Universidad de las Ciencias Informáticas

Facultad 1

Componente para la gestión del servicio del gas licuado en la Universidad de las Ciencias Informáticas

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

Autores: Yadriel Nápoles Rosales

Maykel Ramírez Reyes

Tutores: Ing. Norges Sánchez Tumbarell

Ing. Odaimys Mercedes Hechavarria Vargas

Ing. Walter Daniel Camejo López

La Habana, 2016
"Año 58 de la Revolución"

"El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia"

Síntesis de los tutores:

Ing. Odaimys Mercedes Hechavarria Vargas: Graduada de Ingeniero en Ciencias Informáticas en el curso 2011-2012. Desempeñó el rol de Analista de componente en los sistemas: Ubicación Laboral y Gestión Académica de Pregrado, en el curso 2013-2014 desempeñó el rol de Asesora de calidad en la Dirección de Informatización y actualmente se desempeña como Líder del proyecto “Sistema de Gestión Académica de Pregrado”. Ha sido tutora de 4 trabajos de diploma.

Ing. Norges Sánchez Tumbarell: Graduado de Ingeniero en Ciencias Informáticas en el curso 2006 - 2007. Desempeñó el rol de Jefe de Proyecto, Desarrollador y Analista en el proyecto del Sistema de Gestión de Pregrado. Actualmente es Especialista del Departamento de Tecnología y profesor de capacitación de los desarrolladores en las arquitecturas que se utilizan en la Dirección de Informatización.

Ing. Walter Daniel Camejo López: Recién graduado en adiestramiento (RGA) del centro CIDI. Integra el equipo de desarrollo del buscador ORION.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores del presente trabajo de diploma “Componente para la gestión del servicio del gas licuado en la Universidad de las Ciencias Informáticas” y se reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Maykel Ramirez Reyes — Autor

Yadriel Nápoles Rosales — Autor

_____ _____

Ing. Odaimys M. Hechavarria Vargas — Tutora

Ing. Norges Sánchez Tumbarell — Tutor

_____

Ing. Walter Daniel Camejo López

Tutor

Agradecimientos

A nuestras familias.
A nuestros amigos y amigas.
A nuestros profesores.
A nuestro tribunal y oponente.
A los especialistas del proyecto: Alejandro, Soya, Anisleidy, Laritza, Arianna, Orlando, Alfredo y Yoan Carlos, por su ayuda en todo momento.
A nuestros tutores: Odaimys Mercedes Hechavarría, Norges Sánchez y Walter D. Camejo, por su dedicación durante el desarrollo del Trabajo de Diploma.
A los profes: Geidis, Yari, Serguey, Alejandro, Osiris y Damián.
A nuestros padres por su apoyo incondicional.

Yadriel y Maykel.

Dedicatoria

Yadriel

A mis padres Juana y Benilde por haber depositado su confianza y darme su apoyo en mis decisiones.

A mi hermanita Yacemis que es lo que más quiero en la vida y a mi hermano Yasmanis por haberse comportado como un padre más para mí.

Maykel

A mis padres Estela y Reinaldo por todo su apoyo incondicional y por siempre estar presentes en todo momento.

A mi novia Geidis por estar siempre a mi lado, por haber llegado a mi vida cuando más lo necesitaba.

A mi hermano Michel y mis dos preciosos sobrinitos Meilyn y Maikol.

Resumen

La Dirección de Informatización (DIN) se encuentra desarrollando el Sistema de Gestión del Ciudadano (SGC), el cual tiene como objetivo englobar todos los componentes de software que permiten un mejor control de la ejecución de los procesos de la Guardia Obrera-Estudiantil, la reservación de Transportación y el servicio del gas licuado. Asociado al servicio de gas licuado hay soluciones informáticas desarrolladas, no obstante, ninguna se encuentra implantada.

En el presente trabajo se desarrolló un componente que permite una mejor gestión de la información del proceso del servicio del gas licuado en la Universidad de las Ciencias Informáticas (UCI). Para el desarrollo de la propuesta de solución en función de lograr los objetivos planteados, se realizó un estudio de los sistemas informáticos y conceptos asociados con la gestión del gas licuado. También se definieron un conjunto de herramientas, metodología y lenguajes a utilizar, además de la realización del modelado de los procesos de negocio. Este componente le permite al encargado de brindar el servicio del gas la generación de reportes en correspondencia con lo consumido en la UCI, así como llevar un control de la entrega y recogida de los cilindros por parte de los usuarios consumidores del servicio.

Palabras clave: componente, gas licuado, gestión, servicio.

Índice

Índice de tablas

Índice de figuras

Introducción

La importancia de las Tecnologías de la Información y las Comunicaciones (TIC) para el desarrollo social y económico ha sido reconocida desde el momento en que estas iniciaron un rápido crecimiento a mediados de la década de 1990 (Consejo Económico Social, 2014). En la actualidad, las TIC han alcanzado un lugar privilegiado en muchos sectores de la sociedad, de modo que se hace imprescindible para el crecimiento de cualquier país. Es por ello que las TIC no se han convertido solamente en una herramienta de difusión, sino en un medio de control para las tareas llevadas a cabo por las organizaciones.

Muchas empresas están informatizando sus procesos con el objetivo de resolver los problemas relacionados a la administración de datos y el conocimiento. Por tanto, en el ámbito organizacional la gestión de la información comprende todas las actividades que se relacionan con la obtención de información sólida, viable, confiable y actualizada que determinará el proceso de toma de decisiones en una organización (Ballesteros, 2008).

En la actualidad, muchas organizaciones en el mundo brindan disímiles servicios informáticos que permiten realizar labores con mayor rapidez, posibilitando un notable ahorro de tiempo para agilizar sus procesos. Cuba no queda exenta de los países que brindan servicios informáticos haciendo uso de las TIC. A medida que pasa el tiempo, en la sociedad surgen nuevas necesidades, lo que trae consigo la informatización de nuevos procesos. Es por ello que el Comandante en Jefe Fidel Castro Ruz crea la Universidad de las Ciencias Informáticas, donde tanto profesores como estudiantes tienen la importante tarea de diseñar e implementar aplicaciones que sean de gran utilidad a diversas organizaciones y disímiles áreas de la misma, con el fin de mejorar e informatizar cada proceso que se realiza.

