



Universidad de las Ciencias Informáticas

Facultad 2

Módulo Abastecimiento para el Sistema Integrado de
Alimentación de la Universidad de las Ciencias Informáticas.

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

Autor: Leidy Márquez Barbosa

Tutores: Ing. Kenia González Quintana

Ing. Dairo Roberto Gil Martín

Co-tutor: Lic. Diana Aquino Ruiz

Ciudad de La Habana, 2019

“Año 60 de la Revolución”



*Para el logro del triunfo siempre ha sido indispensable pasar por
la senda de los sacrificios.*

Simón Bolívar

AGRADECIMIENTOS

Hoy después de un largo recorrido este sueño se convierte en realidad. Quisiera comenzar agradeciendo a la persona más especial en mi vida, a mi madre, por todos los sacrificios realizados desde que me dio la vida, por apoyarme en cada decisión, por llamarme la atención cuando no he actuado correctamente, por ser madre, pero ante todo por ser amiga. La vida no me alcanzará para pagarle todo lo que ha hecho por mí y en ocasiones, aunque no se lo demuestre la amo más que a mí misma.

A mi padre por ser desde que tengo uso de razón ese ejemplo a seguir, esa figura masculina que en mi vida lo puede todo, por encaminarme en la vida y desde pequeña inculcarme la idea de ser independiente.

A los dos viejitos más importantes de mi vida, a mi abuelo por poder contar con él en todo momento y mi abuela por ser esa segunda madre que siempre está presente, que toma mis problemas como si fueran suyos, que piensa en mí antes que en ella, que se ha mantenido a mi lado en estos cinco años riendo con cada logro y llorando con cada obstáculo que se ha presentado, sin ella llegar a este día hubiera sido notablemente difícil.

A mi hermana por ser esa amiga en la que puedo confiar, por permitirme ser para ella ese ejemplo a seguir, por brindarme, su apoyo incondicional y estar presente sobre todo en esos momentos que se necesita ser escuchado.

A mis primos, tíos y toda mi familia en general que en su momento me han brindado su apoyo y comprensión.

A mis tutores, que además de tutores son amigos, por su ayuda, paciencia y dedicación, gran parte de este logro se los debo a ellos.

A todos los profesores que a lo largo de la carrera contribuyeron en mi formación personal y profesional.

A todas las personas y amigos que de una forma u otra me han demostrado que puedo contar con ellos cuando más lo he necesitado.

DEDICATORIA

Dedico este trabajo a mis padres, por haberme dado la vida y por ser la fuente de fuerza e inspiración en mi vida.

A mi hermana, que, a pesar de nuestras diferencias e incontables desacuerdos, la quiero con toda el alma y aunque muchas veces diga lo contrario es el mejor regalo que mis padres me han podido dar.

A mis dos abuelos que son mis segundos padres. Aunque hoy lejos, siempre los tengo muy cerca y sin su ayuda me hubiera sido difícil llegar tan lejos.

A la lista de amigos interminable que me han acompañado en estos cinco años y a todas aquellas personas que de una forma u otra me apoyaron para que este día se volviera realidad.

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Leidy Márquez Barbosa, con carné de identidad 96101802999 soy la autora principal del trabajo titulado “Módulo Abastecimiento para el Sistema Integrado de Alimentación de la Universidad de las Ciencias Informáticas” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los 19 días del mes de junio del año 2019.

Autor:

Leidy Márquez Barbosa

Tutores:

Ing. Kenia González Quintana

Ing. Dairo Roberto Gil Martín

Co-tutor:

Lic. Diana Aquino Ruiz

DATOS DE CONTACTO

Tutores:

Ing. Kenia González Quintana: Graduada en el año 2015 como Ingeniera en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Se desempeña como especialista B en Ciencias Informáticas en el Departamento de Tecnología de la Dirección de Informatización ocupando el rol de Analista de Componente del proyecto Sistema Integrado de Alimentación. Correo electrónico: kgonzalez@uci.cu

Ing. Dairo Roberto Gil Martín: Graduado en el año 2011 como Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Se desempeña como Jefe de Departamento en el Departamento de Tecnología de la Dirección de Informatización ocupando el rol de Programador del proyecto Sistema Integrado de Alimentación. Correo electrónico: drgil@uci.cu

Lic. Diana Aquino Ruiz: Graduada en el año 2005 de Licenciatura en Contabilidad y Finanzas en la Universidad de Oriente. Se desempeña como Metodóloga de la Dirección de Informatización desempeñando el rol de Asesora de Control Interno, y profesora de la disciplina de Ciencias Empresariales del Departamento de Ingeniería de Software de la facultad 1 en la Universidad de las Ciencias Informáticas. Correo electrónico: diana@uci.cu

RESUMEN

En la Universidad de las Ciencias Informáticas surge la necesidad de desarrollar sistemas automatizados que apoyen los servicios brindados a la comunidad universitaria. El sistema actual para la gestión de los procesos de alimentación no cuenta con la aceptación, ni con las exigencias de la Dirección General de Alimentos. No existe una manera de planificar el menú en función de la disponibilidad de los platos y de los productos en los almacenes de la Universidad.

La presente investigación estuvo orientada a fundamentar y documentar el proceso de desarrollo del Módulo Abastecimiento. Se realizó un análisis de fuentes bibliográficas relacionadas con la gestión del abastecimiento y de sistemas que contribuyen a la planificación de menú. Análisis que permitió realizar el modelado del negocio y describir los procesos identificados en la investigación, así como definir todos los requisitos funcionales y no funcionales necesarios para el desarrollo del módulo.

El proceso fue guiado por la metodología ágil de desarrollo de *software Scrum* y se utilizó el marco de trabajo *Symfony*. El sistema gestor de base de datos empleado fue *PostgreSQL* y *NetBeans* como entorno de desarrollo. Para lograr una aplicación con la menor cantidad de errores, se le realizaron pruebas de unidad, validación y sistema.

Como resultado se obtuvo un módulo que permite configurar los productos existentes en los almacenes, conformar los platos a partir de la disponibilidad de los productos y planificar el menú; cubriendo de esta forma las necesidades de la Universidad en cuanto a la gestión del proceso de abastecimiento.

Palabras clave:

abastecimiento, menú, módulo, planificación.

ÍNDICE

INTRODUCCIÓN.....	1
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA PARA EL DESARROLLO DEL MÓDULO ABASTECIMIENTO	5
1.1. CONCEPTOS ASOCIADOS.....	5
1.2. SOLUCIONES EXISTENTES QUE GESTIONAN EL ABASTECIMIENTO.....	6
1.2.1. <i>Soluciones existentes que gestionan el abastecimiento a nivel internacional.</i>	6
1.2.2. <i>Soluciones existentes que gestionan el abastecimiento a nivel nacional.</i>	9
1.2.3. <i>Resumen sobre el estudio realizado a las soluciones existentes que gestionan el abastecimiento en el ámbito nacional e internacional.</i>	9
1.3. METODOLOGÍA, TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS.....	11
1.4. CONCLUSIONES PARCIALES.....	18
CAPÍTULO 2. DESCRIPCIÓN Y ANÁLISIS DEL MÓDULO ABASTECIMIENTETO.....	19
2.1. MODELADO DEL PROCESO DE NEGOCIO	19
2.2. PROPUESTA DE SOLUCIÓN	20
2.3. ESPECIFICACIÓN DE LOS REQUISITOS DE SOFTWARE.....	21
2.3.1. <i>Requisitos funcionales</i>	22
2.3.2. <i>Requisitos no funcionales</i>	26
2.4. DEFINICIÓN DE LOS ACTORES	29
2.5. PATRÓN ARQUITECTÓNICO	29
2.6. PATRONES DE DISEÑO	31
2.6.1. <i>Patrones Generales de Asignación de Responsabilidades</i>	31
2.6.2. <i>Patrones GOF</i>	33
2.7. MODELO DE LA BASE DE DATOS	33
2.8. DIAGRAMA DE DESPLIEGUE	35
2.9. CONCLUSIONES PARCIALES.....	37
CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA.....	38
3.1. ESTÁNDARES DE CODIFICACIÓN.....	38
3.2. VALIDACIÓN DE REQUISITOS.....	41

3.2.1.	<i>Criterios para validar los requisitos</i>	42
3.2.2.	<i>Técnicas de validación de requisitos</i>	43
3.3.	PRUEBAS	43
3.3.1.	<i>Prueba de unidad</i>	44
3.3.2.	<i>Prueba de validación</i>	47
3.3.3.	<i>Pruebas de sistema</i>	52
3.4.	CONCLUSIONES PARCIALES.....	54
	CONCLUSIONES GENERALES.....	55
	RECOMENDACIONES	56
	ANEXOS.....	60
	GLOSARIO DE TÉRMINOS	68

Índice de tablas

Tabla 1 Características generales de las soluciones estudiados.....	10
Tabla 2 Historia de Usuario. RF Mostrar menú.	25
Tabla 3 Nomenclatura de los requisitos no funcionales del Módulo Abastecimiento.	27
Tabla 4 Requisitos no funcionales del Módulo Abastecimiento.	27
Tabla 5: Actores del Módulo Abastecimiento.	29
Tabla 6 Pruebas realizadas.	44
Tabla 7 Prueba de unidad.....	45
Tabla 8 CP Registrar clasificación de plato.....	47
Tabla 9 Descripción de las variables del CP Registrar clasificación de plato.	51
Tabla 10 Iteraciones de los casos de prueba.	52
Tabla 11 Resultados de las pruebas de rendimiento y resistencia.	53

Índice de figuras

Figura 1 Modelo del proceso de negocio del Módulo Abastecimiento para el Sistema Integrado de Alimentación	20
Figura 2 Modelo de datos	34
Figura 3 Diagrama de despliegue.	36
Figura 4 Indentación, llaves de apertura y cierre y tamaño de líneas	39
Figura 5 Convención de nomenclatura: variable camelCase.....	39
Figura 6 Convención de nomenclatura. Clase nombre compuesto. StudlyCaps	40
Figura 7 Convención de nomenclatura: función: camelCase.....	40
Figura 8 Estructuras de control	41
Figura 9 Ejemplo de archivo documentado	41

INTRODUCCIÓN

El creciente uso de las Tecnologías de la Información y las Comunicaciones (TIC) en la última década está revolucionando la sociedad (González et al. 2009). Están presentes en gran parte de las actividades humanas como son la educación, la comunicación, la medicina y el mundo empresarial. A lo largo de todo este desarrollo, la búsqueda de un eficiente manejo de la información ha sido objetivo primordial (Martínez y Concepción, 2014). Al tener en cuenta que el acceso a la información y los servicios se extiende cada día más, el uso de las TIC se ha extendido también a la gestión del abastecimiento. El objetivo de esta última, es abastecer los productos necesarios en la cantidad y tiempo requerido, para brindar un mejor servicio a las personas.

En Cuba, informatizar la gestión del abastecimiento significa, para una entidad, una estrategia que permite hacer un uso racional de los productos disponibles. Por tal razón, varias entidades han dedicado tiempo y recursos a impulsar la informatización de la sociedad, una de ellas es la Universidad de las Ciencias Informáticas (UCI). Es una Universidad creada a partir de una idea del Comandante en Jefe Fidel Castro, la cual tiene como meta la formación de profesionales comprometidos y altamente calificados en la rama de la Informática. A partir de la vinculación estudio-trabajo brinda soporte a la industria cubana de la informática con el desarrollo de sistemas que puedan ser integrados con servicios informáticos.

La Universidad brinda un conjunto de servicios a la comunidad universitaria que con el tiempo se han ido informatizando para gestionar los procesos que ocurren dentro de la misma. La Dirección de Informatización (DIN) es el área encargada de la informatización de estos procesos, donde surge la idea de los Sistemas Akademos, Gestión Universitaria (SGU), Sistema de Gestión Ciudadana (SGC) y el Sistema Integrado de Alimentación (SIA). Actualmente la DIN, desarrolla el SIA, el objetivo de este sistema es informatizar los procesos que se llevan a cabo en la Dirección General de Alimentos de la Universidad, donde muchos de ellos se realizan de forma manual. Esta área es la encargada de planificar el menú universitario y distribuir los comensales por complejos comedores.

La Dirección General de Alimentos, para la planificación del menú universitario, trabaja directamente con el sistema ASSETS que es un Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de compras, ventas, producción, taller, inventario, finanzas, contabilidad, presupuesto, activos fijos, útiles y herramientas y recursos humanos (Hernández y Castro, 2007). En este sistema se tiene

registrada la disponibilidad de los productos en cada almacén de la universidad. El registro de esta información en el sistema se obtiene de los vales o facturas que emiten los almacenes y se realiza de forma manual, lo que trae como consecuencias que:

- ✓ No existen datos confiables a la hora de procesar la información que brinda el sistema ASSETS a la Dirección General de Alimentos.
- ✓ Retraso en el procesamiento de la información a la hora de planificar el menú universitario.
- ✓ El desconocimiento de la disponibilidad de los productos alimenticios en los almacenes da paso a que en disímiles ocasiones existan platos sujetos a cambio.
- ✓ No existe un mecanismo definido donde se actualice de forma automática la información que se almacena en el ASSETS UCI, de los productos alimenticios disponibles en los almacenes de la Universidad.
- ✓ El control de los productos alimenticios en ocasiones se torna complejo debido al alto consumo de los mismos.

De la situación antes expuesta se propone como **problema de investigación**: ¿Cómo contribuir a la gestión del abastecimiento en la Universidad de las Ciencias Informáticas desde el Sistema Integrado de Alimentación?

Se define como **objeto de estudio**: sistemas informáticos que gestionan el abastecimiento, siendo el **campo de acción**: la gestión del abastecimiento en el Sistema Integrado de Alimentación de la Universidad de las Ciencias Informáticas.

Para la solución del problema planteado se propone como **objetivo general**: Desarrollar un Módulo Abastecimiento para el Sistema Integrado de Alimentación que contribuya a la planificación del menú en la Universidad de las Ciencias Informáticas.

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

- ✓ Realizar el estudio de la documentación existente en el ámbito nacional e internacional sobre los sistemas que gestionan el abastecimiento para conocer sus características.
- ✓ Describir la metodología de desarrollo de software, herramientas, tecnologías y lenguaje de

programación que conforman la línea base de la arquitectura para el desarrollo del Módulo Abastecimiento.

- ✓ Definir los requisitos funcionales y no funcionales del Módulo Abastecimiento.
- ✓ Implementar los requisitos funcionales del Módulo Abastecimiento.
- ✓ Aplicar pruebas para validar la solución.

La investigación se centra en las siguientes **preguntas científicas**:

- ✓ ¿Cuáles son los referentes teóricos que sustentan el desarrollo de la investigación, relacionados con los sistemas que gestionan el abastecimiento?
- ✓ ¿Qué características cumplen las herramientas, tecnologías y metodologías que conforman la línea base de la arquitectura para el desarrollo del Módulo Abastecimiento?
- ✓ ¿Qué elementos deben tenerse en cuenta para realizar el análisis, diseño e implementación del Módulo Abastecimiento?

Alcance

Con el desarrollo de la presente investigación se esperan los siguientes beneficios:

- ✓ Permitirá la planificación del menú en el Sistema Integrado de Alimentación de la Universidad de las Ciencias Informáticas a partir de la disponibilidad de productos alimenticios en los almacenes.
- ✓ Posibilitará gestionar los tipos de productos, aplicarle una merma, gestionar las unidades de medida, las categorías de los tipos de productos y la clasificación de los platos.
- ✓ Brindará a la comunidad universitaria el servicio de visualización del menú.

