud" correspondiente a un menú determinado, se muestre un formulario con los datos del menú y los datos de la solicitud, todos de forma no editable, para la consulta de los mismos. Se mostrará la opción "Cerrar".	
Observaciones: NA	
Prototipos de interfaces:	



Anexo 21 Historia de usuario "Generar carta técnica"

Historia de usuario	
Número 21	**Nombre de la Historia de Usuario:** Generar carta técnica
Cantidad de modificaciones a la Historia de Usuario: Ninguna	

Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción “Generar carta técnica” correspondiente a una solicitud determinada, realizada a un menú, se cree un documento PDF con los datos válidos de la carta técnica.	
Observaciones: NA	
Prototipos de interfaces:	

Anexo 22 Historia de usuario "Generar vale de solicitud"

Historia de usuario	
Número 22	**Nombre de la Historia de Usuario:** Generar vale de solicitud
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción “Generar vale de solicitud” correspondiente a una solicitud determinada, realizada a un menú, se cree un documento PDF con los datos válidos del vale de solicitud.	
Observaciones: NA	
Prototipos de interfaces:	

Anexo 23 Historia de usuario "Modificar rol asignado"

Historia de usuario	
Número 23	**Nombre de la Historia de Usuario:** Modificar rol asignado
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: ArisneyFigueredoRamos	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:**
Riesgo en desarrollo: Alto	**Puntos reales:**
Descripción: El sistema permitirá que al hacer clic en la opción “Roles” modificar el rol o los roles asignados a los usuarios con acceso al sistema.	
Observaciones: NA	
Prototipos de interfaces:	

Anexo 24 Prueba de aceptación sobre la historia de usuario Adicionar producto

Caso de prueba	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Adicionar producto.	
Descripción: Prueba para la funcionalidad de adicionar un producto.	
Condiciones de ejecución: • El usuario debe estar autenticado.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Agregar producto".	
Resultado esperado: El sistema debe mostrar un formulario en el que se introducirán los datos necesarios para la creación de un nuevo producto. Se mostrarán las opciones "Guardar" y "Cancelar". El hacer clic en la opción "Guardar" se debe guardar el producto.	
Evaluación de la Prueba: Satisfecho.	

Anexo 25 Prueba de aceptación sobre la Historia de usuario Mostrar producto

Caso de prueba	
Código: HU3_P3	**Historia de usuario: 3**
Nombre: Mostrar producto.	
Descripción: Prueba para la funcionalidad de mostrar detalles de un producto.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un producto en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Mostrar detalles del producto".	
Resultado esperado: El sistema debe mostrar un formulario con los datos del producto ya registrado, los cuales no deben ser modificados por el usuario.	
Evaluación de la Prueba: Satisfecho.	

Anexo 26 Prueba de aceptación sobre la Historia de usuario Modificar producto

Caso de prueba	
Código: HU4_P4	**Historia de usuario: 4**
Nombre: Modificar producto.	
Descripción: Prueba para la funcionalidad de modificarun producto.	
Condiciones de ejecución: • El usuario debe estar autenticado.	

• Debe existir al menos un producto en el sistema.
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción “Modificar los datos del producto”.
Resultado esperado: El sistema debe mostrar un formulario con los datos del producto ya registrado, los cuales deben ser modificados por el usuario. Al hacer clic en la opción “Guardar”, el producto debe ser actualizado en la BD.
Evaluación de la Prueba: Satisfecho.

Anexo 27 Prueba de aceptación sobre la Historia de usuario Filtrar producto

Caso de prueba	
Código: HU5_P5	**Historia de usuario: 5**
Nombre: Filtrar producto.	
Descripción: Prueba para la funcionalidad de filtrar un producto por un criterio específico.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un producto en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe introducir el criterio de búsqueda en la opción “Buscar producto”.	
Resultado esperado: El sistema debe mostrar el o los productos que coincidan con el criterio de búsqueda.	
Evaluación de la Prueba: Satisfecho.	

Anexo 28 Prueba de aceptación sobre la Historia de usuario Asignar rol

Caso de prueba	
Código: HU6_P6	**Historia de usuario: 6**
Nombre: Asignar rol.	
Descripción: Prueba para la funcionalidad para asignar rol.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un usuario en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción “Rol”.	
Resultado esperado: Se debe mostrar un listado con los usuarios en el sistema con roles, a partir de ahí el usuario puede asignar nuevos roles a los usuarios del sistema.	
Evaluación de la Prueba: Satisfecho.	