El desarrollo de software en la UCI contribuye tanto al avance económico y tecnológico del país como a la informatización de los procesos universitarios. A partir de las diversas características que presenta la UCI, se brindan un conjunto de servicios a la comunidad universitaria entre ellos se destaca: salud, comercio, gastronomía y gas licuado. Centrando la atención en el último de los servicios mencionados anteriormente, la Dirección de Servicios Generales es la encargada de hacer la gestión del servicio del gas licuado en la Universidad, lo que le atribuye gran importancia ya que satisface la necesidad del consumo de gas de un notable número de usuarios residenciales.

Asociado a este servicio se han desarrollado en la UCI dos sistemas “Servicio de gas” y “Solicitud de gas” que permiten gestionar el proceso del servicio del gas, no obstante, ninguno de los sistemas se encuentra en funcionamiento.

Por lo tanto, en la actualidad el proceso de gestión y planificación relacionado con el servicio del gas licuado se realiza de forma manual, lo que lo hace sumamente engorroso teniendo en cuenta las diversas áreas que reciben este servicio como el complejo residencial Novia del Mediodía, comedores, policlínico, protocolo, rectorado y un importante número de apartamentos. También el volumen de información que se genera diariamente, dificulta el control de la misma debido a que esta se refleja mediante tarjetas de estiba, fichas de clientes, registro de edificio y el control de despacho, dando margen a que pueda existir pérdida o duplicidad de la información y poco control sobre los recursos. Otro inconveniente es que los usuarios que reciben este servicio tienen que llamar por teléfono y en el peor de los casos trasladarse hasta el punto del gas para ver si hay disponibilidad, esto trae consigo la aglomeración de personal frente al área de abastecimiento pudiendo entorpecer el trabajo del auxiliar técnico. Otro elemento a señalar es que al encargado de brindar el servicio se le dificulta realizar reportes semanales y mensuales de las personas que consumieron el servicio, lo que provoca no poder determinar cuántos usuarios han consumido el gas. Existe además dificultad en generar el informe de finanzas para saber la cantidad de dinero que se ha recolectado con la venta de los cilindros a clientes residenciales, lo que pudiera conllevar a posibles equivocaciones en el conteo de los recursos monetarios adquiridos en el proceso. También se hace engorroso para la persona encargada, emitir el reporte con la solicitud del gas a Cubapetróleo (CUPET), debido a la forma de realización de dicho proceso, lo que contribuye a que en un momento determinado se realice un pedido desacertado.

Por los elementos anteriormente expuestos se plantea como **problema de investigación:** ¿Cómo informatizar los procesos del servicio del gas licuado en la Universidad de las Ciencias Informáticas integrado al Sistema de Gestión del Ciudadano?

El **objeto de estudio** está constituido por el proceso de gestión del servicio del gas licuado, delimitándose como **campo de acción** el proceso de gestión del servicio del gas licuado en la UCI.

Para dar respuesta al problema antes mencionado se establece como **objetivo general:** desarrollar un componente que informatice los procesos del servicio del gas licuado en la Universidad de las Ciencias Informáticas integrado al Sistema de Gestión del Ciudadano.

Por el objetivo previamente planteado se definen como **objetivos específicos** los siguientes:

1. Realizar una investigación enfocada a los sistemas de gestión de gas licuado.
2. Describir la propuesta del componente de gestión del servicio del gas licuado en la UCI.
3. Implementar el componente de gestión del servicio del gas licuado en la UCI.
4. Probar el componente de gestión del servicio del gas licuado en la UCI.

La investigación se sustenta en la siguiente **idea a defender:** El desarrollo de un componente que informatice los procesos del servicio del gas licuado en la Universidad de las Ciencias Informáticas integrado al Sistema de Gestión del Ciudadano, contribuirá a un mejor control de la información de dichos procesos.

Luego de haber formulado los objetivos específicos se definen las siguientes **tareas de investigación**.

- ✓ Valoración de sistemas informáticos relacionados con el servicio de gas licuado en el ámbito internacional y nacional.
- ✓ Caracterización de las tecnologías y herramientas necesarias para el desarrollo del componente de software.
- ✓ Caracterización de la metodología de desarrollo de software.
- ✓ Definición de los requisitos funcionales y no funcionales del sistema.
- ✓ Diseño de las funcionalidades del sistema informático.
- ✓ Diseño del modelo de datos.
- ✓ Implementación de la propuesta de solución.
- ✓ Elaboración de los casos de prueba.
- ✓ Aplicación de casos de prueba elaborados.
- ✓ Documentación de las pruebas de carga y estrés, funcionales y de integración.

A continuación, se explican los **métodos científicos** que se utilizan en la investigación.

Métodos teóricos

Histórico – Lógico: se utilizó para satisfacer la necesidad de realizar un estudio acerca de las soluciones informáticas relacionadas con el objeto de estudio, así como las tecnologías y estándares utilizados para su desarrollo.
Modelación: se utilizó con el fin de modelar los procesos de negocios, así como crear los diagramas ingenieriles de la propuesta de solución.
Analítico – Sintético: empleado para el análisis de los elementos esenciales referentes a las teorías y documentos relacionados con el objeto de estudio. La utilización de este método permitió el estudio de sistemas que poseen funcionalidades relacionadas con el proceso de gestión del servicio del gas licuado.

Métodos empíricos

Entrevista: se utilizó para obtener información de cómo se lleva a cabo el proceso de gestión del servicio del gas licuado en la UCI. Además de conocer el estado de opinión del encargado de brindar dicho servicio, pudiendo determinar cuáles son las principales deficiencias del proceso.
Observación: el método permite un mayor acercamiento y familiarización con la organización a la cual se le desea informatizar los procesos que realiza. Se utilizó para observar cómo se desarrolla el proceso de gestión del servicio del gas licuado en la UCI.