Para realizar esta investigación se utilizaron los **siguientes métodos**:

Teóricos:

- ✓ **Modelación:** se utilizó para modelar las diferentes etapas de la ingeniería de *software* por las que transita el desarrollo del Módulo Abastecimiento.
- ✓ **Analítico – Sintético:** permitió analizar las características de los sistemas que gestionan el abastecimiento y definir cuál podrá ser utilizado para el desarrollo del Módulo Abastecimiento.

- ✓ **Histórico-lógico:** se empleó con el objetivo de verificar cómo evolucionan teóricamente los sistemas que gestionan el abastecimiento.

Empíricos:

- ✓ **Tormenta de ideas:** contribuyó para el levantamiento de requisitos, así como para la determinación de las restricciones de *software* e implementación que deben tenerse en cuenta en el proceso de desarrollo.
- ✓ **Consulta a expertos:** se utilizó para evaluar la propuesta que se presenta en la investigación, permitiendo la valoración de la actualidad y significación de los resultados.
- ✓ **Análisis documental:** se empleó para la revisión de la bibliografía necesaria durante la investigación.

Para facilitar la comprensión del documento se estructura de la siguiente forma:

Introducción: se realiza la presentación de la investigación, para incluir los antecedentes y la situación problemática, actualidad, importancia y trascendencia del problema planteado. Se precisa el diseño teórico-metodológico, que constituye el punto de referencia para el procesamiento de los resultados.

Capítulo 1. Fundamentación teórica: en este capítulo se abordan los conceptos fundamentales que tienen relación con el proceso de negocio que gestiona el Módulo Abastecimiento. Se describen algunas soluciones existentes en el ámbito nacional e internacional que gestionan el proceso de abastecimiento de productos permitiendo obtener una valoración de las mismas. Finalmente se justifican las herramientas y tecnologías a utilizar para el diseño y la implementación del módulo en cuestión.

Capítulo 2. Descripción y análisis del Módulo Abastecimiento: en este capítulo se especifican los requisitos funcionales y no funcionales de la propuesta de solución, se describen todos los artefactos que se generan durante el diseño de la misma, se definen los patrones arquitectónicos que se van a utilizar y se abordan las características fundamentales del Módulo Abastecimiento para el Sistema Integrado de Alimentación de la Universidad de las Ciencias Informáticas.

Capítulo 3: Implementación y prueba del Módulo Abastecimiento: se formaliza la implementación de la propuesta de solución. A su vez se confeccionan los casos de prueba a realizar al *software*, se ejecutan y se presentan los resultados que permiten validar la solución propuesta.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA PARA EL DESARROLLO DEL MÓDULO ABASTECIMIENTO

En este capítulo se realiza el estudio de algunas soluciones que gestionan el abastecimiento en el ámbito nacional e internacional, para ello se tienen en cuenta características que permiten comprender su funcionamiento y cuánto aportan a la investigación en curso. Se abordan conceptos asociados a la gestión del abastecimiento. Se describen brevemente la metodología, las herramientas, tecnologías y el lenguaje, que se utilizan en el desarrollo del Módulo Abastecimiento.

1.1. Conceptos asociados

Producto

Según la Real Academia de la Lengua Española, producto se define como “Caudal que se obtiene de algo que se vende, o el que ello reditúa.”

Un producto es un objeto que se ofrece en un mercado con la intención de satisfacer aquello que necesita o que desea un consumidor. En este sentido, el producto trasciende su propia condición física e incluye lo que el consumidor percibe en el momento de la compra (atributos simbólicos, psicológicos, etc.). Es algo material que se elabora de manera natural o industrial mediante un proceso, para el consumo o utilidad de los individuos (Arencibia y Jiménez, 2014).

En la presente investigación producto es: todo aquello que se encuentra disponible en los almacenes, que tiene una unidad de medida, una clasificación, una categoría, se le aplica una merma y se utiliza finalmente para la confección del menú de la Universidad.

ETL

Extraer, transformar y cargar (ETL por sus siglas en inglés *Extract, Transform and Load*). Son herramientas que proporcionan a su destinatario, informes o estrategias que pueden beneficiar la toma de decisiones en una empresa, recogiendo la información más relevante de distintas fuentes e incluso siendo éstas, fuentes heterogéneas. Las herramientas ETL ayudan a realizar la unificación e integración de los datos contenidos en estas fuentes diversas y realizan un pre-procesado y carga de todos esos datos en un único sistema o base de datos (Pérez, 2014).

En la presente investigación ETL se utiliza para: obtener la información mediante un usuario de lectura teniendo en cuenta reglas para la integridad de la información desde la base de datos del ASSETS con el objetivo de registrarlos en la base de datos del Sistema Integrado de Alimentación.

Sistema de Abastecimiento

Es un conjunto interrelacionado de políticas, objetivos, procedimientos y procesos técnicos orientados al flujo racional, dotación o suministro, empleo y conservación de medios materiales; así como aquellas acciones especializadas, trabajo o resultado para asegurar la continuidad de los procesos productivos que desarrollan las organizaciones (Medrano y Rodríguez, 2013).

En la presente investigación el Módulo Abastecimiento del Sistema Integrado de Alimentación se encargará de la planificación del menú universitario, en función de los platos disponibles, a partir de la disponibilidad de los productos en los almacenes y la configuración de los mismos.

1.2. Soluciones existentes que gestionan el abastecimiento.

Actualmente existen soluciones encargadas de gestionar el proceso de abastecimiento. Por tal razón se hace necesario realizar un estudio de las mismas, con el objetivo de definir si cubren las necesidades planteadas en la problemática descrita y si pueden ser adaptadas a la Universidad de las Ciencias Informáticas. En caso contrario, este estudio permitirá analizar todos los aspectos que puedan ser de utilidad para el desarrollo de la propuesta de solución, ajustada a las necesidades de la universidad en cuanto a la gestión del abastecimiento.

1.2.1. Soluciones existentes que gestionan el abastecimiento a nivel internacional.

En la actualidad, existen soluciones informáticas que gestionan el proceso de abastecimiento de una empresa o entidad, luego de un estudio realizado en el ámbito nacional e internacional se tienen las siguientes:

- ✓ **Sistema Informático Web de gestión de pedidos y abastecimiento de materiales para la empresa Proyersac.**

Es un sistema informático creado como parte de un Trabajo de Suficiencia en la Universidad Privada Antenor Orrego en Perú para agilizar el proceso de abastecimiento de materiales para la empresa *Proyersac*, este sistema gestiona requisitos de compra, solicitudes de cotización, orden de compras, guías de remisión y

aprobación de documentos con el objetivo de tener la información integrada, tener un mejor control de los inventarios y brindar reportes de forma eficiente para mejorar la producción.

Para el desarrollo de este sistema se usó el *Rational Rose* como herramienta de modelado, *Adobe Dreamweaver CS4* para realizar el prototipado de interfaces en el proyecto y *MySQL* como sistema de gestión de bases de datos relacionales (Medrano y Rodríguez, 2013).

¿Por qué no utilizarlo?

A pesar de ser una solución que cubre parte de los requisitos funcionales de la propuesta de solución incorpora otras funcionalidades que no son necesarias. No contempla entre sus procesos de negocio la gestión de platos, la gestión de unidades de medida y la gestión de los productos disponibles en el almacén.

- ✓ **Sistema Web Móvil para realizar reservas de menú en el Centro de Prácticas Pre-Profesionales de alimentos y bebidas El Mesón del Estudiante de la Universidad Ricardo Palma.**

Es un sistema informático creado como parte de un Trabajo de Diploma en la Universidad Ricardo Palma en Perú que brinda, en primera instancia, a sus estudiantes una manera sencilla de tener información sobre los platos ofrecidos durante el día o la semana. De igual forma se gestionan las reservas de la administración y los estudiantes de la universidad de una manera interactiva, rápida, innovadora y eficiente, otorgando al Mesón del Estudiante un mejor servicio y aumentar sus ingresos.

Para el desarrollo de este sistema se utilizó *MySQL* como sistema gestor de base de datos, Java como lenguaje de programación, Apache como servidor web, *Rational Rose* como herramienta de modelado y Windows XP como sistema operativo (León y Barbaimon, 2014).

¿Por qué no utilizarlo?

A pesar de ser una solución que en su proceso de negocio contempla la gestión de menú, no responde completamente al proceso de abastecimiento que se lleva a cabo en la Universidad. Además, es una solución desarrollada específicamente para la plataforma móvil, la tecnología sobre la que está desarrollada no permite adherirlo al Sistema Integrado de Alimentación. Adaptarla a las necesidades de la Universidad en cuanto a la gestión del abastecimiento implicaría tiempo y trabajo, por tal motivo se desarrolla la nueva propuesta de solución.

✓ **Sistema de Información de Logística para la Gestión de Insumos y Productos en una empresa del Rubro de Panadería y Pastelería.**

Es un sistema de información creado como parte de un Trabajo de Diploma en la Universidad Católica de Perú que permita gestionar y controlar los insumos y productos de la empresa mediante la administración y mantenimiento de las actividades diarias de compra y venta de los productos. Con el fin de disponer de información actualizada de ellos con mayor rapidez y facilidad, así como de los movimientos que se realicen en el área de almacén; para ello, se realiza previamente el análisis de sus procesos, la identificación de los requerimientos y funcionalidades que serán implementadas y automatizadas. Además, el sistema utiliza como base la información registrada para poder generar los reportes que den soporte a la toma de decisiones.

Cuenta con tres módulos implementados que permiten el registro y consulta de las actividades de compra y venta de los productos, la validación de los ingresos y salidas de los productos para la actualización del *stock* (existencia en el almacén) del almacén y la generación de reportes relacionada a la gestión de insumos y productos. Para el desarrollo de este sistema se usó la herramienta *BizAgi Process Modeler* utilizada para el modelado de procesos bajo la notación BPMN, PHP como lenguaje de programación, *MySQL* como gestor de base de datos, *Kohana* como *framework* PHP, *Netbeans* como entorno integrado (IDE) de desarrollo siguiendo el patrón Modelo-Vista-Controlador (Sone, 2015).

¿Por qué no utilizarlo?

Es una solución que su proceso de negocio no se aplica a las necesidades de la gestión del abastecimiento en la Universidad de las Ciencias Informáticas. Entre sus funcionalidades comprende la gestión de compras, ventas y el registro de entradas de insumos al almacén, las cuales no son necesarias en el desarrollo de la nueva propuesta. No es capaz de aplicarle una merma a un producto, clasificarlo, agruparlo en tipos de productos, asignarle una categoría, una unidad de medida, etc.

✓ **Diseño de una propuesta de gestión de abastecimiento e inventarios para un astillero en Colombia.**

Es un sistema de gestión de abastecimiento e inventarios para un astillero en Colombia creado como parte de un Trabajo de Grado en la Universidad Nacional de Colombia con el fin de establecer estrategias para el aprovisionamiento y almacenamiento de los materiales necesarios para la construcción y reparación de

buques, lo que permitirá mejorar la competitividad de la empresa a través de la disminución de los tiempos de respuesta, los costos logísticos y la mejora del nivel de servicio en el suministro de bienes (Otero, 2012).

¿Por qué no utilizarla?

Es una propuesta diseñada para una entidad específica, en este caso un Astillero en Colombia. Cada uno de los procesos que se tienen en cuenta se fundamentan con el desarrollo de estrategias y herramientas que responden al proceso de negocio de esta entidad. Por tal motivo se considera una propuesta que no puede ser aplicada a la Universidad, pues no responde a la necesidad de la misma en cuanto a la gestión del abastecimiento.

1.2.2. Soluciones existentes que gestionan el abastecimiento a nivel nacional.

- ✓ **Diseño de un modelo general para la gestión de sistemas logísticos en empresas cubanas: consideraciones teóricas y prácticas.**

Es un sistema logístico creado en la Universidad de Santiago de Cuba cuyo objetivo es garantizar que las empresas estatales y organizaciones superiores de dirección logren en su gestión integral la máxima eficiencia y eficacia, permite obtener el producto, en el tiempo oportuno, en el sitio apropiado, y al menor costo posible, también permitirá mejorar la competitividad de las empresas, la adquisición, el movimiento, el almacenamiento de productos y el control de inventarios, así como todo el flujo de información asociado a estas actividades, de forma tal que la rentabilidad presente y futura de la empresa sea maximizada en términos de costos y efectividad (Hernández, 2011).

¿Por qué no utilizarla?

Es una propuesta diseñada para empresas cubanas donde se aplique el perfeccionamiento empresarial. Los procesos que la componen no se describen detalladamente porque depende de las características de cada empresa. La Universidad de las Ciencias Informáticas a pesar de ser una institución creada para la producción de *software* que vincula los procesos de docencia-investigación-producción es considerada una Universidad y no una empresa, por tal razón este modelo no responde a las necesidades de la misma en cuanto a la gestión del abastecimiento.

1.2.3. Resumen sobre el estudio realizado a las soluciones existentes que gestionan el abastecimiento en el ámbito nacional e internacional.

El estudio de las soluciones anteriormente descritas, permitió comprobar que estas no responden a las necesidades actuales de la Dirección General de Alimentos de la Universidad de las Ciencias Informáticas. Demostró la necesidad de desarrollar una solución que permita la informatización de los procesos de negocio de dicha área. A continuación, se presenta una tabla resumen donde se recogen los datos más significativos de cada una de las soluciones analizadas.

Los criterios a comparar fueron definidos por la autora de la presente investigación, a continuación, se realiza una breve descripción de cada uno de ellos:

Administración de productos: es el proceso que administra el ciclo de vida completo de un producto desde su fabricación, hasta su servicio y eliminación.

Disponibilidad de información actualizada: es la manera en que la información una vez que ha sido capturada en un sistema de cómputo es almacenada de manera segura para que los usuarios accedan a ella cuando la necesiten.

Aplicación de merma a los productos: es una pérdida o reducción de un cierto número de mercancías o de la actualización de un inventario, que provoca, una diferencia entre el contenido del inventario y la cantidad real de productos dentro de un establecimiento.

Control de ingresos y salidas en el almacén: es un proceso integrado que incluye la planeación, registro y control de todas las entradas y salidas de un almacén.

Planificación de menú: la planificación es un método que permite ejecutar planes de forma directa, en este caso menús, los cuales serán realizados y supervisados en función del planeamiento.

Tabla 1 Características generales de las soluciones estudiados.

Soluciones Criterios	Propuesta de solución.	Sistema informático o web para la empresa <i>Proyersac.</i>	Sistema web móvil para la universidad Ricardo Palma.	Sistema de información de logística para una empresa del rubro de Panadería y Pastelería	Propuesta de gestión de abastecimiento para un astillero en Colombia.	Modelo general para la gestión de sistemas logísticos para empresas cubanas.

Administración de productos.	No	Sí	Sí	No	Sí	Sí
Disponibilidad de información actualizada.	Sí	Sí	Sí	Sí	Sí	Sí
Aplicación de merma a los productos.	Sí	No	No	No	No	No
Control de ingresos y salidas en el almacén.	No	Sí	Sí	No	Sí	Sí
Planificación de menú	Sí	No	Sí	No	No	No

1.3. Metodología, tecnologías y herramientas utilizadas.

Para el desarrollo del Módulo Abastecimiento se utilizará una metodología de desarrollo ágil, así como herramientas y tecnologías libres de licencia que favorezcan posteriormente la creación de nuevos módulos o subsistemas. A continuación, se describen cada una de ellas.

Scrum: es considerada un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos desde principios de los años 90. Muestra la eficacia relativa de las prácticas de gestión de producto y de desarrollo. Los equipos son auto organizados y multifuncionales. Los auto organizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. Entregan productos de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación. Las entregas incrementales de producto “Terminado” aseguran que siempre estará disponible una versión potencialmente útil y funcional del producto (Schwaber y Sutherland, 2013).