Anexo 29 Prueba de aceptación sobre la historia de usuario Listar plato

Caso de prueba	
Código: HU7_P7	**Historia de usuario:** 7
Nombre: Listar plato.	
Descripción: Prueba para la funcionalidad de listar los platos.	
Condiciones de ejecución: • El usuario debe estar autenticado.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Gestionar plato".	
Resultado esperado: El sistema debe mostrar un listado con los platos existentes, además de las opciones correspondientes a cada plato. En caso de no existir debe mostrar el mensaje "No hay platos para mostrar".	
Evaluación de la Prueba: Satisfecho.	

Anexo 30 Prueba de aceptación sobre la historia de usuario Adicionar plato

Caso de prueba	
Código: HU8_P8	**Historia de usuario: 8**
Nombre: Adicionar plato.	
Descripción: Prueba para la funcionalidad de adicionar un plato.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un producto registrado	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Agregar plato".	
Resultado esperado: El sistema debe mostrar un formulario en el que se introducirán los datos necesarios para la creación de un nuevo plato. Se mostrarán las opciones "Guardar" y "Cancelar". El hacer clic en la opción "Guardar" se debe guardar el plato.	
Evaluación de la Prueba: Satisfecho.	

Anexo 31 Prueba de aceptación sobre la Historia de usuario Modificar plato

Caso de prueba	
Código: HU10_P10	**Historia de usuario:** 10
Nombre: Modificar plato.	
Descripción: Prueba para la funcionalidad de modificar un plato.	
Condiciones de ejecución:	

• El usuario debe estar autenticado. • Debe existir al menos un plato en el sistema.
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción “Modificar los datos del plato”.
Resultado esperado: El sistema debe mostrar un formulario con los datos del plato ya registrado, los cuales deben ser modificados por el usuario. Al hacer clic en la opción “Guardar”, el plato debe ser actualizado en la BD.
Evaluación de la Prueba: Satisfecho.

Anexo 32 Prueba de aceptación sobre la Historia de usuario Filtrar plato

Caso de prueba	
Código: HU11_P11	**Historia de usuario:** 11
Nombre: Filtrar plato.	
Descripción: Prueba para la funcionalidad de filtrar un plato por un criterio específico.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un plato en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe introducir el criterio de búsqueda en la opción “Buscar plato”.	
Resultado esperado: El sistema debe mostrar el o los platos que coincidan con el criterio de búsqueda.	
Evaluación de la Prueba: Satisfecho.	

Anexo 33Prueba de aceptación sobre la Historia de usuario Modificar roles asignados

Caso de prueba	
Código: HU23_P23	**Historia de usuario:** 23
Nombre: Modificar roles asignados.	
Descripción: Prueba para la funcionalidad exportar el listado de platos existentes.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un usuario en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción “Rol”.	
Resultado esperado: Se debe mostrar un listado con los usuarios en el sistema con roles definidos desde donde se puede modificar los roles de los mismos.	
Evaluación de la Prueba: Satisfecho.	

Anexo 34 Prueba de aceptación sobre la Historia de usuario Listar menú

Caso de prueba	
Código: HU13_P13	**Historia de usuario:** 13
Nombre: Listarmenú.	
Descripción: Prueba para la funcionalidad de listar los menús.	
Condiciones de ejecución: • El usuario debe estar autenticado.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Gestionar menú".	
Resultado esperado: El sistema debe mostrar un listado con los menús existentes, además de las opciones correspondientes a cada menú. En caso de no existir debe mostrar el mensaje "No hay menú para mostrar".	
Evaluación de la Prueba: Satisfecho.	

Anexo 35 Prueba de aceptación sobre la historia de usuario Adicionar menú

Caso de prueba	
Código: HU14_P14	**Historia de usuario:** 14
Nombre: Adicionar menú.	
Descripción: Prueba para la funcionalidad de adicionar un menú.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un plato registrado.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Agregar menú".	
Resultado esperado: El sistema debe mostrar un formulario en el que se introducirán los datos necesarios para la creación de un nuevo menú. Se mostrarán las opciones "Guardar" y "Cancelar". El hacer clic en la opción "Guardar" se debe guardar el plato.	
Evaluación de la Prueba: Satisfecho.	