Estructura del documento

Capítulo 1. Fundamentación teórica asociada al dominio del problema: se exponen un conjunto de conceptos y criterios fundamentales asociados al objeto de estudio de la investigación. Además, se estudian los principales sistemas relacionados con la gestión del servicio del gas licuado, con la finalidad de brindar una solución a la problemática planteada. Finalmente, se caracterizan las distintas tecnologías a utilizar en el desarrollo del componente, así como la metodología de desarrollo de software a utilizar.

Capítulo 2. Análisis y diseño del Componente para la gestión del servicio de gas licuado: se describen los procesos actuales que incluye la gestión del servicio del gas en la Universidad.

Además, se exponen las características del componente, incluyendo los requisitos funcionales y no funcionales, patrones de diseño y arquitectura; además algunos de los artefactos que requiere la metodología de desarrollo utilizada.

Capítulo 3. Implementación y prueba del Componente para la gestión del servicio de gas licuado: contiene aspectos asociados con la implementación de la solución informática. Además, se presentan los diseños de casos de prueba utilizados en la validación del componente y se analizan los resultados de las pruebas realizadas que permiten evaluar la calidad de la propuesta de solución.

Capítulo 1: Fundamentación teórica asociada al dominio del problema

1.1 Introducción

En este capítulo se dan a conocer los elementos que son objeto de estudio en el proceso de gestión del servicio del gas licuado. También se evidencia un estudio del estado del arte referente a los sistemas informáticos existentes que responden a este proceso, así como el marco de trabajo, tecnologías y herramientas a utilizar para el desarrollo del sistema.

1.2 Definiciones asociadas

Con el objetivo de lograr un mayor entendimiento de la investigación, se mencionan algunos de los principales conceptos relacionados con el tema, además de optar por estas definiciones en la solución propuesta.

A continuación, se muestran definiciones de **sistema** por varios autores:

Componente: según Philippe Krutchen de Rational Rose, un componente es una parte no trivial, casi independiente y reemplazable de un sistema que cumple una función dentro del contexto de una arquitectura bien definida. Un componente cumple con un conjunto de interfaces y provee la realización física de ellas (Wallnau, 1998).

Gestión: a consideración de los autores de la investigación se asume que la gestión es el proceso en el cual se manejan una variedad de recursos esenciales con el fin de alcanzar los objetivos de la organización.

Sistema de gestión: es un conjunto de reglas y principios relacionados entre sí de forma ordenada, para contribuir a la gestión de procesos generales o específicos de una organización. Permite establecer una política, unos objetivos y alcanzar dichos objetivos. (Thinkandsell, 2013)

Servicio: Stanton, Etzel y Walker, definen los servicios "*como actividades identificables e intangibles que son el objeto principal de una transacción ideada para brindar a los clientes satisfacción de deseos o necesidades*" (Stanton, 2006). Dicho concepto será asumido en la presente investigación.

Servicio informático: los autores consideran que un servicio informático es un conjunto de actividades asociadas al manejo electrónico de la información para satisfacer las necesidades de los usuarios.

Sistema de reservación: es un medio para acceder a los servicios de una empresa o institución en general durante un tiempo determinado con antelación a que este sea brindado (Gertrudys, 2010).

1.3 Soluciones informáticas relacionadas con el objeto de estudio de la investigación

A continuación, se realiza un estudio de soluciones informáticas existentes que controlan la información que se genera en los procesos asociados a la gestión de gas licuado.

1.3.1 Soluciones existentes en el ámbito internacional

Pemex Gas y Petroquímica Básica

PGPB[1] es una empresa mexicana cuya misión es maximizar el valor de los activos petroleros y los hidrocarburos de la nación, satisfaciendo la demanda nacional de productos petrolíferos con la calidad requerida, de manera segura, confiable, rentable y sustentable con la visión de ser reconocida por los mexicanos como un organismo socialmente responsable (PGPB, 2015).

Pemex Gas y Petroquímica Básica cuenta con un sistema informático que le permite al usuario ver los productos y servicios que comercializa, entre ellos se encuentran: **Gas natural, Gas licuado, Petroquímicos básicos, Azufre, Transporte de hidrocarburos por ductos y Coberturas de gas natural.** Además, brinda información sobre los precios estableciendo el costo según el tipo de envase en que se brinde el servicio (10 kg, 20kg, 30 kg, 45 kg, 1 L) y el lugar al que se distribuye.

El sistema informático utilizado por esta subsidiaria tiene la ventaja de ser multiplataforma, lo que brinda la posibilidad de ser usado en cualquier sistema operativo, además de tener una interfaz amigable para los usuarios, permite que estos realicen pedidos en línea, así como realizar el pago de diferentes formas. No es recomendable la utilización de este sistema debido a que es propietario, razón que no permite modificaciones en el código para satisfacer las necesidades de otras organizaciones.

[1] **PGPB:** Pemex Gas y Petroquímica Básica

Precios de Cilindros de Gas Licuado de Petróleo en Línea

En Chile, la Comisión Nacional de Energía, en el marco de sus funciones y atribuciones legales, implementó el portal de “Precios de Cilindros de Gas Licuado de Petróleo en Línea”, el cual es parte del Sistema de Información de Precios de Distribución de Gas desarrollado por esta Comisión. (Gasenlinea, 2014)

En esta página web, el usuario puede acceder a las zonas geográficas de cobertura de distribuidores de cilindros de Gas Licuado de Petróleo (GLP) con el fin de encontrar un distribuidor que responda a sus necesidades. El objetivo fundamental del sistema es comercializar gas normal y catalítico en cilindros de tamaños diferentes: 2 kg, 5kg, 11 kg, 15kg y 45 kg.

El portal Precios de Cilindros de Gas Licuado de Petróleo en Línea tiene cobertura nacional, abarcando a aquellos distribuidores con capacidad de almacenamiento igual o superior a 5.000 kg lo que posibilita que siempre haya disponibilidad de gas. A pesar de ser un sistema con una interfaz intuitiva lo que posibilita que los usuarios naveguen con facilidad, presenta problemas de seguridad ya que cualquier usuario luego de acceder a la vista principal del sitio puede solicitar un distribuidor sin necesidad de recibir el servicio, además de brindar la posibilidad de llenar un formulario en caso de alguna queja en cuanto al precio, formulario en el que los datos introducidos pueden ser incorrectos y este es un sistema propietario.