Al ser una metodología de desarrollo ágil tiene como base la idea de creación de ciclos breves para el desarrollo, que comúnmente se llaman iteraciones y que en *Scrum* se llamarán *sprints* (Trigas, 2015).

Cómo funciona *Scrum*:

Para entender todo el proceso de desarrollo de *Scrum*, se describirá de forma general las fases, los roles y los elementos que forman parte de dicha metodología.

- ✓ **Primera fase: Planificación del *Backlog*:** se realizará un documento en el que se reflejarán los requisitos del sistema por prioridades. Se definirá también la planificación del *Sprint 0*, en la que se decidirá cuáles van a ser los objetivos y el trabajo que hay que realizar para esa iteración. Se obtendrá además un *Sprint Backlog*, que es la lista de tareas y el objetivo más importante del *Sprint*.
- ✓ **Segunda fase: Seguimiento del *sprint*:** en esta fase se hacen reuniones diarias en las que las tres preguntas principales para evaluar el avance de las tareas son:
 - ¿Qué trabajo se realizó desde la reunión anterior?
 - ¿Qué trabajo se realizará hasta una nueva reunión?
 - Inconvenientes que han surgido y que hay que resolver para poder continuar.
- ✓ **Tercera fase: Revisión del *sprint*:**

Cuando se finaliza el *Sprint* se realiza una revisión del incremento que se ha generado y se presentan los resultados finales.

Roles de *Scrum*

- ✓ ***Product Owner*:** es la persona que toma las decisiones, la que realmente conoce el negocio del cliente y su visión del producto. Se encarga de escribir las ideas del cliente, las ordena por prioridad y las coloca en el *Product Backlog*.
- ✓ ***ScrumMaster*:** es el encargado de comprobar que el modelo y la metodología funcionan. Elimina todos los inconvenientes que hagan que el proceso no fluya e interactuará con el cliente y con los gestores.
- ✓ **Equipo de desarrollo:** suele ser un equipo pequeño de unas cinco a nueve personas y tienen autoridad para organizar y tomar decisiones para conseguir su objetivo. Está involucrado en la estimación del esfuerzo de las tareas del *Backlog*.

Elementos de *Scrum*

- ✓ **Product Backlog (PB):** Es una lista que abarca todo lo necesario para que el producto satisfaga las necesidades de clientes potenciales. Es el documento central de un proyecto basado en *Scrum* y la principal referencia a la hora de realizar cambios o plantear soluciones.
- ✓ **Sprint Backlog (SB) o Lista de objetivos pendientes del *sprint*:** Es un subconjunto de elementos del PB seleccionados a abordar en el *Sprint*, más un plan para entregarlos como Incremento del producto. Cuando ciertos objetivos no se cumplen y las soluciones no pueden implementarse en el momento, es necesario trasladarlas al siguiente ciclo de trabajo. La lista de objetivos pendientes ayudará a tenerlos presentes.
- ✓ **Incremento:** es la forma en que se mide el progreso que ha tenido el proceso en cada etapa. Para esta metodología, es esencial que cada iteración tenga un incremento; si no es así, esto revelará que algo ha fallado.

El módulo a desarrollar es parte de un sistema donde el equipo que lo conforma tiene definidas políticas de desarrollo, entre ellas se encuentran, usar la metodología *Scrum* para guiar el ciclo de vida del proyecto. Se utilizó con el objetivo de obtener un *software* eficiente, mediante la planificación total del trabajo a realizar.

Symfony 2.8

Symfony es un proyecto de *software* libre que publica pequeñas librerías PHP independientes llamadas "componentes". Combinando varios de esos componentes, también ha publicado un *framework* para desarrollar aplicaciones web. Debido al uso de los componentes, la arquitectura interna del *framework* está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un determinado proyecto. *Symfony* también es el *framework* que más ideas incorpora del resto de *frameworks*, incluso de aquellos que no están programados con PHP. *Symfony 2.8* se publicó en noviembre de 2015 y es la última versión de la rama 2.x (Eguiluz, 2016). Para el desarrollo de la propuesta de solución se utilizó esta distribución ya que es la usada en la arquitectura sobre la cual es desarrollado el sistema al que pertenece el Módulo Abastecimiento.

Twig

Twig es un motor y lenguaje de plantillas para PHP que es rápido, eficiente y seguro. La sintaxis de *Twig* se ha diseñado para que las plantillas sean concisas y muy fáciles de leer y de escribir, por lo que *Symfony* recomienda utilizar *Twig* para la creación de las mismas. El objetivo de *Twig* es conseguir que los diseñadores sin conocimientos de programación sean capaces de crear todas las plantillas de la aplicación de forma autónoma, sin ayuda de los programadores. De esta forma se acelera el desarrollo de las aplicaciones y se mejora la productividad. Por eso *Twig* ha sido ideado para que sea realmente fácil de aprender, leer y escribir por parte de profesionales sin un perfil técnico avanzado (Eguiluz, 2016). Se utilizó con el objetivo de modelar las plantillas que serán usadas del lado del cliente en el desarrollo del Módulo Abastecimiento.

PHP 7

PHP es un lenguaje de programación usado generalmente para la creación de contenido para sitios Web. Es un acrónimo recurrente que significa "PHP *Hypertext Pre-processor*". Su interpretación y ejecución se da en el servidor, en el cual se encuentra almacenado el *script*, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, generada por un *script* PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el *script* solicitado que generará el contenido de manera dinámica, pudiendo modificar el contenido a enviar, y regresa el resultado al servidor, el cual se encarga de regresárselo al cliente. Además, es posible utilizar PHP para generar archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como *MySQL*, *Postgres*, *Oracle*, *ODBC*, *DB2*, *Microsoft SQL Server*, *Firebird* y *SQLite*; lo cual permite la creación de Aplicaciones Web muy robustas. PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX (y de ese tipo, como *Linux*), *Windows* y *MAC OS X*, y puede interactuar con los servidores de Web más populares ya que existe en versión CGI y módulo para Apache (Sæther y Aulbach, 2002). Para el desarrollo del Módulo Abastecimiento se utilizó php como lenguaje de programación por sus ventajas y facilidades anteriormente descritas.

Ajax

Ajax es un conjunto de tecnologías: *JavaScript*, XML y un lenguaje del lado del cliente. Con *Ajax* el *JavaScript* puede comunicarse directamente con el usuario, usando el objeto de *XMLHttpRequest* del *JavaScript*. La idea esencial de *Ajax* es que con este objeto puede hacer una petición al servidor sin tener que recargar la página. Esta tecnología usa la transferencia de datos asíncrona (peticiones del HTTP) entre el navegador y el servidor web, permitiendo que las páginas web envíen pedazos de pequeñas informaciones al usuario en vez de enviar las páginas enteras. La técnica de *AJAX* hace que aplicaciones de Internet sean más pequeñas, más rápidas y más *userfriendly* (amigables para el usuario) (Ayoze, 2017).

Para el desarrollo del Módulo Abastecimiento fue utilizada para realizar peticiones asíncronas desde el cliente al servidor de manera transparente al usuario, permitiendo realizar acciones al servidor sin la necesidad de recargar toda la página, y de esta forma mejorar en cuestión de rendimiento.

jQuery

Es un *framework JavaScript* que ofrece una infraestructura con la que se tiene mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Con *jQuery* se obtiene ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de *Ajax*, etc. Implementa una serie de clases (de programación orientada a objetos) que permiten programar sin preocuparse del navegador con el que está visitando el usuario, ya que funcionan de forma exacta en todas las plataformas más habituales. Además, tiene licencia para uso en cualquier tipo de plataforma, personal o comercial (The jQuery Foundation., 2018). Para el desarrollo del Módulo Abastecimiento permitió hacer uso de funciones *JavaScript* de una manera sencilla, como el uso de los selectores de esta herramienta y su integración con la librería *Ajax*, permitiendo un código más sencillo y fácil de mantener.

Nginx

Nginx es un servidor web y proxy inverso (mecanismo desplegado en una red para proteger a los servidores HTTP de una intranet, realiza funciones de seguridad que protegen a los servidores internos de ataques de usuarios en internet). Gestiona las peticiones por medio de una arquitectura orientada a eventos donde las peticiones son aceptadas mediante *sockets* asíncronos y son procesadas en un único hilo de ejecución, esto a fin de reducir memoria y uso de CPU. La comunidad de *Nginx* no es tan amplia en comparación a la de Apache debido a su corto tiempo de vida en el mercado. Los tiempos de respuesta son menores con *Nginx*, Apache se torna cada vez más lento cuando las peticiones aumentan y además tiende a usar más

ancho de banda para servir las mismas peticiones (Campoverde y Hernández, 2015). Utilizado como servidor web para visualizar el proceso de desarrollo del Módulo Abastecimiento.

PostgreSQL 9.5

PostgreSQL es un potente sistema de base de datos relacionales de objetos de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas. Los orígenes de *PostgreSQL* se remontan a 1986 como parte del proyecto *POSTGRES* en la Universidad de California en *Berkeley* y tiene más de 30 años de desarrollo activo en la plataforma central.

PostgreSQL se ha ganado una sólida reputación por su arquitectura probada, confiabilidad, integridad de datos, conjunto de características sólidas, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer constantemente soluciones innovadoras y de alto rendimiento. *PostgreSQL* se ejecuta en todos los sistemas operativos principales, ha sido compatible con ACID desde 2001 y tiene complementos poderosos como el popular extensor de base de datos geoespacial *PostGIS*. No es sorprendente que *PostgreSQL* se haya convertido en la base de datos relacional de código abierto elegida por muchas personas y organizaciones (The PostgreSQL Global Development Group, 2018). Se utilizó con el objetivo de almacenar la información con la que trabaja el Módulo Abastecimiento.

Pencil 3.0.4

Programa gratuito de código abierto en inglés para *Windows* y *GNU/Linux*. También puede ser instalado como una extensión (ad-don) en el navegador *Firefox*. Es una solución muy flexible que ofrece una librería formada por más de 50 elementos gráficos, que admiten una edición posterior y la incorporación de nuevos elementos gráficos externos. Provee la exportación a *html*, *open office*, *word*, *pdf* y *pencil*; pero sólo importa su propio formato. Soporta la creación de prototipos dinámicos y anotaciones (Pérez, 2010). Se utilizó para modelar los prototipos no funcionales del Módulo Abastecimiento.

NetBeans 8.0

Es un entorno integrado de desarrollo o *Integrated Development Environment* (IDE, por sus siglas en inglés) en el cual se realizan todas las tareas asociadas a la programación: Editar el código, Compilarlo, Ejecutarlo y Depurarlo. Es modular, de base estándar (normalizado), escrito en el lenguaje de programación *Java*. El proyecto *NetBeans* constituye un IDE de código abierto y una plataforma de aplicación, las cuales pueden

ser usadas como una estructura de soporte general (*framework*) para compilar cualquier tipo de aplicación. Entre sus principales ventajas se encuentran: (González y Gimeno, 2011):

- ✓ Simplifica alguna de las tareas que, sobre todo en proyectos grandes, son tediosas.
- ✓ Asiste (parcialmente) en la escritura de código, aunque no nos libera de aprender el lenguaje de programación.
- ✓ Ayuda en la navegación de las clases predefinidas en la plataforma.
- ✓ Aunque puede ser costoso su aprendizaje, los beneficios superan las dificultades.

Se utilizó para la realización de todas las tareas asociadas a la programación (editar el código, ejecutarlo y compilarlo) durante el desarrollo del Módulo Abastecimiento.

PgAdmin 3

Es una herramienta gráfica de código abierto para administrar y desarrollar bases de datos en *PostgreSQL*. Incluye una interfaz administrativa gráfica, una herramienta de consulta SQL (con un *EXPLAIN* gráfico), editor de código procedural, agente de planificación *SQL/shell/batch* y soporte para el motor de replicación *Slony-I*. Esta herramienta está diseñada para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta bases de datos complejas. La interfaz gráfica soporta todas las características de *PostgreSQL* y hace simple la administración. Está disponible en varios lenguajes y para varios sistemas operativos (Vaillant y Ruiz 2015). Para el desarrollo del Módulo Abastecimiento se utilizó con el objetivo de trabajar con los objetos que están almacenados en la base de datos, examinar sus propiedades y realizar tareas administrativas.

Visual Paradigm 8.0

Es una herramienta UML que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones de gran calidad, mejoras y a un menor costo de producción. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Está diseñado para desarrollar software con Programación Orientada a Objetos, permite reducir la duración del ciclo de desarrollo brindando ayuda a los arquitectos, analistas, diseñadores y desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores. Dentro de sus características fundamentales están (Ruiz y López, 2014):

- ✓ Multiplataforma: Soportado en plataformas *Java* para Sistemas Operativos *Windows*, *Linux* y *Mac*.
- ✓ Interoperabilidad: Intercambia diagramas UML y modelos con otras herramientas. Soporta exportar e importar a XML y archivos *Excel*. Importa archivos de proyectos de *Rational Rose*.

Para el desarrollo del Módulo Abastecimiento se utilizó con el objetivo de modelar la base de datos.

1.4. Conclusiones parciales

Una vez finalizado este capítulo, se arribaron a las siguientes conclusiones:

- ✓ Se definieron los conceptos relacionados con el dominio del problema contribuyendo a una mayor comprensión del entorno del negocio.
- ✓ Después de realizar una valoración acerca de algunas soluciones que gestionan el proceso de abastecimiento en el ámbito nacional e internacional, se decidió no emplear ninguna de ellas, ya que no se ajustan a las necesidades de la Universidad en cuanto a la gestión del abastecimiento.
- ✓ Se determinaron las herramientas, tecnologías y metodología que se utilizarán para el desarrollo del Módulo Abastecimiento, formando las bases propicias para la creación de una propuesta de solución, que cumpla con el objetivo de la investigación.

CAPÍTULO 2. DESCRIPCIÓN Y ANÁLISIS DEL MÓDULO ABASTECIMIENTO

El presente capítulo tiene como objetivo describir la propuesta del Módulo Abastecimiento para el Sistema Integrado de Alimentación y la confección del menú universitario a partir de la disponibilidad de los productos alimenticios en cada uno de los almacenes de la Universidad. A partir del flujo descrito se extraen las características con las que contará el módulo a desarrollar y se modela su proceso de negocio, se listan los requerimientos funcionales y no funcionales. También se muestra el diseño de la base de datos, el diagrama de despliegue, se describe el estilo arquitectónico y los patrones de diseño utilizados en el Módulo Abastecimiento.

2.1. Modelado del proceso de negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente, llevadas a cabo para generar productos y servicios. Modelar los procesos de negocio es una parte esencial de cualquier proceso de desarrollo de *software*. Permite capturar el esquema general y los procedimientos que gobiernan el negocio. Este modelo provee una descripción de dónde se va a ajustar el sistema de *software* considerado dentro de la estructura organizacional y de las actividades habituales. También provee la justificación para la construcción del sistema de *software* al capturar las actividades manuales y los procedimientos automatizados habituales que se incorporarán en el nuevo sistema (Fernández y Varona, 2016).