Anexo 36 Prueba de aceptación sobre la Historia de usuario Mostrar menú.

Caso de prueba	
Código: HU15_P15	**Historia de usuario:** 15
Nombre: Mostrar menú.	
Descripción: Prueba para la funcionalidad de mostrar menú.	
Condiciones de ejecución: • El usuario debe estar autenticado.	

• Debe existir al menos un menú en el sistema.
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Mostrar detalles del menú".
Resultado esperado: El sistema debe mostrar un formulario con los datos del menú ya registrado, los cuales no deben ser modificados por el usuario.
Evaluación de la Prueba: Satisfecho.

Anexo 37 Prueba de aceptación sobre la Historia de usuario Modificar menú

Caso de prueba	
Código: HU16_P16	**Historia de usuario:** 16
Nombre: Modificar menú.	
Descripción: Prueba para la funcionalidad de modificar un menú.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un menú en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Modificar los datos del menú".	
Resultado esperado: El sistema debe mostrar un formulario con los datos del menú ya registrado, los cuales deben ser modificados por el usuario. Al hacer clic en la opción "Guardar", el menú debe ser actualizado en la BD.	
Evaluación de la Prueba: Satisfecho.	

Anexo 38 Prueba de aceptación sobre la Historia de usuario Filtrar menú

Caso de prueba	
Código: HU17_P17	**Historia de usuario:** 17
Nombre: Filtrar plato.	
Descripción: Prueba para la funcionalidad de filtrar un menú por un criterio específico.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un menú en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe introducir el criterio de búsqueda en la opción "Buscar menú".	
Resultado esperado: El sistema debe mostrar el o los menús que coincidan con el criterio de búsqueda.	
Evaluación de la Prueba: Satisfecho.	

Anexo 39 Prueba de aceptación sobre la Historia de usuario Listar solicitud

Caso de prueba	
Código: HU19_P19	**Historia de usuario:** 19
Nombre: Listar solicitud.	
Descripción: Prueba para la funcionalidad de listar las solicitudes.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos una solicitud.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Gestionar solicitud".	
Resultado esperado: El sistema debe mostrar un listado con las solicitudes existentes, además de las opciones correspondientes a cada solicitud. En caso de no existir debe mostrar el mensaje "No hay solicitudes para mostrar".	
Evaluación de la Prueba: Satisfecho.	

Anexo 40 Prueba de aceptación sobre la historia de usuario Ingresar solicitud

Caso de prueba	
Código: HU20_P20	**Historia de usuario:** 20
Nombre: Ingresar solicitud.	
Descripción: Prueba para la funcionalidad de ingresar una solicitud.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos un menú registrado.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Ingresar solicitud".	
Resultado esperado: El sistema debe mostrar un formulario en el que se introducirán los datos necesarios para la realización de una solicitud a un menú determinado. Se mostrarán las opciones "Guardar" y "Cancelar". El hacer clic en la opción "Guardar" se debe guardar la solicitud.	
Evaluación de la Prueba: Satisfecho.	

Anexo 41 Prueba de aceptación sobre la Historia de usuario Modificar solicitud

Caso de prueba	
Código: HU21_P21	**Historia de usuario:** 21
Nombre: Modificar solicitud.	

Descripción: Prueba para la funcionalidad de modificar una solicitud.
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos una solicitud en el sistema.
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Modificar los datos de la solicitud".
Resultado esperado: El sistema mostrará un formulario los datos del menú de forma no editable y de los datos de la solicitud solo se mostrará de forma editable el campo cantidad. Al hacer clic en la opción "Guardar", la solicitud debe ser actualizada en la BD.
Evaluación de la Prueba: Satisfecho.

Anexo 42 Prueba de aceptación sobre la Historia de usuario Mostrar solicitud

Caso de prueba	
Código: HU22_P22	**Historia de usuario:** 22
Nombre: Mostrar solicitud.	
Descripción: Prueba para la funcionalidad de mostrar menú.	
Condiciones de ejecución: • El usuario debe estar autenticado. • Debe existir al menos una solicitud en el sistema.	
Entradas/Pasos de Ejecución: Una vez autenticado el usuario debe dar clic en la opción "Mostrar detalles de la solicitud".	
Resultado esperado: El sistema debe mostrar un formulario con los datos de la solicitud ya registrada, los cuales no deben ser modificados por el usuario.	
Evaluación de la Prueba: Satisfecho.	