Abastible

Abastible es un sistema chileno cuyo objetivo es distribuir y comercializar gas licuado normal y catalítico en distintos tamaños de cilindros: (5, 11, 15 y 45 kg). El sistema brinda a sus clientes la posibilidad de hacer uso de un Servicio de Asistencia Técnica, el cual consiste en brindar ayuda para el caso de que alguna balita de gas tenga salidero o cualquier otro artefacto. También brinda la posibilidad de realizar el pago de varias formas: en línea y en efectivo. (Abastible, 2014)

El sistema Abastible cuenta con diferentes opciones para realizar un pedido. Los usuarios pueden elegir un distribuidor en dependencia de la zona geográfica en la que se encuentre, así como el tipo de cilindro que desea. Abastible cuenta con una seguridad de la información adecuada pero no se puede hacer uso de este debido a que es un sistema privativo.

Empresas Lipigas S.A

Empresas Lipigas se dedica a la comercialización de gas licuado de petróleo (GLP) en Chile, Perú y Colombia. Además, la compañía participa en el mercado de gas natural (GN) y a partir de 2014 comercializa gas natural licuado (GNL) como alternativa a sus clientes industriales (LarrainVial , 2015).

Lipigas es una sucursal chilena que cuenta con un sistema informático que permite realizar pedidos en línea de gas licuado en cilindros de 4 tipos: 5, 11, 15 y 45 kg. También brinda una ayuda técnica en caso de existir problemas con el cilindro de gas (Lipigas, 2014). A pesar de que es un sistema que permite a sus clientes realizar el pago en línea y tiene una interfaz amigable; este presenta problemas de seguridad, ya que permite llenar un formulario con una serie de información sin verificar previamente que se trata de un cliente registrado y es un sistema propietario lo que impide su uso en un área no establecida por Lipigas.

GASCO

GASCO comercializa gas licuado normal y gas licuado catalítico en formatos de 2, 5, 11, 15 y 45 kg para uso residencial, empresas e industrias. Todos están equipados con válvulas de alta seguridad y que por sus características permiten la conexión a cualquier artefacto diseñado para funcionar con Gas Licuado, junto al regulador de GLP. (Gasco, 2014)

Gasco es un sistema chileno que permite a sus usuarios realizar pedidos de gas y el pago en línea, así como localizar alguna sucursal en línea. También brinda la posibilidad que los clientes realicen consultas y reclamos a la empresa distribuidora de gas licuado que presta el servicio y a que los resuelvan en un plazo no mayor que 15 días, esto permite que cada vez se brinde un buen servicio logrando la satisfacción de los clientes.

Este sistema presenta una seguridad adecuada, ya que para realizar cualquier operación el usuario debe entrar una serie de información que muestre que realmente es un cliente. Sin embargo, este no puede ser usado debido a que es una solución privativa por lo que no puede ser modificado para ser usado por otros centros.

En general, los sistemas anteriormente estudiados brindan funcionalidades que responden a las necesidades de la Dirección de Servicios Generales y de la comunidad universitaria. A pesar de ello no existe ninguno que cumpla en su totalidad con las especificaciones del cliente debido a que son aplicaciones propietarias a las cuales no se les puede realizar modificaciones o adaptaciones. Estos sistemas tampoco están en concordancia con las políticas de migración a software libre que lleva a cabo la Universidad. Por otra parte, para dar soporte a los mismos existe total dependencia de la empresa propietaria del sistema.

1.3.2 Soluciones o propuestas realizadas para la gestión del servicio del gas licuado en la UCI

En el 2006 se desarrolló el sistema “Solicitud del servicio de gas”, este fue creado para gestionar todo el proceso del servicio del gas licuado en la Universidad y con ello permitir a la comunidad realizar la solicitud del servicio mediante la web. Dentro de las prestaciones del sistema se encontraba la funcionalidad “Gestionar usuario” la cual permite crear, modificar y eliminar los usuarios que tenían acceso al sistema, además de diferenciar mediante roles las funcionalidades que podía acceder determinado usuario (Salina, 2012). Posteriormente se desarrolló el “Sistema de gestión del Servicio del gas licuado” para mejorar el sistema que se encontraba en explotación, añadiendo las funcionalidades de permitir el registro de las lecturas a los locales que reciben gas a granel, brindar reportes y llevar el control económico del balance mensual y anual del ingreso registrado.

Utilizar una de estas soluciones no es lo más recomendable, ya que resulta muy costoso en cuanto a tiempo y recursos para adaptarlos a la nueva arquitectura que definió la Dirección de Informatización para unificar los procesos de gestión ciudadana. Como parte de la adaptación se incluiría rediseñar todas las vistas de estos sistemas, de forma tal que cumplan con las pautas de diseño definidas en el documento de proyecto “010216_3a Arquitectura de Software Vista de Presentación DAC” así como unificar un conjunto de funcionalidades que estas soluciones deben hacer sin la necesidad de la actividad humana. Actualmente ninguna de las dos se encuentra en funcionamiento, por lo que no se cuenta en la UCI con una solución informática que permita llevar un control de la información que se genera en los procesos asociados a la gestión de gas licuado.

Teniendo en cuenta lo anteriormente planteado, tanto las soluciones estudiadas a nivel internacional como los sistemas realizados en la UCI, no satisfacen las expectativas del cliente, lo que evidencia la necesidad de desarrollar un componente de software que permita llevar un control del proceso de gestión del gas en la Universidad que se integre además al Sistema de Gestión del Ciudadano.

Sin embargo, fue posible identificar un conjunto de funcionalidades a tener en cuenta en el desarrollo de la nueva propuesta de solución, entre las que se encuentran: gestionar cilindro, gestionar casos excepcionales, gestionar tipos de casos excepcionales, registro de consumo de gas y gestionar tipo de cliente, el registro de lecturas para los clientes institucionales y reportes de la información gestionada en el sistema.