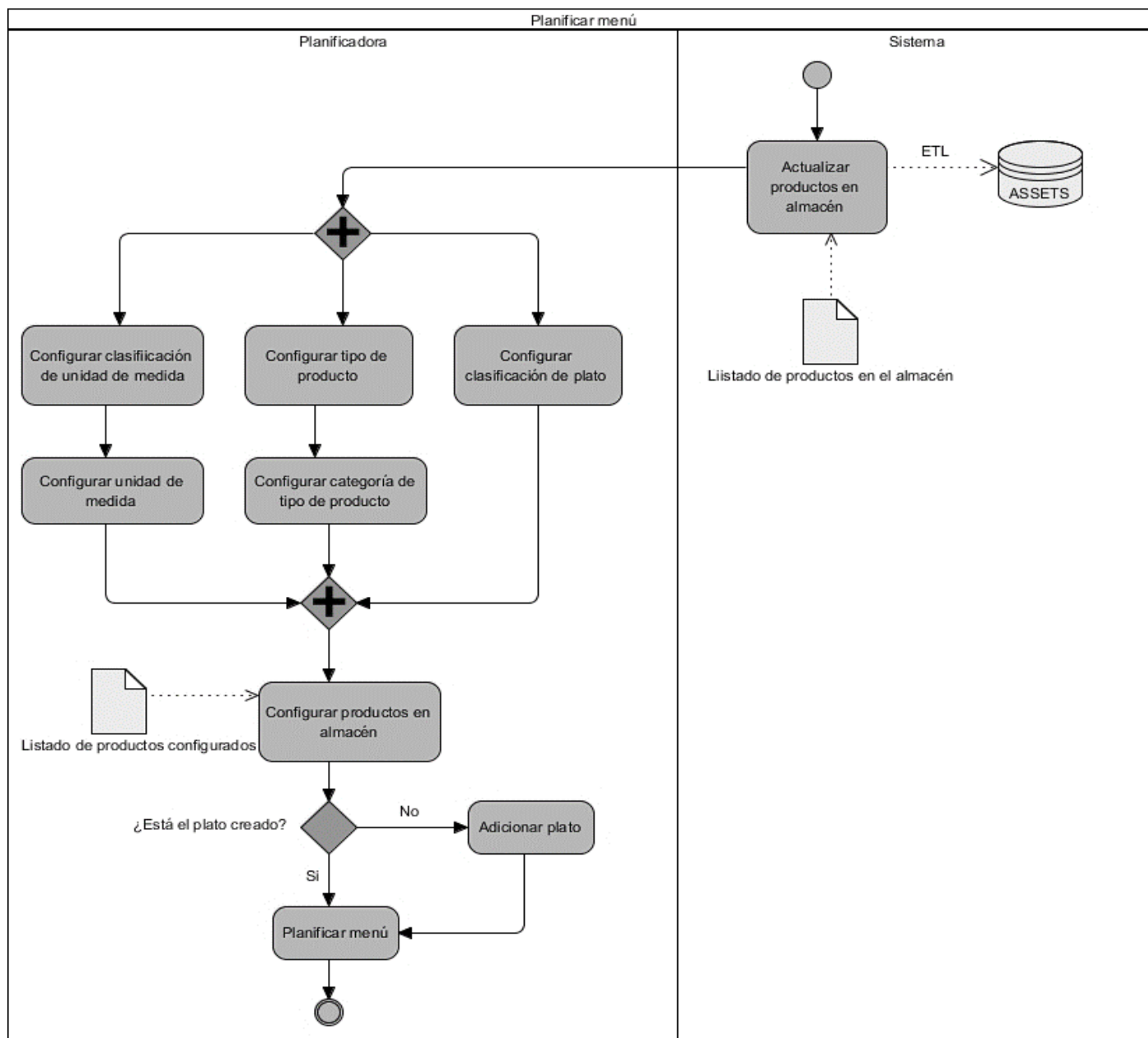


Figura 1 Modelo del proceso de negocio del Módulo Abastecimiento.

2.2. Propuesta de solución

Para dar solución a la problemática planteada se propone la realización de un Módulo Abastecimiento, el cual consiste en una aplicación web, donde su implementación proporcionará al Sistema Integrado de

Alimentación la posibilidad de planificar el menú universitario, en función de los platos disponibles, a partir de la disponibilidad de los productos en los almacenes y la configuración de los mismos.

El módulo a desarrollar, contribuirá en gran medida a facilitar el trabajo de los especialistas de la Dirección General de Alimentos. Contará con 10 funcionalidades que posibilitarán el aumento de la agilidad y organización de este proceso: Clasificación de platos, Unidades de medida, Clasificación de unidades de medida, Tipos de producto, Categorías del tipo de producto, Productos en almacén, Planificación de menú, Productos disponibles, Platos y Menú. Estas funcionalidades estarán divididas en 2 módulos, Abastecimiento como módulo principal y Configuración para un mejor orden de la aplicación.

Para la planificación del menú será necesario realizar la configuración de los productos, conociendo su disponibilidad en almacén. Esto se hará efectivo a través de un mecanismo de extracción y carga de datos que se alimentará de la información que proporciona el sistema ASSETS y la enviará a la base de datos del sistema. Los elementos necesarios para configurar un producto son: categoría, unidad de medida, cantidad disponible, gramaje y la depreciación del producto en el tiempo o merma. Luego de concluido este proceso se configuran los platos que conformarán el menú, para ello se necesita definir previamente una clasificación, la composición, la unidad de medida y el gramaje. Una vez concluido este proceso se procede a la planificación del menú.

Teniendo en cuenta que los productos en almacén responden a las compras que se realizan en la Universidad, su disponibilidad en almacén varía en el tiempo. Para mantener actualizada la disponibilidad real se ejecutará un procedimiento que verificará si el producto ya tiene una configuración previa en el sistema, si la tiene, actualiza su disponibilidad en función de la configuración definida anteriormente sino se realiza todo el proceso definido previamente.

La solución será desarrollada teniendo en cuenta las reglas de negocio definidas por la Dirección General de Alimentos, así como con tecnologías actualizadas, de código abierto y libres de licencia.

2.3. Especificación de los requisitos de software

La captura de los requisitos es una de las actividades fundamentales que se desarrollan durante el proceso de construcción de un *software*. Esta constituye las condiciones o capacidades que el *software* debe cumplir, por tanto, es necesario que sean verificados y validados para evitar errores en futuras etapas que conllevarán a resultados inesperados y atrasos en el desarrollo.

2.3.1. Requisitos funcionales

Son capacidades o condiciones que el sistema debe cumplir, de cómo debería reaccionar el mismo a entradas particulares y de cómo debería comportarse en situaciones específicas. Además, surgen de la razón fundamental de la existencia de un producto (Sommerville, 2011). Luego del análisis de los procesos de negocio estudiados y las actividades a informatizar identificadas se pueden definir los siguientes requisitos funcionales:

Requisitos funcionales del Módulo Configuración.

Funcionalidad Clasificación de platos.

- ✓ **RF1:** Mostar clasificación de platos.
- ✓ **RF2:** Registrar clasificación de plato.
- ✓ **RF3:** Modificar clasificación de plato.
- ✓ **RF4:** Eliminar clasificación de plato.
- ✓ **RF5:** Exportar a Excel.
- ✓ **RF6:** Exportar a PDF.

Funcionalidad Unidades de medida.

- ✓ **RF7:** Mostrar unidades de medida.
- ✓ **RF8:** Registrar unidad de medida.
- ✓ **RF9:** Modificar unidad de medida.
- ✓ **RF10:** Eliminar unidad de medida.
- ✓ **RF11:** Definir equivalencia.
- ✓ **RF12:** Exportar a Excel.
- ✓ **RF13:** Exportar a PDF.

Funcionalidad Clasificación de unidades de medida.

- ✓ **RF14:** Mostrar clasificación de unidades de medida.
- ✓ **RF15:** Registrar clasificación de unidad de medida.
- ✓ **RF16:** Modificar clasificación de unidad de medida.
- ✓ **RF17:** Eliminar clasificación de unidad de medida.
- ✓ **RF18:** Configurar clasificación de unidad de medida.

- ✓ **RF19:** Exportar a Excel.
- ✓ **RF20:** Exportar a PDF.

Funcionalidad Tipos de producto.

- ✓ **RF21:** Mostrar tipos de producto.
- ✓ **RF22:** Registrar tipo de producto.
- ✓ **RF23:** Modificar tipo de producto.
- ✓ **RF24:** Eliminar tipo de producto.
- ✓ **RF25:** Exportar a Excel.
- ✓ **RF26:** Exportar a PDF.

Funcionalidad Categorías de tipos de productos.

- ✓ **RF27:** Mostrar categorías de tipos de productos.
- ✓ **RF28:** Registrar categoría de tipo de producto.
- ✓ **RF29:** Modificar categoría de tipo de producto.
- ✓ **RF30:** Eliminar categoría de tipo de producto
- ✓ **RF31:** Exportar a Excel.
- ✓ **RF32:** Exportar a PDF.

Funcionalidad Productos en almacén.

- ✓ **RF33:** Mostrar productos en almacén.
- ✓ **RF34:** Ver detalles de producto.
- ✓ **RF35:** Configurar producto.
- ✓ **RF36:** Exportar a Excel.
- ✓ **RF37:** Exportar a PDF.

Funcionalidad Planificación de menú.

- ✓ **RF38:** Mostrar planificación de menú.
- ✓ **RF39:** Crear planificación de menú.
- ✓ **RF40:** Editar planificación de menú

Requisitos funcionales del Módulo Abastecimiento.

Funcionalidad Productos disponibles.

- ✓ **RF41:** Mostrar productos disponibles.
- ✓ **RF42:** Modificar configuración del producto.
- ✓ **RF43:** Exportar a Excel.
- ✓ **RF44:** Exportar a PDF.

Funcionalidad Platos.

- ✓ **RF45:** Mostrar platos.
- ✓ **RF46:** Registrar plato.
- ✓ **RF47:** Asociar producto.
- ✓ **RF48:** Modificar plato.
- ✓ **RF49:** Ver detalles del plato.
- ✓ **RF50:** Activar plato.
- ✓ **RF51:** Desactivar plato.
- ✓ **RF52:** Exportar a Excel.
- ✓ **RF53:** Exportar a PDF.

Funcional Menú.

- ✓ **RF54:** Mostrar menú.
- ✓ **RF55:** Registrar menú.
- ✓ **RF56:** Ver detalles del menú.
- ✓ **RF57:** Activar menú.
- ✓ **RF58:** Desactivar menú.
- ✓ **RF59:** Descontar disponibilidad de productos.

La siguiente tabla muestra la descripción del requisito funcional Mostrar menú.

Tabla 2 Historia de Usuario. RF Mostrar menú.

Número:	51	Nombre del requisito: Mostrar menú.	
Programador:	Leidy Márquez Barbosa	Iteración Asignada: 1	
Prioridad:	Baja	Tiempo Estimado: 1	
Riesgo en Desarrollo:		Tiempo Real: 1	
Descripción:			
Se muestra el campo <i>Buscar</i> por el criterio nombre, las <i>Opciones de búsquedas</i> que contiene con el campo de selección <i>Fecha de inicio desde</i> , <i>Fecha de inicio hasta</i> , <i>Sujeto a cambio</i> , <i>Descontado de almacén</i> , <i>Activo</i> y la Tabla que muestra la lista de menú que han sido creados en el sistema, con los datos: <i>Fecha</i> , <i>Sujeto a cambio</i> , <i>Descontado</i> , <i>Activo</i> y la columna <i>Opciones</i> que permite Ver detalles, Activar, Desactivar y descontar de almacén con 2 días de antelación el plato según corresponda.			
Observaciones:			
Cuando no existen elementos creados se muestra un listado vacío y el mensaje: "No existen registros en el sistema".			
Prototipo elemental de interfaz gráfica de usuario:			

Menú +

▼ Opciones de búsqueda (click para mostrar/ocultar)

Fecha de inicio desde	Fecha de inicio hasta	Sujeto a cambio
<input style="width: 100%;" type="text" value="--Seleccione--"/>	<input style="width: 100%;" type="text" value="--Seleccione--"/>	<input style="width: 100%;" type="text" value="--Seleccione--"/>
Descontado de almacén	Activo	
<input style="width: 100%;" type="text" value="--Seleccione--"/>	<input style="width: 100%;" type="text" value="--Seleccione--"/>	

Lista de menú Registro por página

Fecha	Sujeto a cambio	Descontado	Activo	Opciones
1 23/05/2019	No	No	Si	
2 1/05/2019	No	Si	Si	
2 30/04/2019	No	No	Si	

« « Página 1 de 1 » »

Mostrando 1 - 4 de 4

2.3.2. Requisitos no funcionales

El sistema debe ser eficiente en el campo de acción en que se utiliza, no solo es importante especificar las funcionalidades que debe desempeñar, también se tienen que definir todas las cualidades que lo hacen más atractivo al cliente, usable, rápido y confiable. Los requisitos no funcionales se reconocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el *software*.

La siguiente tabla muestra la nomenclatura usada para clasificar los requisitos no funcionales identificados en el desarrollo del Módulo Abastecimiento:

Tabla 3 Nomenclatura de los requisitos no funcionales del Módulo Abastecimiento.

Categoría	Prefijo
Usabilidad	RNU
Confiabilidad	RNC
Eficiencia	RNE
Soporte	RNSE
Diseño e implementación	RNDI
Interfaz	RNI
Seguridad	RNS
Funcionamiento	RNF

La siguiente tabla muestra los requisitos no funcionales identificados en el desarrollo del Módulo Abastecimiento:

Tabla 4 Requisitos no funcionales del Módulo Abastecimiento.

Requerimiento	Descripción
RNU 1. Facilidad de empleo para usuarios sin experiencia.	El módulo debe tener una interfaz de fácil aprendizaje para que usuarios inexpertos puedan familiarizarse rápidamente.
RNU 2. Utilizar patrón de navegación.	El módulo permitirá el fácil acceso a las funcionalidades del mismo, brindando accesos directos a dichas funcionalidades mediante íconos flotantes e internos del sistema.

RNC 1. Disponibilidad.	El módulo contribuye a la muestra del menú, por lo cual debe estar disponible para el usuario en el momento en que lo necesite.
RNC 2. Salvas.	El módulo debe realizar salvas automáticas del código fuente con frecuencia mensual, para en caso de fallas por perdidas de datos, el sistema pueda ser recuperado a partir de las salvas realizadas.
RNE 1. Capacidad.	El sistema debe soportar al menos 300 usuarios conectados.
RNSE 1. Documentación.	Se documenta el módulo con un manual de usuario con el objetivo de explicar su uso.
RNDI 1. Lenguaje de programación.	Se utiliza para la construcción del módulo los lenguajes de programación php, HTML, JavaScript y las herramientas que se utilizan son de distribución bajo licencias libres, para favorecer el desarrollo de nuevos módulos o subsistemas.
RNDI 2. Forma de acceso.	Al sistema se puede acceder a través de un navegador web desde los sistemas operativos Windows y GNU/Linux.
RNDI 3. Entorno Integrado de Desarrollo.	Se utiliza NetBeans como entorno integrado de desarrollo y dicho IDE soporta php como lenguaje de programación.
RNI 1. Interfaces de usuario.	Para acceder al Sistema debe usarse una versión del navegador Mozilla/Firefox v 35.0 o superior. No se garantiza la correcta visualización en otros navegadores. Además, que para hacer uso de todas las funcionalidades el navegador debe tener habilitado el soporte para Java Script.
RNS 1. Protección de los datos almacenados.	El sistema debe garantizar la protección de los datos almacenados. Para ello se establecerán diferentes niveles de acceso, permitiendo que cada usuario acceda solamente a las funcionalidades que necesite. En el Módulo Abastecimiento esto se logra mediante las listas de control de acceso definidas en el fichero de configuración de Symfony security.yml

RNF 1. Requisitos de software.	El sistema debe integrarse con el Gestor de base de datos PostgreSQL 9.5, usando como servidor web Nginx.
RNF 2. Requisitos de hardware.	El hardware donde se instalará el sistema debe poseer al menos una Interfaz de red cuya velocidad de transferencia iguale o supere los 100 Mbps. El Servidor donde se instale el Sistema debe tener como mínimo un Microprocesador Intel-Core i3 a 3.0 Mhz3 o equivalente, 4 GB de memoria RAM y un espacio libre en Disco Duro de 2 GB.

2.4. Definición de los actores

Un actor es toda entidad externa al sistema que guarda una relación con el mismo y que le demanda una funcionalidad. Esto incluye a los operadores humanos, los sistemas externos y entidades abstractas como el tiempo. Un actor representa un rol en el sistema, no un usuario en específico del mismo (Vaillant y Ruiz, 2015). En el Módulo Abastecimiento interactúan 2 actores, los cuales se definen en la siguiente tabla:

Tabla 5: Actores del Módulo Abastecimiento.

Actor	Descripción
Usuario	<ul style="list-style-type: none"> ✓ Persona que interactúa con el sistema. ✓ Utiliza las funcionalidades que brinda el Módulo Abastecimiento. ✓ Consulta la información que le ha sido proporcionada.
Administrador	<ul style="list-style-type: none"> ✓ Persona que tiene acceso y permiso a todas las funcionalidades del sistema. ✓ Controla y otorga a otros usuarios el acceso y los permisos sobre las funcionalidades. ✓ Realiza las configuraciones generales y de seguridad para el uso del sistema.

2.5. Patrón Arquitectónico

Un patrón arquitectónico se puede considerar como una descripción abstracta estilizada de buena práctica, que se ensayó y puso a prueba en diferentes sistemas y entornos. Debe describir una organización de sistema que ha tenido éxito en sistemas previos. Debe incluir información sobre cuándo es y cuándo no es

adecuado usar dicho patrón, así como sobre las fortalezas y debilidades del patrón (Sommerville, 2011). El patrón arquitectónico utilizado para el desarrollo del Módulo Abastecimiento es el Modelo-Vista-Controlador (MVC).

El MVC es un patrón de arquitectura de software que separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí. El componente Modelo maneja los datos del sistema y las operaciones asociadas a esos datos. El componente Vista define y gestiona cómo se presentan los datos al usuario. El Controlador dirige la interacción del usuario (por ejemplo, teclas oprimidas, clics del mouse, etcétera) y pasa estas interacciones a Vista y Modelo (Sommerville, 2011).

El modelo: Debe representar la parte de la aplicación que implementa la lógica de negocio y el acceso a datos. Esto significa que debe ser responsable de la recuperación de datos convirtiéndolos en conceptos significativos para la aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos. En el desarrollo del Módulo Abastecimiento para hacer uso del modelo se explica a continuación:

- ✓ **Lógica de negocio:** se localizan las entidades y los repositorios. Las entidades son la representación de las tablas de la base de datos previamente mapeadas por el ORM *Doctrine* que se encuentran en *Src/systems/Alimentación/Configuración/AbastecimientoBundle/Entity*. Para abstraer la lógica de negocio de los controladores se crean repositorios de entidades personalizados que permiten llevar el manejo de la base de datos a una clase aislada del controlador y se pueden encontrar en la siguiente dirección:
Src/systems/Alimentación/Configuración/AbastecimientoBundle/Repository.
- ✓ **Acceso a datos:** establece la conexión con la base de datos, además, se encuentra ubicado el ORM *Doctrine*, que posibilita la separación en la aplicación del gestor de base de datos mediante su lenguaje propio de consultas SQL que se encuentra en *app/config/parameters.yml*.

La vista: Debe hacer una presentación de los datos del modelo estando separada de los objetos del modelo. Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente. En el desarrollo del Módulo Abastecimiento para la presentación de los datos se utiliza *twig* como motor de plantillas, la biblioteca *jQuery*, en conjunto con los archivos CSS, *JavaScript* y HTML, que se encargarán de estructurar y aplicar estilos a las interfaces

creadas. Dichos archivos se encuentran ubicados en el directorio */Resources/views* dentro de cada *bundle* de la aplicación en *Src/systems/Alimentación/Configuración/AbastecimientoBundle/Resouces/Views*.

El controlador: Gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda tanto del modelo como de la vista. Los controladores pueden ser vistos como administradores, espera peticiones de los clientes, comprueba su validez de acuerdo a las normas de autenticación o autorización, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado según las preferencias del cliente. Finalmente delega este proceso de presentación a la capa de la Vista. Para el desarrollo del Módulo Abastecimiento este patrón se pone de manifiesto en todas las clases *Controller*, que controlarán el flujo de información que se recibe y se envía hacia la vista y estarán ubicadas en *Src/systems/Alimentación/Configuración/AbastecimientoBundle/Controller* junto a las demás clases que se localizan en el directorio */Controller* de cada *buldle* de la aplicación.

2.6. Patrones de diseño

Un patrón de diseño se caracteriza como “una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución”. Para el diseño de *software*, el contexto permite al lector entender el ambiente en el que reside el problema y qué solución sería apropiada en dicho ambiente. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas que influyen en la manera en la que puede interpretarse el problema en este contexto y en cómo podría aplicarse con eficacia la solución (Pressman, 2009).

2.6.1. Patrones Generales de Asignación de Responsabilidades

Los patrones GRASP (*General Responsibility Assignment Software Patterns*, “Patrones de *Software* para la Asignación General de Responsabilidad”) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Se pueden destacar 5 patrones principales que son: Bajo Acoplamiento, Alta Cohesión, Experto, Creador y Controlador.

Experto: el objetivo es asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir con la funcionalidad. Los objetos se valen de su propia información para hacer lo que se les pide, esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas

más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase sencillas y más cohesivas que son fáciles de comprender y de mantener. La responsabilidad de realizar una labor es de la clase que tiene los datos involucrados (Pressman, 2009). En el Módulo Abastecimiento este patrón se pone en práctica en la clase MenúManager que es la encargada de proporcionar información necesaria para la gestión del menú.

Bajo Acoplamiento: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente y presentan los siguientes problemas:

- ✓ Los cambios de las clases afines ocasionan cambios locales.
- ✓ Son más difíciles de entender cuando están aisladas.
- ✓ Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

El Bajo Acoplamiento soporta el diseño de clases más independientes que reducen el impacto de los cambios, son más reutilizables y acrecientan la oportunidad de una mayor productividad (Pressman, 2009). Las clases contenidas en el Módulo Abastecimiento fueron implementadas de forma independientes, esto mitiga el impacto de los cambios que se puedan realizar en una clase.

Alta Cohesión: el objetivo es asignar una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuan relacionadas y enfocadas están todas las responsabilidades de una clase. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:

- ✓ Son difíciles de comprender.
- ✓ Son difíciles de reutilizar.
- ✓ Son difíciles de conservar.
- ✓ Son delicadas: las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos (Pressman, 2009). En el Módulo Abastecimiento este patrón se pone de manifiesto al definir a cada clase del módulo una función específica y única.

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. El uso de este patrón se refleja en las clases controladoras pertenecientes al sistema. En el Módulo Abastecimiento este patrón se pone en práctica en todas las clases controladoras. Ejemplo *ProductoController*, *AlmacénController*, *PlatoController*.

Creador: este patrón como su nombre lo indica es el que crea, ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos. Es donde se asigna la responsabilidad de que una clase B cree un objeto de la clase A. En el Módulo Abastecimiento este patrón se pone de manifiesto en todos los servicios definidos en los ficheros *service.yml*.

2.6.2. Patrones GOF

Los patrones "Banda de los Cuatro" (*Gang-of-Four*) describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas (Flores et al. 2014).

Instancia única (*Singleton*): es un patrón diseñado para restringir la creación de objetos pertenecientes a una clase u objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Este patrón se refleja en las clases controladoras. Ejemplo *ProductoController*, *AlmacénController*, *PlatoController*.

Mediador (*Mediator*): define un objeto que coordine la comunicación entre objetos de distintas clases. Se refleja en las librerías que funcionan como mediadoras entre las clases controladoras y los modelos de acceso a datos. Este patrón se refleja en las clases *manager*. Ejemplo *MenúManager*.

2.7. Modelo de la base de datos

Modelo: conjunto de conceptos que permite construir una representación organizacional de una institución.

Universo de discurso: visión del mundo real del diseñador de la Base de Datos. Su definición, constituye el primer paso del diseño para poder conseguir los objetivos del diseñador.

Modelo de Datos: conjunto de conceptos, reglas y convenciones que permiten describir, a distintos niveles de abstracción, los datos del Universo de Discurso o la estructura de una base de datos, la cual es denominada esquema, o, con otras palabras, que permiten construir una representación organizada de un sistema real. Por lo tanto, se considera un modelo de datos como una herramienta que facilita la interpretación del Universo de Discurso y la representación de un sistema de información (Vaillant y Ruiz 2015).

A continuación se muestra en la Figura 2, un fragmento del modelo de datos del Módulo Abastecimiento, el cual ofrece una descripción abstracta sobre la representación de los datos en un Sistema Gestor de Base de Datos (SGBD). Los componentes que lo conforman, son entidades existentes en el módulo. Este modelo contiene a su vez, características propias de estos objetos y la forma en que se relacionan entre sí. Para visualizar en su totalidad el modelo de datos ver Anexo 3.

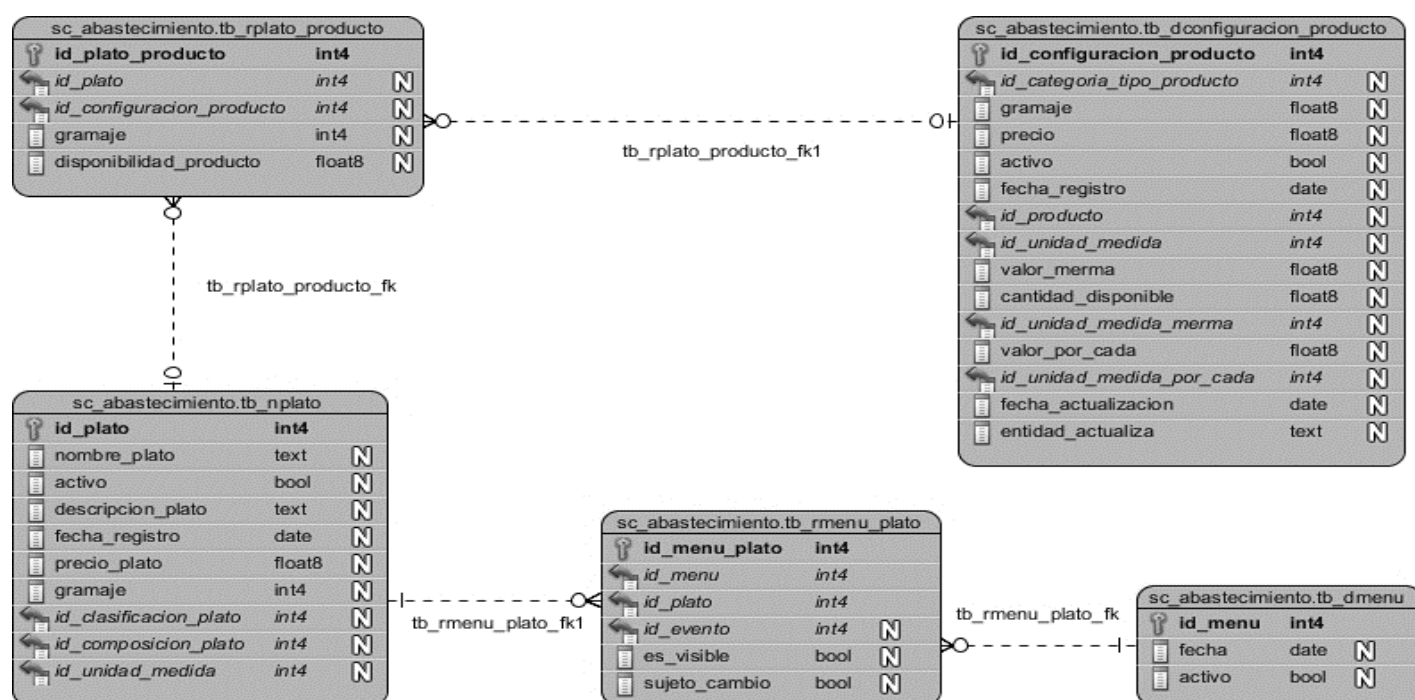


Figura 2 Fragmento del modelo de datos del Módulo Abastecimiento.

Descripción de las tablas del modelo de datos

Almacén: Gestiona la información de los almacenes.

Producto_almacén: Relación entre las tablas Almacén y Producto.

Producto: Almacena los productos que se encuentran disponibles en cada uno de los almacenes.

Clasificación_producto: Almacena las clasificaciones de un producto definidas por el usuario.

Tipo_producto: Almacena los tipos producto definidos por el usuario.

Categoría_tipo_producto: Almacena las categorías definidas por el usuario.

Configuración_producto: Almacena las configuraciones de los productos.

Unidad_medida: Almacena las unidades de medida definidas por el usuario.

Clasificación_Unidad_medida: Almacena las clasificaciones de las unidades de medida definidas por el usuario.

Platos: Almacena los platos definidos por el usuario.

Plato_producto: Relación entre las tablas Menú y Plato.

Eventos: Almacena los eventos definidos por el usuario.

Plato_evento: Almacena los platos que están asignados a un determinado evento.

Clasificación_plato: Almacena las clasificaciones de un plato definidas por el usuario.

Menú: Almacena los menús definidos por el usuario.

Menú_plato: Relación entre las tablas Menú y Platos.

Fecha: Almacena la fecha de la configuración de un producto.

2.8. Diagrama de despliegue

El diagrama de despliegue se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el *hardware* y el *software* que se despliega, estas relaciones son representadas por los protocolos de comunicación que se utilizan para acceder a cada uno. En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta:

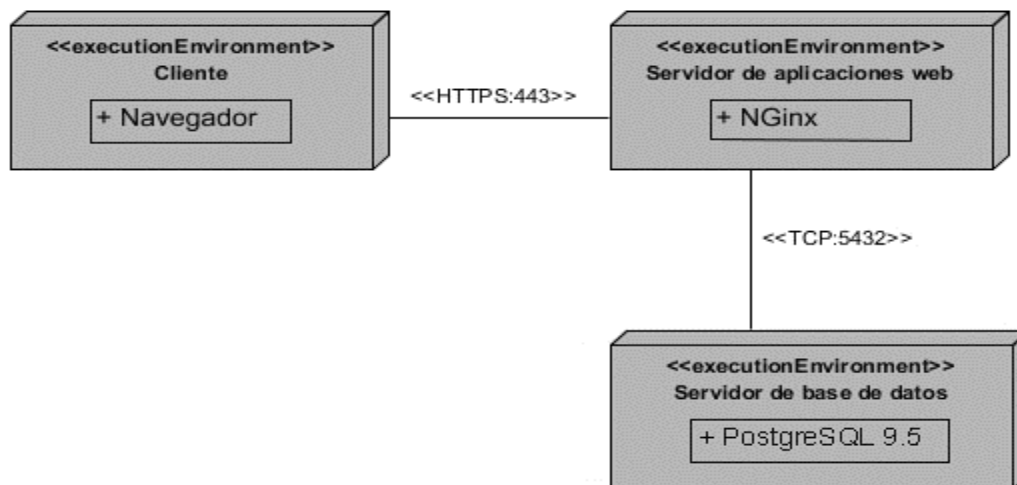


Figura 3 Diagrama de despliegue del Módulo Abastecimiento.

El diagrama de despliegue representado muestra la siguiente distribución:

Cliente: dispositivo cliente capaz de conectarse al servidor de aplicaciones mediante el protocolo de comunicaciones HTTPS.