1.4 Ambiente de desarrollo

A continuación, se describen las herramientas y tecnologías a utilizar en el desarrollo de la propuesta de solución, siguiendo la base tecnológica definida por la DIN para el desarrollo de los componentes a integrar al Sistema de Gestión del Ciudadano.

1.4.1 Lenguaje de programación

Del lado del servidor

PHP5

Se trata indudablemente del lenguaje script de servidor más popular. Hoy en día se pueden instalar módulos para interpretar PHP "*Hypertext Preprocessor*" en casi todos los servidores de aplicaciones web. En especial PHP tiene una gran relación con Apache. Es un lenguaje basado en C y en Perl, que se ha diseñado pensando en darle la máxima versatilidad y facilidad de aprendizaje, por encima de la rigidez y coherencia semántica. Dentro de sus ventajas se encuentra que es un lenguaje que se puede lanzar en casi todas las plataformas de trabajo, pertenece al software licenciado como GNU, la licencia del sistema Linux; lo que permite su distribución gratuita, por ser un lenguaje de código abierto, permite tener una comunidad amplia y muy dinámica a la que acudir en caso de necesidad y dispone de un enorme número de extensiones que permiten ampliar las capacidades del lenguaje, facilitando la creación de aplicaciones web complejas (Sánchez, 2012).

La sintaxis de PHP es similar a la del lenguaje C, debido a esta razón cualquier usuario con experiencia en C podrá entender rápidamente PHP. También puede interactuar con muchos gestores de bases de datos tales como: MySQL, MS SQL, Oracle, Informix, PostgreSQL y otros. Teniendo en cuenta las ventajas que proporciona PHP, se decide su utilización en su versión 5.5.9.

Del lado del cliente

HTML4, CSS3 y JavaScript

HTML[2] es un lenguaje extensible al que se le pueden añadir nuevas características, marcas y funciones. Los documentos HTML están formados por una serie de bloques de texto con una entidad lógica. La interpretación de estas entidades se deja al navegador, lo cual da una gran flexibilidad a la presentación del documento que puede ser mostrado, este es el lenguaje que permite diseñar los hipertextos (Lemay, 2012). Mientras tanto, CSS[3] es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML, este es quien define la estética de un sitio web en un documento externo y eso mismo permite que modificando dicho documento se pueda cambiar la estética entera de un sitio web, el cual puede variar totalmente de estética cambiando solo el CSS, sin tocar para nada los documentos HTML (Gauchat, 2012). De manera general, HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y JavaScript hace el resto que es extremadamente significativo (Gauchat, 2012).

1.4.2 Marco de trabajo para PHP

Puede decirse que un marco de trabajo (*framework)* es el esqueleto de una aplicación que debe ser adaptado por el programador para desarrollar una aplicación específica.

Existen diversos *frameworks* para desarrollo en PHP. De entre ellos se seleccionaron los siguientes (cuatro de los más usados), con el propósito de medir sus características y determinar qué tan bien aplican los conceptos de la Programación Orientada a Objeto (POO) (Acosta, 2012). En la tabla 1 se muestran varios framework estudiados.

[2] **HTML:** lenguaje de marcas de hipertexto (*HyperText Markup Language*).
[3] **CSS:** Hojas de Estilo en Cascada

Tabla 1. Marco de trabajo para PHP.

PHP Framework	Arquitectura MVC	Validación incorporada	Soporte mapeador de objetos	Soporte múltiples base de datos	Almacena-miento en caché de objeto	Comunidad de código abierto	Gran cantidad de documentación
CodeIgniter	Si	Si	-	Si	Si	Si	Si
Symfony	Si	Si	Si	Si	Si	Si	Si
Yii	Si	Si	Si	Si	Si	Si	Si
Zend	Si	Si	Si	Si	Si	Si	Si

En la tabla 1 se pueden apreciar las facilidades que proporciona usar alguno de estos *frameworks*. Sin embargo, se decide utilizar Symfony en su versión 2.8 debido a que posee un reducido número de requisitos previos, lo cual hace que sea muy fácil de instalar en cualquier configuración (*Linux o Windows*). Es compatible con diversos sistemas de base de datos. Tiene una baja curva de aprendizaje y permite construir aplicaciones robustas en un contexto empresarial (Acosta, 2012). Este es un marco de trabajo que está basado en el patrón Modelo-Vista-Controlador, aunque como muestra de su flexibilidad, no obliga a los desarrolladores hacer uso del mismo. Además, cuenta con sistema de comandos que permite simplificar el trabajo. Permite hacer uso de componentes de terceras partes lo cual posibilita facilitar mucho las tareas, este presenta un buen rendimiento, está muy bien documentado y cuenta con una comunidad de desarrolladores, elemento que garantiza el soporte de este marco de trabajo.

1.4.3 Metodología de desarrollo

Para lograr una mayor organización en el proceso de desarrollo de software se seleccionan las metodologías de desarrollo, las cuales constituyen una filosofía de trabajo que proporciona una base de procesos para llevar a cabo con éxito cualquier proyecto informático. Las metodologías de desarrollo brindan soporte a la toma de decisiones en un equipo de trabajo. (Condori, 2010)

La metodología Desarrollo Ágil con Calidad (DAC), es la que se emplea en la DIN para el desarrollo de software. DAC está diseñada para el desarrollo de proyectos con un enfoque ágil mediante un proceso colaborativo, recursivo-iterativo, incremental y guiado por procesos y requisitos. Este

proceso tiene 8 actividades del marco de trabajo del proceso común llamadas fases o procesos del ciclo de vida: inicio, análisis y diseño arquitectónico, requisitos, construcción, cierre de iteración, liberación, transición y cierre, ocurriendo las iteraciones concurrentes entre las fases de requisitos, construcción y cierre de iteración. Las entregas en DAC son a nivel de iteración, en la que habrá obligatoriamente un incremento del producto a partir de la solución de un componente del mismo. Las iteraciones no tienen que desarrollarse todas al mismo tiempo, sino que, al contar con un equipo pequeño, este se va a ir moviendo de una iteración a otra a medida que estas vayan terminando de acuerdo a un orden de prioridad establecido en el plan del proyecto (Sánchez, 2013).