Servidor de aplicaciones web: computadora en que se encuentra el servidor web *NGinx*, este será el lugar en que se gestione todo el contenido de la aplicación. El mismo establecerá comunicación con los ordenadores clientes mediante protocolo HTTPS y con el servidor de base de datos por medio del protocolo TCP.

Servidor de base de datos: ordenador en que se encuentra el gestor de base de datos *PostgreSQL* capaz de mantener persistente la información generada y a utilizar.

HTTPS: protocolo de transferencia de hipertexto seguro, por sus siglas en inglés, *Hypertext Transfer Secure Protocol* (HTTPS), es un protocolo de red basado en HTTP por lo que está orientado a transacciones sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente y un servidor (Forouzan, 2015).

TCP/IP: base de Internet y sirve para enlazar computadoras que utilizan diferentes sistemas operativos. Familia de protocolos utilizada para la conexión entre el servidor web y el servidor donde se encuentra ubicada la base de datos (Forouzan, 2014).

2.9. Conclusiones parciales

Una vez finalizado este capítulo, se arribaron a las siguientes conclusiones:

- ✓ Se determinaron los requisitos funcionales y no funcionales constituyendo una guía fundamental para el desarrollo del Módulo Abastecimiento.
- ✓ Con la realización de modelo de datos se obtuvo una descripción abstracta sobre la representación de los datos en el Sistema Gestor de Base de Datos referente al Módulo Abastecimiento.
- ✓ Con la utilización de la arquitectura Modelo-Vista-Controlador y el uso de los patrones de diseño se garantizará una mayor organización y código más legible.
- ✓ Con la realización del modelo de despliegue se ilustra la comunicación de los distintos componentes en los cuales se divide el sistema.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se propone explicar a partir de los resultados del diseño, la implementación del Módulo Abastecimiento para el Sistema Integrado de Alimentación. Se definen los estándares de codificación y las pruebas de *software* como elemento esencial que garantiza la calidad del módulo y revisión final del cumplimiento de los requisitos, especificaciones de diseño y codificación.

3.1. Estándares de codificación

Los estándares de codificación son reglas de codificación que permiten tener una programación homogénea, comprendiendo todos los aspectos de la generalización del código, pues la aplicación debe de estar implementada como si un único programador escribiera el código de una sola vez. La usabilidad de estos permite conservar el código fuente entendible y fácil de mantener, además de mejorar la forma en la que se programa (Laínez, 2015). Con vistas a mejorar el entendimiento del código perteneciente al Módulo Abastecimiento para el Sistema Integrado de Alimentación por otros desarrolladores y alcanzar una uniformidad en el mismo, a continuación, se muestran algunos estándares de codificación utilizados en la implementación.

Indentación, llaves de apertura y cierre y tamaño de líneas

El código debe usar cuatro espacios para la indentación en vez de usar el tabulado, esto minimiza problemas con otras herramientas de desarrollo.

Las líneas podrían tener 80 caracteres o menos evitando tener más de 120 caracteres. Las llaves de apertura deben ir en la siguiente línea y la llave de cierre debe ir en la siguiente línea después del cuerpo.

Los paréntesis en las estructuras de control, no deben usar espacios antes ni después. Se añade un solo espacio después de cada limitador de coma y alrededor de los operadores (`==`, `&&`, ...).

Añadir una línea en blanco antes de una declaración de *return*, a no ser que esté dentro de una declaración como un grupo como *if*.

```
public function activarDesactivarAction(DConfiguracionProducto $idConfiguracionProducto, Request $request)
{
    $em = $this->getDoctrine()->getManager();
    $id_almacen= $request->getSession()->get('pAlmacen');
    if ($idConfiguracionProducto->isActivo() == 1) {
        $idConfiguracionProducto->setActivo( activo: 0);
        $em->persist($idConfiguracionProducto);
        $em->flush();
    } else {
        $idConfiguracionProducto->setActivo( activo: 1);
        $em->persist($idConfiguracionProducto);
        $em->flush();
    }
    $message = $this->get('translator')->trans('app.msg.inf_success');

    return self::showMessage( success: false, $message, array());
}
```

Figura 4 Indentación, llaves de apertura y cierre y tamaño de líneas.

Convención de nomenclatura

Variables: Se rigen por la nomenclatura *camelCase*¹ para declarar. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Se muestra en el método un ejemplo donde se emplean las variables:

```
public function exportPdfAction(Request $request)
{
    $handlerFop = $this->get('inagbe_app.handler_fop');
    $filters = $request->getSession()->get('filters_list_ready_product', NULL);

    return $handlerFop->exportToPdf(new ExportProductReady($filters));
}
```

Figura 5 Convención de nomenclatura: variable camelCase.

¹ *camelCase*: Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en *camelCase* se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.

Clases: Se rigen por la nomenclatura *StudyCaps*². El patrón que sigue para declarar las clases es que siempre comienzan con mayúscula y en caso de nombre compuesto cada palabra comienza en mayúscula, sin espacios o guion bajo, como se muestra a continuación:

```
namespace Systems\Alimentacion\Configuracion\AbastecimientoBundle\Controller;
use ...

class ProductTypeCategoryController extends ApplicationController
{
```

Figura 6 Convención de nomenclatura. Clase nombre compuesto. StudyCaps.

Funciones: Se rigen por la nomenclatura *camelCase*. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa como se muestra a continuación:

```
public function updateAction(NCategoriaTipoProducto $idCategoriaTipoProducto, Request $request)
{
    $form = $this->createForm(new NCategoriaTipoProductoType(), $idCategoriaTipoProducto);
    $form->handleRequest($request);
}
```

Figura 7 Convención de nomenclatura: función: camelCase.

Estructuras de control

Las estructuras de control incluyen *if*, *for*, *for each*, *while*, *switch*, entre estas estructuras y los paréntesis que encierran la condición debe de existir un espacio. Es recomendable utilizar en cualquier caso llaves de apertura y cierre al comienzo de una nueva línea, incluso en situaciones en las que técnicamente son opcionales como se ejemplifica a continuación:

² StudyCaps es una forma de notación de código en el que la capitalización de letras varía según un patrón, arbitrariamente, por lo general también omitiendo espacios entre las palabras y, a menudo omitiendo algunas letras.

```
if ( empty($exist) || $exist[0]->getNombreCategoriaTipoProducto() == $idCategoriaTipoProducto->getNombreCategoriaTipoProducto() ) {  
    $id = $form->getData();  
    $em->persist($id);  
    $em->flush();  
}
```

Figura 8 Estructuras de control.

Documentación

Todos los archivos deben tener su documentación asociada. Se debe de cumplir con el siguiente bloque al principio de cada clase, como se muestra a continuación:

```
<?php  
/**  
 * Created by NetBeans  
 * User: Leidy  
 * Date: 23/1/2019  
 * Time: 9:07 AM  
 */  
  
namespace Systems\Alimentacion\Configuracion\AbastecimientoBundle\Entity;  
  
use Doctrine\ORM\Mapping as ORM;  
  
/**  
 * DAlmacen  
 *  
 * @ORM\Table(name="sq_asset.tb_dalmacen")  
 * @ORM\Entity(repositoryClass="DAlmacenRepository")  
 */  
class DAlmacen  
{
```

Figura 9 Ejemplo de archivo documentado.

3.2. Validación de requisitos

La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proyecto y el producto (Ramos et al. 2015).

3.2.1. Criterios para validar los requisitos

Para validar los requisitos del sistema según Pressman, se pueden chequear mediante un cuestionario guiado por un conjunto de interrogantes, con el objetivo de descubrir la mayor cantidad de errores posibles (Pressman, 2009).

A continuación, se muestran las preguntas utilizadas, algunas fueron agregadas por la autora de la presente investigación a las planteadas por Pressman por ser consideradas necesarias para la validación.

Interrogantes para la validación de requisitos:

- ¿Está el requisito claramente definido?
- ¿Puede interpretarse mal?
- ¿Está identificado el origen del requisito (por ejemplo: persona, norma, documento)?
- ¿El planteamiento final del requisito ha sido contrastado con la fuente original?
- ¿El requisito está delimitado en términos cuantitativos?
- ¿Qué otros requisitos hacen referencia al requisito estudiado?
- ¿El requisito incumple alguna restricción definida?
- ¿El requisito es verificable? Si es así, ¿se pueden efectuar pruebas para verificar el requisito?
- ¿Se puede seguir el requisito en el modelo del sistema que se ha desarrollado?
- ¿Está el requisito asociado con los rendimientos del sistema o con su comportamiento?
- ¿El requisito está implícitamente definido?
- ¿El requisito es modificable?
- ¿El requisito está completo?
- ¿El requisito puede ser implementado?
- ¿El requisito puede ser probado?
- ¿El resultado de la evaluación de impacto es positivo?

Resultado de aplicar los criterios de validación

Luego de aplicar el conjunto de interrogantes para validar los requisitos definidos para el desarrollo del sistema, se obtuvo el 100 % de aprobación por parte del cliente.

3.2.2. Técnicas de validación de requisitos

Con el objetivo de obtener una mayor calidad y demostrar que los requisitos definidos realmente describen el sistema que el cliente necesita; se utilizaron las técnicas para la validación de requisitos siguientes:

Prototipado de interfaz: permite al cliente entender fácilmente la propuesta de solución, al brindar la representación aproximada de la interfaz de usuario que tendrá el sistema. Existen dos tipos principales de prototipo de interfaz:

- ✓ **Desechables:** se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en *software*.
- ✓ **Evolutivos:** una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final. El tipo utilizado finalmente para la propuesta de solución fue Evolutivos pues de esta forma se reutiliza el prototipo diseñado en todo el proceso de desarrollo del sistema.

Generación de casos de prueba: se realizaron los diseños de casos de pruebas para cada uno de los requisitos obtenidos, permitiendo verificar que todos se pudieran probar y determinar en la medida de la complejidad del diseño de caso de prueba, identificando requisitos que deberían ser reconsiderados y cuáles pueden ser los más difíciles de implementar.

Resultado de aplicar las técnicas de validación

Como resultado de este proceso se identificaron inconsistencias en las especificaciones tales como la falta de concordancia entre la complejidad de la especificación y la registrada en el documento de Evaluación de requisitos. Descripciones de requisitos poco detalladas o ambiguas y numerosos errores ortográficos.

3.3. Pruebas

Para evaluar la calidad del sistema que se está desarrollando y verificar el cumplimiento de los objetivos trazados, se aplicaron un conjunto de pruebas definidas por Pressman en su libro de Ingeniería del software

“Un enfoque práctico”, en su séptima edición. A continuación, se muestra la estrategia de prueba diseñada para aplicar en la solución desarrollada:

Tabla 6 Pruebas realizadas.

Prueba	Método	Técnica
Prueba de unidad	Caja blanca	Camino Básico
Prueba de validación	Caja negra	Partición de equivalencia
Prueba de sistema	Caja negra	Automático

3.3.1. Prueba de unidad

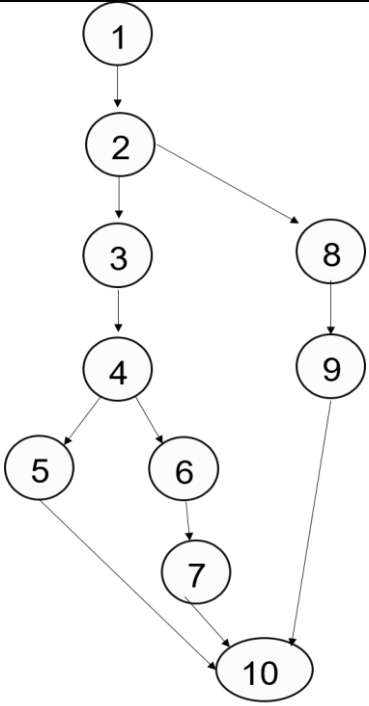
Las pruebas de unidad tienen como objetivo verificar la unidad más pequeña del diseño del *software*. Estas pruebas se concentran en la lógica del procesamiento interno y en las estructuras de datos tales como: código fuente, archivos binarios, archivos de datos, entre otros. Este tipo de prueba se puede aplicar en paralelo a varios componentes (Gómez y Moraleda, 2014).

Las pruebas de unidad se realizan mediante el **método de caja blanca o estructural**, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. La técnica de prueba de caja blanca utilizada en la investigación es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Sommerville, 2016).

Complejidad ciclomática

La Complejidad Ciclomática se realiza a todos los métodos o algoritmos de un sistema informático. A continuación, se presenta la prueba realizada al método *createAction (request \$request)* ya que posee importancia en el contexto de la investigación, el cual posibilita que se cree un nuevo plato.

Tabla 7 Prueba de unidad.

Fórmula 1	Fórmula 2	Fórmula 3	Grafo resultante
$V(G) = (A - N) + 2$ $V(G) = 11 - 10 + 2$ $V(G) = 3$	$V(G) = P + 1$ $V(G) = 2 + 1 = 3$	$V(G) = R$ $V(G) = 3$	 <pre> graph TD 1((1)) --> 2((2)) 2 --> 3((3)) 3 --> 4((4)) 4 --> 5((5)) 4 --> 6((6)) 5 --> 10((10)) 6 --> 7((7)) 7 --> 10 2 --> 8((8)) 8 --> 9((9)) 9 --> 10 </pre>
<p>A: es la cantidad de aristas. N: la cantidad de nodos. P: es el número de nodos predicado contenidos en el grafo de flujo G R: representa la cantidad de regiones en el grafo</p>			
<p>Complejidad ciclomática</p>			
<p>Como se puede observar, después de aplicadas las fórmulas 1, 2 y 3 a la funcionalidad createAction, posee una complejidad ciclomática igual a 3, lo cual demuestra que el grafo está bien diseñado y que existen 3 caminos lógicos e independientes.</p>			
<p>Prueba de camino básico</p>			
<p>Una vez calculada la complejidad ciclomática se define como límite superior 3, lo que indica que hay que realizarle al código tres pruebas, para garantizar que este se ejecute completamente al menos una vez. La funcionalidad createAction posee poco riesgo debido a que el resultado arrojado por la métrica pertenece al intervalo entre 1 -10.</p> <p>El total de caminos independientes establecidos fue de 3 y a continuación se muestran:</p> <ul style="list-style-type: none"> ✓ Camino 1: 1, 2, 3, 4, 6, 7, 10. <p>El primer camino comienza con la creación del formulario y la obtención de los datos enviados en la variable request, seguidamente verifica si el formulario es válido y si el elemento a crear existe, si se</p>			

cumplen estas dos condiciones no se crea el elemento en base de datos y se retorna el mensaje del elemento ya existe.

✓ Camino 2: 1, 2, 8, 9, 10.

El segundo camino comienza con la creación del formulario y la obtención de los datos enviados en la variable request, seguidamente verifica si el formulario es válido, si no se cumple esta condición no se crea el elemento en base de datos y se retorna el mensaje de error en el formulario.

✓ Camino 3: 1, 2, 3, 4, 5, 10.