1.4.4 Visual Paradigm para UML 8.0

Visual Paradigm es una herramienta CASE (*Ingeniería de Software Asistida por Computador*) que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

Permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta, controla que el modelado con UML sea correcto, al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad, permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo, permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software y posibilita generar diversos informes a partir de la información introducida en la herramienta (Ruiz, 2013). Se propone la utilización de esta herramienta en su versión 8.0.

1.4.5 Entorno de desarrollo integrado

El uso de los entornos de desarrollo integrados se ratifica y afianza en los 90 y hoy en día se cuenta con infinidad de Entornos de Desarrollo Integrados (*IDE*), tanto de licencia libre como privada (Comesaña, 2012). A continuación, se muestra en la tabla 2 detalles de algunos IDE.

Tabla 2. IDE más relevantes.

Entorno de desarrollo	Lenguajes que soporta	Tipo de licencia
NetBeans	C/C++, Java, JavaScript, PHP, Python	De uso público
Eclipse	Ada, C/C++, Java, JavaScript, PHP	De uso público
Microsoft Visual Studio	Basic, C/C++, C#	Propietario
C++ Builder	C/C++	Propietario
JBuilder	Java	Propietario

Se decide utilizar como IDE de programación el NetBeans en su versión 8.0. Este presenta características fundamentales como: es gratuito y de código abierto para el desarrollo de aplicaciones web de escritorio y para móviles. La base en la que se sustenta su elección es que permite desarrollar sistemas informáticos y ejecutar los comandos del mismo directamente desde la interfaz del IDE. NetBeans es de fácil instalación y compatible con el lenguaje de programación seleccionado, además de ser un IDE multiplataforma.

1.4.6 Servidor web (Apache)

Para atender las peticiones de los usuarios mediante una interfaz web se utiliza como servidor web HTTP Apache. Apache 2 es un servidor web de código abierto que implementa el protocolo HTTP 1.1, caracterizado fundamentalmente por su alto nivel de configuración, modularidad, robustez y estabilidad. Desarrollado bajo la licencia ASF por *The Apache Software Foundation*, es considerado una de las mejores y más aceptadas creaciones del mundo del software libre. Teniendo en cuenta las estadísticas históricas y uso diario proporcionadas por NetCraft, este servidor llegó a usarse durante el 2005 en el 70% de los sitios web en el mundo, representando su cuota máxima en el mercado hasta la actualidad. Las estadísticas presentadas demuestran que Apache 2 ha sido considerado como el servidor web HTTP por excelencia desde Julio de 1998, según las estadísticas

(NetCraft, 2015), logrando que millones de servidores mundiales ratifiquen su utilización. En el desarrollo de la investigación se emplea este en su versión 2.4.7.

1.4.7 JMeter

JMeter es un proyecto de Apache que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. JMeter puede ser usado como una herramienta de pruebas unitarias para conexiones de bases de datos, FTP, LDAP y Servicios web (Foundation-Tika, 2014). Se hará uso de esta en su versión 2.12 para comprobar el rendimiento de la propuesta de solución.

1.4.8 Gestor de base de datos

Un Sistema Gestor de Base de Datos (SGBD) es un software de propósito general que facilita los procesos de definición, construcción y manipulación de la base de datos para distintas aplicaciones (Garzón, 2010)**.** Estos sistemas facilitan el control sobre los datos almacenados y posibilitan garantizar la integridad y confiabilidad de la información a través de las políticas de seguridad que implementan.

Actualmente existen muchos SGBD, algunos más robustos y confiables que otros, pero todos inclinados a la manipulación sencilla de la información residente en una base de datos. A continuación, se realiza un estudio del gestor de base de datos PostgreSQL.

PostgreSQL

Se trata de un sistema de administración de base de datos que incorpora el modelo relacional para sus bases de datos y es compatible con el lenguaje de consulta SQL estándar. Resulta ser muy capaz y confiable, y posee buenas características de rendimiento. Es un sistema multiplataforma por lo que opera en varios sistemas operativos como Unix, Mac OS, Windows, Linux, es de código abierto, lo que hace posible que los usuarios puedan realizar las modificaciones pertinentes al código fuente según sus necesidades (Neil, 2005).

Otras de las ventajas de Postgres es la rapidez de respuesta para múltiples usuarios conectados concurrentemente. Existen sistemas PostgreSQL en ambientes de producción que manejan más de

4 terabytes de datos. Este incluye la mayoría de los tipos de datos de SQL: 2008, como INTEGER, NUMERIC, BOOLEAN, CHAR, VAR-CHAR, DATE, INTERVAL y TIMESTAMP (López, 2014).

Este gestor es robusto y por ende hoy en día es muy usado con respecto a gestores libres existentes como SQLite, MySQL, FireBird, entre otros. Dadas las características mencionadas previamente se aprecian las ventajas que este gestor brinda. Se ha decidido utilizar PostgreSQL, pues provee a la base de datos consistencia, además proporciona buen rendimiento bajo grandes cargas de trabajo. Se propone la utilización de este en su versión 9.4.

1.4.9 Evolus Pencil

Se trata de una herramienta gratuita y de código abierto para diseñar prototipos de interfaz. Es de gran ayuda para diseñadores y desarrolladores web, pues permite diseñar rápida y fácilmente documentos de propuesta para clientes. Con la utilización de la misma se puede obtener un modelo del proyecto antes de ponerse a construirlo.

Características fundamentales (MeriFont, 2012)**:**

- ✓ Diseño de plantillas y creación de prototipos.
- ✓ Vínculos y enlaces internos entre tus páginas.
- ✓ Edición de texto.
- ✓ Instalación de plantillas definidas por el usuario.
- ✓ Operaciones estándar de dibujo: alineación, rotación, dimensiones, etc.
- ✓ La exportación a *HTML*, PNG, documento de *OpenOffice*, *Word* y PDF.