El tercer camino comienza con la creación del formulario y la obtención de los datos enviados en la variable request, seguidamente verifica si el formulario es válido y si el elemento a crear no existe, si se cumplen estas condiciones se crea el elemento en base de datos y se retorna el mensaje de la acción ha sido realizada satisfactoriamente.

```

public function createAction(Request $request)
{
    $form = $this->createForm(new DishClassificationType(), new NClasificacionPlato());
    $form->handleRequest($request);

    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $post = $request->request->all();

        $exist = $em->getRepository( className: 'SystemsAlimentacionConfiguracionAbastecimientoBundle:NClasificacionPlato' )->findBy(array(
            'nombreClasificacionPlato' => $post['dishClassification']['nombreClasificacionPlato']
        ));
        if (empty($exist)) {
            $newEntity = $form->getData();
            $newEntity->setFechaRegistro(new \DateTime(date( format: "Y-m-d H:i:s")));
            $em->persist($newEntity);
            $em->flush();
            $message = $this->get('translator')->trans('app.msg.inf_success');
        } else {
            $message = $this->get('translator')->trans('app.msg.existent_item');
        }
    } else {
        $message = $this->get('translator')->trans('app.msg.inf_error_form');
    }

    return self::showMessage( success: true, $message, array(),
        $this->generateUrl( route: 'systems_alimentacion_configuracion_abastecimiento_dish_classification_index'));
}

```

3.3.2. Prueba de validación

La validación se consigue cuando el *software* funciona de acuerdo con las expectativas razonables del cliente. Una vez que se procede con cada prueba de validación, puede darse una de las dos condiciones siguientes: primera, las características de funcionamiento o de rendimiento estén de acuerdo con las especificaciones y son aceptables; o segunda, se descubre una desviación de las especificaciones y se crea una lista de deficiencias (Sánchez, 2015). Para el desarrollo de las pruebas de validación se aplicó el **método de caja negra** con la **técnica de partición de equivalencia**, donde se demuestra la conformidad de los requisitos.

Los **métodos de caja negra**, también denominados pruebas de comportamiento, se centran en los requisitos funcionales del *software*. O sea, permite al ingeniero del *software* obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a base de datos externas, errores de rendimiento y errores de inicialización y de terminación (Mera, 2016).

La técnica de **partición de equivalencia** consiste en ejecutar el flujo básico de las funcionalidades utilizando datos válidos e inválidos, donde se generaran los artefactos “Diseño de casos de pruebas” (Mera, 2016). Por cada requisito funcional del sistema se generó un documento donde se recogen todos los datos necesarios para probar la interfaz.

Se ejecutaron un total de cincuenta y nueve (59) casos de pruebas, a los cuales se le realizaron tres (3) iteraciones, hasta garantizar la eliminación de las no conformidades. A continuación, se muestra un ejemplo del caso de prueba (CP) para el requisito funcional Registrar clasificación de plato y una tabla con la información de los resultados obtenidos de las pruebas en cada iteración:

Tabla 8 CP Registrar clasificación de plato.

Escenario	Descripción	V1-Nombre	V2-Descripción	V1-Activo	Respuesta del sistema	Flujo central

EC1.1	Mediante este escenario se inserta en el sistema una nueva clasificación.	V Gran os	V Bu3en plato11	V Sele ccio nado	El sistema crea la nueva clasificación y muestra el mensaje: "La acción ha sido realizada satisfactoriamente."	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo de Configuración y luego el módulo Abastecimiento. El sistema muestra las opciones de menú. El usuario selecciona en la agrupación funcional "Clasificación de platos" la funcionalidad "Registrar clasificación de platos". El sistema muestra las clasificaciones registradas. El usuario selecciona en el área de iconos flotantes la opción: Registrar, llena los datos correctamente y presiona el botón Aceptar.
EC 1.2	Mediante este escenario se introducen datos para insertar una clasificación que ya existe en el sistema.	I Gran os	NA Bu3en plato11	NA Sele ccio nado	El sistema muestra un mensaje de error indicando: El elemento ya existe.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo de Configuración y luego el módulo Abastecimiento. El sistema muestra las opciones de menú. El usuario selecciona en la agrupación funcional "Clasificación de platos" la funcionalidad "Registrar clasificación de platos". El sistema muestra las clasificaciones registradas. El usuario selecciona en el área de iconos flotantes la opción: Registrar, llena los datos de un elemento que ya esté en el sistema y presiona el botón Aceptar.
EC 1.3	Mediante este escenario	I	NA	NA	El sistema muestra debajo del	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo de Configuración y

incompletos.	no se introducen todos los datos para registrar una clasificación.	Vacío	Bu3en plato11	Seleccionado	componente un mensaje en color rojo indicando: Campo requerido.	luego el módulo Abastecimiento. El sistema muestra las opciones de menú. El usuario selecciona en la agrupación funcional "Clasificación de platos" la funcionalidad "Registrar clasificación de platos". El sistema muestra las clasificaciones registradas. El usuario selecciona en el área de iconos flotantes la opción: Registrar, llena los datos de un elemento de forma incompleta y presiona el botón Aceptar.
EC 1.4 Insertar datos incorrectos.	Mediante este escenario se introducen datos incorrectos.	Ig	NA Bu3en plato11	NA Seleccionado	El sistema muestra en rojo el mensaje debajo del componente : Por favor no escriba menos de 2 caracteres.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo de Configuración y luego el módulo Abastecimiento. El sistema muestra las opciones de menú. El usuario selecciona en la agrupación funcional "Clasificación de platos" la funcionalidad "Registrar clasificación de platos". El sistema muestra las clasificaciones registradas. El usuario selecciona en el área de iconos flotantes la opción: Registrar, llena los datos de un elemento de forma incorrecta y presiona el botón Aceptar.
		I Se escriben más de 250 cara	NA Bu3en plato11	NA Seleccionado	El sistema muestra en color rojo encima del componente el mensaje: Por favor no escriba más	

		ctere s			de 250 caracteres.	
		NA Gra mos	I Se escribe más de 250 caracte res	NA Sele ccio nado	El sistema no permite la entrada de más caracteres.	
		NA Gra mos	Se escribe n más de 30 caracte res por palabra .	NA Sele ccio nado	El sistema muestra en rojo encima del componente : Ha excedido el número de caracteres permitidos para una palabra.	
EC 1.5 Cancelar operación.	Se cancela la creación de la nueva clasificaci ón.	NA Gra mos	NA Bu3en plato11	NA Sele ccio nado	El sistema muestra el mensaje de confirmació n: ¿Está seguro de realizar la	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo de Configuración y luego el módulo Abastecimiento. El sistema muestra las opciones de menú. El usuario selecciona en la agrupación funcional "Clasificación de platos" la funcionalidad

					<p>acción? Si el usuario presiona el botón Si el sistema retorna a la página que le dio inicio sin guardar las últimas operaciones Si el usuario presiona el botón No se mantiene en la misma vista, mostrando las últimas operaciones realizadas</p>	<p>"Registrar clasificación de platos". El sistema muestra las clasificaciones registradas. El usuario selecciona en el área de iconos flotantes la opción: Registrar, llena o no los datos y presiona el botón Aceptar.</p>
--	--	--	--	--	---	--

Tabla 9 Descripción de las variables del CP Registrar clasificación de plato.

No	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Nombre de la clasificación	Campo de texto	No	Es un campo requerido, admite espacios antes de las palabras, acepta cualquier tipo de carácter, la cantidad

				de caracteres por palabra es de 30 y la cantidad mínima es 2 y máxima 250.
2	Descripción de la clasificación	Campo de texto	Si	Admite espacios antes de las palabras, acepta cualquier tipo de carácter, la cantidad de caracteres por palabra es de 30 y la cantidad mínima es 2 y máxima 250.
3	Activo	Casilla de verificación	Si	Activa o desactiva la clasificación

Tabla 10 Iteraciones de los casos de prueba.

Iteración	Total de no conformidades	Asociadas a
1	25	Errores de interfaz, validación y ortografía.
2	10	Errores de interfaz y validación.
3	0	No se detectaron errores.

3.3.3. Pruebas de sistema

Las pruebas de sistema están constituidas por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas (Sommerville, 2016). Entre las pruebas de sistema que se ejecutan están las pruebas de rendimiento y resistencia.

Pruebas de rendimiento: están diseñadas para probar el rendimiento del *software* en tiempo de ejecución dentro del contexto de un sistema integrado. Lo que posibilita determinar cuán rápida es la respuesta del sistema ante un conjunto de peticiones concurrentes (Toledo, 2014).

Pruebas de resistencia: están diseñadas para enfrentar a los programas con situaciones anormales. La prueba de resistencia ejecuta un sistema de forma que demande recursos en cantidad o volúmenes anormales (Toledo, 2014).

Resultados de las pruebas de rendimiento y resistencia

Una prueba de rendimiento y resistencia se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. El sistema se prueba con un número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la prueba. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Para llevar a cabo las pruebas de rendimiento y resistencia se utilizó la herramienta JMeter 2.12. La prueba consistió en realizar a una funcionalidad tres pruebas de 50, 100 y 200 hilos, los cuales simulan 50, 100 y 200 accesos de usuarios respectivamente. Se definió una lista de enlaces a los que se simuló el acceso aleatorio y a partir de ahí, se recolectaron los datos necesarios para su interpretación.

Para un mejor entendimiento de los datos que se verán a continuación, se explica cada parámetro que compone la tabla.

#Muestras: cantidad de hilos utilizados para la URL.

Media: tiempo promedio en milisegundos para un conjunto de resultados.

Min: tiempo mínimo que demora un hilo en acceder a una página.

Max: tiempo máximo que demora un hilo en acceder a una página.

Pet/seg: hace referencia al número de peticiones que el servidor puede procesar en un segundo.

Kb/seg: rendimiento medido en *Kilobytes* por segundo.

A continuación, se muestran en la Tabla 11 los resultados obtenidos con la herramienta:

Tabla 11 Resultados de las pruebas de rendimiento y resistencia.

Aplicado a	Cantidad de hilos	Tiempo de ejecución (ms)			Rendimiento		
		Mín.	Máx.	Media	% Error	Pet/seg	Kb/seg
Crear salida	50	232	1683	963	0.00%	18.6 seg	133.2
	100	471	1724	1116	0.00%	25.8 seg	185.2
	200	624	1832	1358	0.00%	36.2 seg	231.3

3.4. Conclusiones parciales

- ✓ Se detallaron los estándares de codificación empleados en la implementación del Módulo Abastecimiento permitiendo una mejor organización y comprensión del código.
- ✓ Con los resultados de la validación de los requisitos identificados se logró obtener un listado de requisitos correctamente redactados y descritos.
- ✓ La descripción de las pruebas utilizadas para asegurar la calidad del *software*, permitió obtener resultados satisfactorios, asegurando que el sistema implementado no contiene errores y tiene la aceptación requerida.

CONCLUSIONES GENERALES

Al terminar la presente investigación se dio solución a la problemática planteada, contribuyendo a una mejor gestión del abastecimiento en el proceso de alimentación, se obtuvieron resultados que permiten arribar a las siguientes conclusiones:

- ✓ Con el estudio de soluciones informáticas homólogas se identificaron las principales tendencias en el desarrollo de sistemas de gestión de abastecimiento. Aunque los sistemas estudiados no brindan una solución directa al problema de la investigación posibilitó identificar funcionalidades básicas comunes y de valor agregado para la definición de los requisitos del producto final.
- ✓ El análisis de los principios teóricos de la investigación, enfocados a los sistemas de gestión de abastecimiento, permitió un mejor entendimiento de los procesos de este tipo en la universidad.
- ✓ El empleo de un proceso de desarrollo con enfoque ágil, herramientas y tecnologías libres favoreció el trabajo del equipo y como resultado se obtuvo un sistema que cumple con los requerimientos definidos.
- ✓ Se generaron los artefactos correspondientes al modelado del negocio donde se describieron y se diseñaron las funcionalidades del proceso.
- ✓ Con la adopción de un estándar de código se logró la organización, legibilidad y comprensión del código generado en la implementación del sistema propuesto.
- ✓ El diseño y ejecución de los casos de prueba, permitieron identificar errores de implementación en la aplicación y darle cumplimiento al objetivo general de esta investigación.

RECOMENDACIONES

Para garantizar el perfeccionamiento progresivo de la solución propuesta se propone la siguiente recomendación que tributará a un producto de mejor calidad:

- ✓ Realizar el diseño *reponsive* o adaptativo del sistema.

REFERENCIAS BIBLIOGRÁFICAS

1. The jQuery Foundation. 2018. jQuery write less, do more. [En línea] 2018. <https://jquery.com/>.
2. ARENCIBIA, Á.M. y JIMÉNEZ, Q.J.Y., 2014. Desarrollo del módulo de Distribución y Pedido de productos para el Sistema Integral de Gestión Cedrux. S.I.: Universidad de las Ciencias Informáticas. pp 5-8
3. AYOZE, C.A., 2017. *Curso de Programación Web. JavaScript, Ajax y jQuery*. S.I.: s.n.
4. CAMPOVERDE, A.M. y HERNÁNDEZ, D.L., 2015. Cloud computing con herramientas open -source para Internet de las cosas. . Ecuador: pp 1-5
5. EGUILUZ, J., 2016. Desarrollo web ágil con Symfony2. S.I.: s.n. pp 19-21, 451-453
6. FERNÁNDEZ, R.R. y VARONA, C.A., 2016. Sistema para la gestión de las transportaciones nacionales de la Universidad de las Ciencias Informáticas. S.I.: Universidad de las Ciencias Informáticas. pp 22-24, 44-46
7. FLORES, A., REYNOSO, L. y MOORE, R., 2014. A Formal Model of Object-Oriented Design and GoF Design Patterns. . S.I.:
8. FOROUZAN, B.A., 2014. TCP/IP Protocol Suite. S.I.: s.n. pp 18-22
9. FOROUZAN, B.A., 2015. *Data Communications AND Networking*. S.I.: s.n.
10. GÓMEZ, P.S.R. y MORALEDA, G.E., 2014. Aproximacion a la Ingeniería del Software. S.I.: s.n. pp 50-54
11. GONZÁLEZ, J.L. y GIMENO, J.M., 2011. Introducción a Netbeans. . S.I.: pp 1-4
12. GONZÁLVEZ, G.N., ANTÓNIO TRIGO, PEDRO SOTO-ACOSTA y FRANCISCO JOSÉ MOLINA CASTILLO, 2009. El papel de las TIC en el rendimiento de las cadenas de suministro: el caso de las grandes empresas de España y Portugal. , pp. 2, 15.
13. HERNÁNDEZ, D.N. y CASTRO, S.F.A., 2007. APLICACIÓN WEB PARA GESTIONAR Y MONITOREAR LA MIGRACIÓN DE DATOS DEL SISTEMA ASSETS AL SISTEMA TRABAJADORES EN LA UCI. S.I.: Universidad de las Ciencias Informáticas. pp 10-18

14. HERNÁNDEZ, R.N.R., 2011. Diseño de un modelo general para la gestión de sistemas logísticos en empresas cubanas: consideraciones teóricas y prácticas. . Cuba: pp 1-4
15. LAÍNEZ, F.J.R., 2015. Desarrollo de Software Ágil. S.I.: s.n. pp 71-79
16. LEÓN, R.J.D. y BARBAIMON, L.G.M., 2014. Sistema Web Móvil para realizar reservas de menú en el Centro de Practicas Pre- Profesionales de alimentos y bebidas El Mesón del Estudiante de la Universidad Ricardo Palma. S.I.: Universidad Ricardo Palma. pp 47-56
17. MARTÍNEZ, S.M. y CONCEPCIÓN, R.D., 2014. Sistema de control de acceso de los comedores de la Universidad de las Ciencias Informáticas basado en plataformas libres. S.I.: Universidad de las Ciencias Informáticas. pp 1
18. MEDRANO, H.M.E. y RODRÍGUEZ, A.P.C., 2013. DISEÑO DE UN SISTEMA INFORMÁTICO WEB DE GESTIÓN DE PEDIDOS Y ABASTECIMIENTO DE MATERIALES PARA LA EMPRESA PROYERSAC UTILIZANDO METODOLOGÍA RUP. Perú: s.n. pp 21-32
19. MERA, P.J.A., 2016. Análisis del proceso de pruebas de calidad de software. . Colombia:
20. OTERO, P.M.A., 2012. Diseño de una propuesta de gestión de abastecimiento e inventarios para un astillero en Colombia. Colombia: Universidad Nacional de Colombia. pp 4-10
21. PÉREZ, M.M., 2010. Software de prototipado para la arquitectura de la información: funcionalidad y evaluación. . S.I.: pp 1-5
22. PÉREZ, M.S., 2014. Ingeniería Técnica en Informática de Gestión. S.I.: Universidad Carlos III de Madrid. Departamento de Informática. pp 7-13
23. PRESSMAN, R.S., 2009. Software Engineering. A practitioner's Approach. New York : McGraw Hill: s.n. pp 375-377
24. RAMOS, D., NORIEGA, R., LAÍNEZ, J.R. y DURANGO, A., 2015. Curso de Ingeniería de Software. S.I.: s.n. pp 89-95
25. RUIZ, F. y LÓPEZ, P., 2014. Ingeniería ded Software I, Tema 2: Lenguaje Unificado de Modelado - UML. . S.I.: Universidad Cantabria – Facultad de Ciencias. pp 10-14
26. SÆTHER, B.S. y AULBACH, A., 2002. Manual de PHP. S.I.: s.n. pp 1-4