Se hará uso de esta herramienta para el diseño y construcción de los prototipos de interfaz en su versión 2.0.5.

1.5 Conclusiones del capítulo

En este capítulo se trataron los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones:

- ✓ El estudio de los conceptos asociados a la problemática planteada permitió lograr un mejor entendimiento de la investigación que se realiza.
- ✓ El análisis de las funcionalidades que brindan algunos de los sistemas que permiten la gestión del gas, permitió identificar requerimientos a tener en cuenta en la implementación.
- ✓ La selección de las tecnologías, metodología y herramientas con soporte multiplataforma y basadas en software libre, posibilitó obtener una base tecnológica enfocada en las políticas de desarrollo definidas en la Universidad.
- ✓ Se hace necesario el desarrollo de un componente para llevar un control de la información referente al servicio del gas.

Capítulo 2: Análisis y diseño del Componente para la gestión del servicio del gas licuado

2.1 Introducción

En el presente capítulo se abordan los aspectos fundamentales relacionados con el análisis y diseño del componente a desarrollar, así como la descripción de la propuesta de solución. Como parte del diseño de la aplicación se caracteriza la arquitectura y patrones de diseño empleados en el marco de trabajo. Además, se exponen las principales características de la solución informática, se describen los procesos que se realizan en la gestión del servicio del gas licuado en la UCI, identificando las principales actividades que se informatizarán. También se describen las técnicas utilizadas para la obtención de los requisitos de software y se describen los mismos.

2.2 Modelado de procesos del negocio

Un proceso de negocio es una colección de actividades diseñadas para producir una salida específica para un cliente o un mercado en particular. Esto implica un fuerte énfasis en cómo se realiza el trabajo dentro de una organización (Sparks, 2014). El modelado de los procesos constituye en detallar haciendo uso de herramientas de manera que se pueda visualizar el orden en que ocurren las tareas dentro de una determinada organización.

2.2.1 Flujo actual de los procesos comprendidos en la gestión del servicio del gas licuado en la UCI

La Dirección de Servicios Generales es la encargada de gestionar el servicio del gas licuado en la Universidad de las Ciencias Informáticas. Esta gestión incluye cuatro procesos fundamentales los cuales se describen y modelan a continuación.

Proceso Realizar reservación

El proceso inicia cuando el cliente residencial se presenta en el punto o en la oficina de la Dirección de Servicios Generales para realizar la renovación del servicio del gas licuado. Para esto el cliente muestra su identificación (solapín) al auxiliar técnico. Este verifica la identificación y si el cliente recibe gas o no, en la ficha de cliente. De tener los datos anteriores correctos se verifica si el cliente

residencial está en tiempo de realizar la reservación. En caso de ser correcto el tiempo de realizar la reservación, se verifica que haya entregado el cilindro vacío y si entregó el cilindro vacío se registran los datos de la reservación, el cliente se retira. De no cumplir con las verificaciones se le notifica al cliente que no puede realizar la reservación (Ver Figura 1).

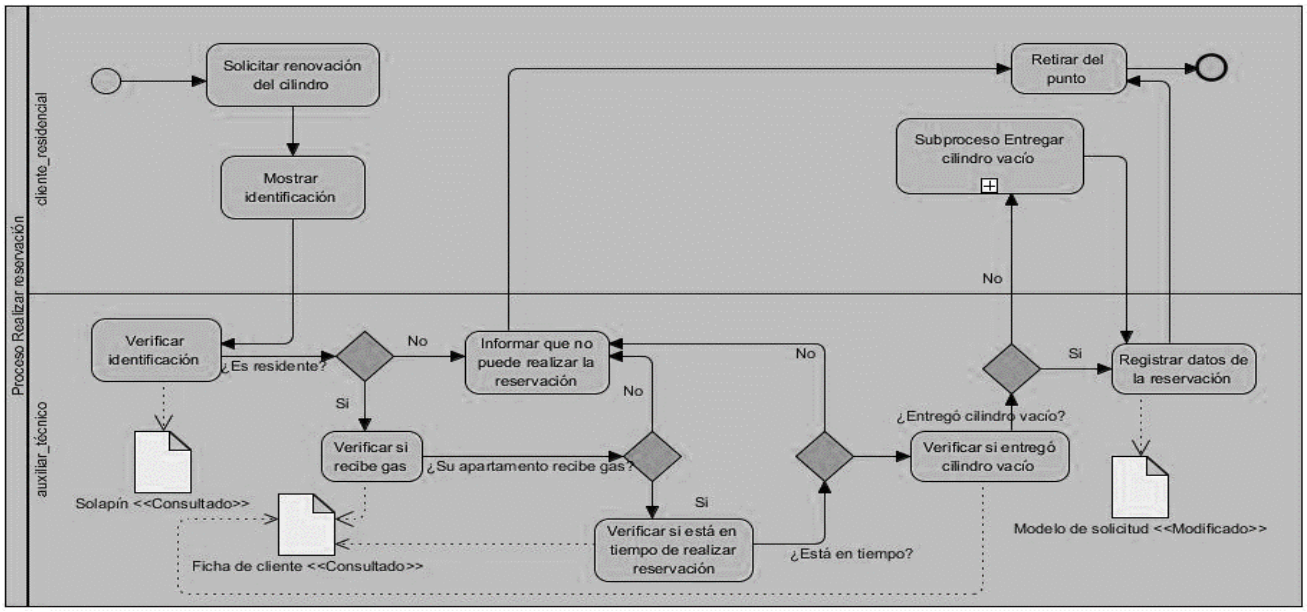


Figura 1. Proceso Realizar reservación.

Para consultar el modelado del resto de los procesos de negocio ver en el **Anexo 1, 2 y 3**.