27. SÁNCHEZ, P.J.M., 2015. Pruebas de Software. Fundamentos y Técnicas. Madrid: s.n. pp 49
28. SCHWABER, K. y SUTHERLAND, J., 2013. La Guía Definitiva de Scrum: Las Reglas del Juego. S.I.: s.n. pp 3-6
29. SOMMERVILLE, I., 2011. Ingeniería de Software. 9. México: Pearson Educación: s.n. ISBN ISBN 978-607-32- 0603-7. pp 83-85, 155-157
30. SOMMERVILLE, I., 2016. Software Engineering, 10th Edition. University of St Andrews, Scotland: s.n. pp 206-212
31. SONE, Y.E.S., 2015. Sistema de Información de Logística para la Gestión de Insumos y Productos en una empresa del Rubro de Panadería y Pastelería. Perú: Universidad Católica de Perú. pp 2-20
32. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2018. About. [en línea], Disponible en: <https://www.postgresql.org/about/>.
33. TOLEDO, R.F., 2014. *Introducción a las Pruebas de Sistemas de Información*. Uruguay: s.n.
34. TRIGAS, G., Manuel, 2015. Metodología Scrum. . S.I.: pp 32-35
35. VAILLANT, V.C. de la C. y RUIZ, R.B., 2015. Módulo de Recomendaciones para el Sistema REPXOS. S.I.: Universidad de las Ciencias Informáticas. pp 25, 32-36

ANEXOS

Anexo 1: Historia de Usuario correspondiente a la funcionalidad Registrar menú.

Anexo 1 Historia de Usuario RF Registrar menú.

Número	55	Nombre del requisito: Registrar menú.
Programador: Leidy Márquez Barbosa		Iteración Asignada: 1
Prioridad: Alta		Tiempo Estimado: 1
Riesgo en Desarrollo:		Tiempo Real: 2
<p>Descripción: El usuario podrá insertar un menú seleccionando en el área de iconos flotantes la opción Registrar. Para ello debe llenar los siguientes campos:</p> <p>Fecha: ✓ Campo requerido.</p> <p>Evento: ✓ Campo requerido.</p> <p>Platos: ✓ Campo requerido.</p>		
<p>Observaciones:</p> <ul style="list-style-type: none"> ✓ El usuario debe estar autenticado en el sistema. ✓ Si no existen menús creados el listado se mostrará vacío. ✓ Si el usuario introduce la información correcta el sistema muestra el mensaje “La acción ha sido realizada satisfactoriamente.” y adiciona el nuevo menú a la Lista de menús. ✓ En caso que el valor de un campo único exista se muestra un mensaje de error: “No fue posible ejecutar la operación porque el elemento ya existe. 		

- ✓ En caso que se deje un campo de los obligatorios vacíos se muestra un mensaje en rojo “Campo requerido” debajo del campo que debe ser llenado obligatoriamente.
- ✓ Si el usuario desea cancelar la operación el sistema muestra el mensaje de advertencia “¿Está seguro que desea cancelar la operación? Si el usuario selecciona la opción “Si” el sistema regresa a la interfaz que lista los menús creados. Si el usuario selecciona la opción “No” el sistema se mantiene en la vista actual.

Prototipo elemental de interfaz gráfica de usuario:

Registrar menú

Fecha * Activo

Eventos * Platos * Visible

Lista de productos Registro por página 5

Platos	Precio	Gramaje	Unidad de medida	Clasificación	Visible	Sujeto a cambio	Opciones
- 08/02/2019							
- Comida							
Frijoles negros	1	75	Gramos	Granos	Si	Si	
- 14/02/2019							
- Almuerzo							
Arroz	1	150	Gramos	Granos	Si	no	
- 21/02/2019							
- Comida							
Frijoles negros	1	125	Kilogramo	Granos	Si	Si	

Mostrando 1 - 3 de 3

Anexo 2: Caso de prueba correspondiente a la funcionalidad Mostrar menú.

Anexo 2 Caso de prueba RF Mostrar menú.

Escenario	Descripción	Buscar	Registro por página	Número de página	Respuesta del sistema	Flujo central

EC 1.1 Mostrar datos correctamente.	Mediante este escenario se muestra el listado de menú existentes en el sistema. En el listado se muestra un conjunto de iconos internos con las acciones a desarrollar sobre cada elemento: Ver detalles de menú, Activar y Desactivar menú. Además se muestra un área de iconos flotantes con la acción de registrar.	NA	NA	NA	El sistema muestra el listado de elementos actualizados hasta la fecha.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el Módulo de Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El usuario selecciona o no los elementos por los que desea filtrar la búsqueda. -El sistema muestra los menú registrados.
		V	NA	NA		
EC 1.2 Buscar.	Mediante este escenario se muestra en el listado los elementos que coincidan con los criterios de búsqueda especificados.	V	NA	NA	El sistema muestra el listado de los elementos que coincidan con el nombre especificado.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo de Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El sistema muestra los menú registrados.
		V	NA	NA		

		vacío			El sistema muestra un listado de todos los elementos existentes.	-El usuario introduce los datos por los que desea filtrar la búsqueda y presiona el botón Buscar. -El sistema muestra los menú que se corresponden.
		I	NA	NA	El sistema muestra el listado vacío porque no encontró coincidencias.	
		\$253nhg				
EC 1.3 Seleccionar correctamente la cantidad de elementos por página.	Mediante este escenario el usuario escoge la cantidad de elementos a mostrar por página (los valores a escoger son 5, 10, 20, 40, 500 y 1000).	NA	V	NA	El sistema muestra la cantidad de elementos según la opción escogida (Se puede escoger 5, 10, 20, 40, 500 y 1000)	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El sistema muestra los menú registrados. -El usuario selecciona la cantidad de elementos que desea mostrar.
			El usuario escoge la opción de 5			
		NA	V	NA		
			El usuario escoge la opción de 10			
		NA	V	NA		
			El usuario escoge la opción de 20			
		NA	V	NA		
	El usuario escoge la opción de 40					
		NA	NA	V		

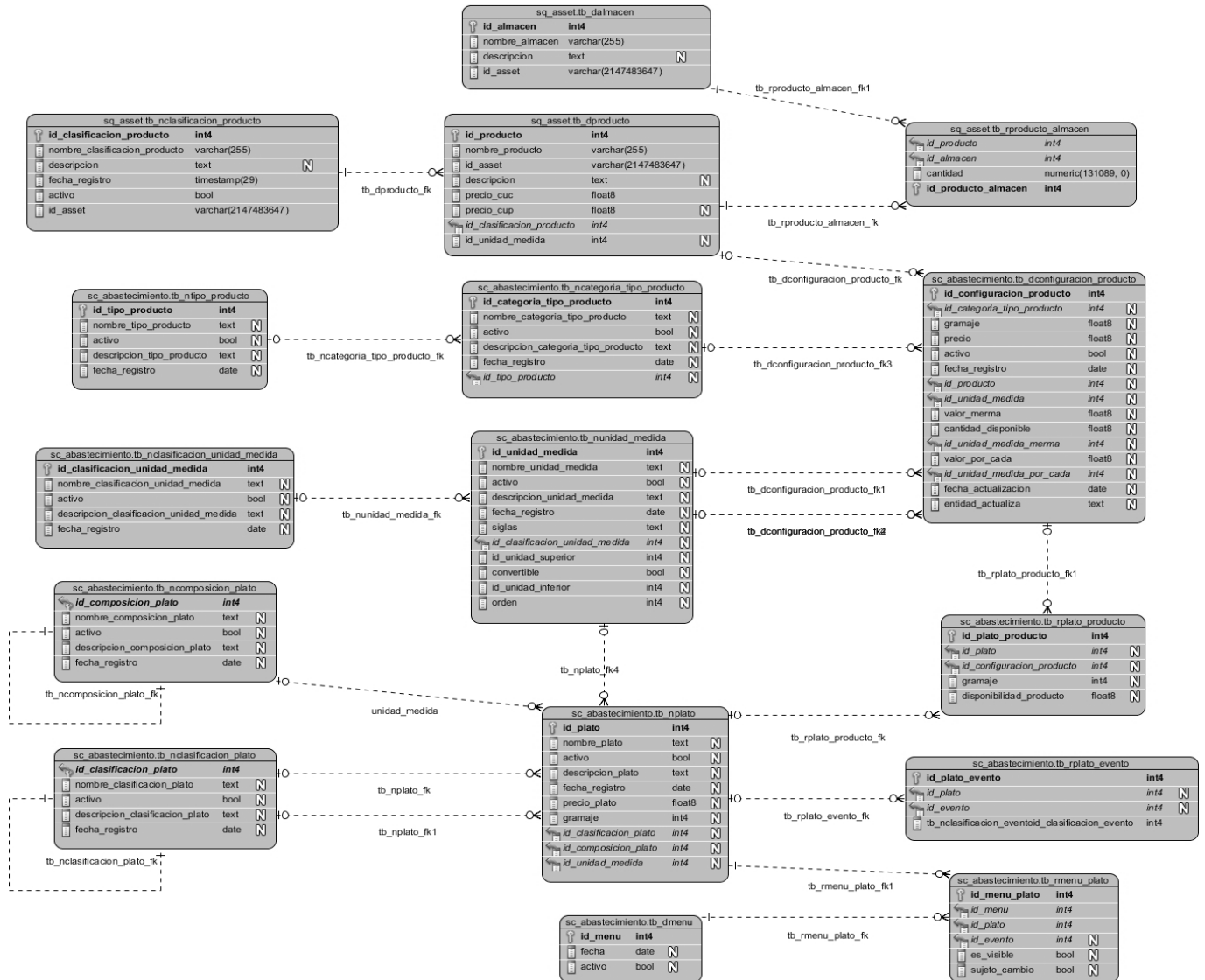
EC 1.4 Escribir correctamente el número de página.	Mediante este escenario el usuario escribe debajo del listado el número de la página a la que desea acceder.			2	El sistema muestra la página solicitada por el usuario.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo Abastecimiento -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El sistema muestra los registrados. -El usuario escribe correctamente el número de la página a la que desea acceder y presiona Enter en el teclado.
EC 1.5 Escribir 0 en el número de página.	Mediante este escenario el usuario escribe debajo del listado el número de la página a la que desea acceder.	NA	NA	V	El sistema muestra la página en la que se encontraba el usuario.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El sistema muestra los menú registrados. -El usuario escribe correctamente el número de la página a la que desea acceder y presiona Enter en el teclado.
		NA	NA	V		

EC 1.6 Dejar vacío el número de página.	Mediante este escenario el usuario deja vacío el número de la página debajo del listado.			Vacío	El sistema muestra la primera página.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El sistema muestra los menú registrados. -El usuario deja vacío el número de la página y presiona Enter en el teclado.
EC 1.7 Escribir número de página mayor que la cantidad existente.	Mediante este escenario el usuario escribe debajo del listado un número de la página a la que desea acceder, que es mayor que el total de páginas que posee el listado.	NA	NA	I 1150	El sistema muestra la última página del listado.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El sistema muestra los menú registrados. -El usuario escribe el número de la página a la que desea mayor que la cantidad de páginas que contienen los elementos registrados en el sistema y presiona Enter en el teclado.
EC 1.8 Escribir en el campo número de página un carácter diferente a	Mediante este escenario el usuario escribe en el campo de texto debajo del listado	NA	NA	I Letras y caracteres extraños(/*-+ etc)	El sistema muestra la primera página.	El usuario una vez autenticado en el Sistema Integrado de Alimentación selecciona el módulo Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la

un valor numérico.	algún carácter que sea diferente a un valor numérico (carácter extraño, letras, etc).					agrupación funcional "Menú". -El sistema muestra los menú registrados. -El usuario inserta en el número de la página caracteres diferentes a un valor numérico y presiona Enter en el teclado.
EC 1.9 No existen elementos creados.	Mediante este escenario en caso de que no exista creado ningún elemento se muestra un listado vacío.	NA	NA	NA	El sistema muestra el listado vacío.	El usuario una vez autenticado en el Sistema de Integrado de Alimentación selecciona el módulo Abastecimiento. -El sistema muestra las opciones de menú. -El usuario selecciona la agrupación funcional "Menú". -El sistema muestra el listado vacío.

El resto de Historias de Usuarios y Casos de Pruebas estarán contenidas en el Expediente de Proyecto.

Anexo 3: Modelo de datos del Módulo Abastecimiento.



Anexo 3 Modelo de datos del Módulo Abastecimiento.

GLOSARIO DE TÉRMINOS

Abastecimiento: entendido como aprovisionamiento, el abastecimiento es una función logística o de apoyo al trabajo interno de una institución, con los elementos materiales que estas necesitan para funcionar en las mejores condiciones de calidad y productividad. También se entiende como cosa de la que se abastece a una persona o población para cubrir ciertas necesidades. "alimentos, medicinas, materiales y abastecimientos esenciales para las necesidades civiles".

Almacén: es un área física seleccionada bajo criterios y técnicas adecuadas, destinada a la custodia y conservación de los bienes que van a emplearse para la producción de servicios o de bienes económicos. También se entiende como local, edificio o parte de este que sirve para depositar o guardar gran cantidad de artículos, productos o mercancías para su posterior venta, uso o distribución. Es considerado además como un establecimiento de grandes dimensiones y dividido en secciones en el que se venden todo tipo de productos.

Merma: es una pérdida o reducción de un cierto número de mercancías o de la actualización de un inventario que provoca una fluctuación, es decir, la diferencia entre el contenido de los libros de inventario y la cantidad real de productos o mercancía dentro de un establecimiento, negocio o empresa.

Menú: conjunto de platos que componen una comida. Además, es considerado como la comida que ofrece un restaurante por un precio fijo, con posibilidad limitada de elección de platos.

Producto alimenticio: es todo alimento que ha cambiado sus caracteres físicos, composición química y características Físico-Químicas, como consecuencia de la manipulación industrial. Ejemplos: quesos, manteca, yogur, pan, etc.

Unidad de medida: es una cantidad estandarizada de una determinada magnitud física, definida y adoptada por convención o por ley. Cualquier valor de una cantidad física puede expresarse como un múltiplo de la unidad de medida.

Planificación: la planificación es un método que permite ejecutar planes de forma directa, los cuales serán realizados y supervisados en función del planeamiento.

Integración de sistemas informáticos: tiene como objetivo conectar aplicaciones con la intención de compartir información entre ellas. Persigue una manera eficiente y flexible de combinar recursos y reutilizar

soluciones para perfeccionar su funcionamiento. Brinda una solución a problemas comunes en el desarrollo del *software*, cuando es necesario conectar sistemas aislados y posibilitar su funcionamiento como un único sistema.