Proceso Recogida de cilindro lleno

El proceso inicia cuando el auxiliar técnico notifica (correo electrónico o teléfono) a los clientes residenciales que pueden ir a recoger el cilindro de gas lleno. El cliente se presenta en el punto de despacho para la recogida del cilindro lleno, entregando su identificación (solapín). El auxiliar técnico verifica la identificación y si el cliente posee una reservación, mediante la ficha de cliente y del modelo de solicitud. Si al verificar los datos son correctos, el auxiliar técnico verifica si entregó el cilindro vacío. Si todo es correcto este registra la entrega del cilindro lleno mediante los documentos ficha de cliente y control de despacho. Posteriormente hace entrega del cilindro lleno, el cliente residencial lo recibe y se retira del punto. De lo contrario el auxiliar técnico le notifica al cliente residencial que no puede hacer entrega del cilindro lleno.

Proceso Abastecer local

El proceso se inicia cuando el cliente institucional se presenta en el punto para recoger el cilindro, mostrando el modelo de autorizo y el solapín. Si los datos son correctos el auxiliar técnico los registra y despacha el cilindro. El cliente recibe el cilindro lleno y se retira del punto de despacho. De existir algún dato incorrecto el auxiliar técnico le notifica que no puede entregar el cilindro lleno.

Subproceso Entregar cilindro vacío

El proceso inicia cuando el cliente se presenta en el punto para la entrega del cilindro vacío, el auxiliar técnico verifica que coincida el número de serie del cilindro que entrega el cliente con el registrado en la ficha de cliente. De coincidir, el auxiliar técnico recibe el cilindro, de no ser correcto, se analiza el hecho ocurrido y posteriormente se registran los datos.

2.2.2 Análisis crítico de la ejecución actual de los procesos

A pesar de que los procesos descritos previamente se desarrollan adecuadamente cumpliendo con los objetivos se realizan de forma manual, dando margen a la pérdida de tiempo en el intercambio de información, recogida de datos y en la búsqueda de los documentos necesarios para tomar una decisión acertada, trayendo como consecuencia que la persona que brinda el servicio de gas se agote innecesariamente.

2.2.3 Procesos objeto de la informatización

Se desean informatizar todas las actividades relacionadas con el manejo de la información referente a los procesos anteriormente descritos, de forma tal que los datos asociados a los consumidores del servicio puedan ser gestionados con mayor rapidez. Además, la persona que brinda dicho servicio podrá consultar, introducir y modificar datos de acuerdo con los permisos establecidos en el componente que se pretende desarrollar.

2.3 Reglas del negocio

- ✓ Las personas que pueden reservar son los trabajadores residentes en apartamentos que reciben el servicio del gas en la Universidad.
- ✓ La reservación del servicio, solo la pueden realizar los responsables para el tipo de cliente institucional, dígase clientes institucionales los locales que brindan servicios tercerizados o cuentan con el servicio de gas a granel.
- ✓ La recogida del cilindro lleno se debe hacer cuando se acabe realmente el cilindro que tienen y previamente se debe haber efectuado la reservación, de lo contrario no se realiza el cambio.
- ✓ El modelo de autorizo debe estar firmado por el responsable del local.
- ✓ Para la recogida del cilindro, el usuario debe presentar en el punto de abastecimiento su identificación (solapín).
- ✓ Para los apartamentos en que residan niños menores de dos años, enfermos, complejo residencial Novia del Mediodía y embarazadas, este ciclo de cambio será una vez al mes.
- ✓ Para el tipo de cliente residencial que no constituya uno de los casos mencionados anteriormente, el cambio será cada tres meses.
- ✓ El reaprovisionamiento para los pantrys y entidades tercerizadas que reciben el servicio del gas licuado, será como norma de un cilindro de 10 kg mensualmente, a excepciones de algunos casos previamente establecidos.

2.4 Gestión de requisitos

Los requisitos del software permiten establecer lo que el sistema debe hacer, sus características fundamentales, y las restricciones en el funcionamiento del sistema y los procesos de desarrollo del software. De manera general, estos requisitos expresan las necesidades objetivas que presentan los usuarios, ante un sistema que resuelve un problema en particular de un determinado dominio. (Sommerville, 2005)

Luego de haber modelado los procesos actuales del negocio, se presentan las técnicas para la captura de los requisitos, así como los requisitos funcionales y no funcionales de la solución informática a desarrollar.

2.4.1 Identificación de requisitos

Para la identificación de los requisitos existen varias técnicas que permiten establecer una correcta comunicación con los interesados y equipo de desarrollo (Pressman, 2001).
A continuación, se describen las técnicas utilizadas durante el proceso de desarrollo del software para recopilar los requisitos de la propuesta de solución:

Entrevistas: se realizaron entrevistas no formales al técnico encargado del servicio del gas, con el objetivo de que la persona explique con mayor comodidad todo el proceso de negocio. Estas entrevistas posibilitaron evidenciar la necesidad del desarrollo de un sistema informático que permita gestionar de una forma más adecuada dicho servicio. Con la entrevista realizada fue posible arribar a un conjunto de requisitos del sistema.

Tormenta de ideas (brainstorming): se realizaron reuniones en grupo con la participación de analistas, programadores del proyecto con el objetivo de generar variedades de ideas con el problema en cuestión.

Sistemas existentes: se analizaron soluciones informáticas que permiten de cierto modo gestionar el servicio del gas licuado. Estos contribuyeron a la obtención de nuevas ideas a tener en cuenta en la realización del sistema a desarrollar.

2.4.2 Requisitos funcionales

Los requisitos funcionales son capacidades o funcionalidades que el sistema debe cumplir, están condicionados por la pregunta sobre qué debe hacer el sistema (Sommerville, 2005). Seguidamente se muestra una tabla con los requisitos funcionales de la propuesta de solución.

Tabla 3. Listado de requisitos funcionales de la propuesta de solución.

No.	Nombre	Prioridad para el cliente
RF1.	Mostrar tipos de cilindro	Media
RF2	Crear tipo de cilindro	Alta

RF3	Modificar tipo de cilindro	Alta
RF4	Ver detalle de tipo de cilindro	Media
RF5	Mostrar estados de cilindro	Media
RF6	Crear tipo de estado de cilindro	Media
RF7	Modificar tipo de estado de cilindro	Alta
RF8	Ver detalle de tipo de estado de cilindro	Media
RF9	Mostrar tipos de cliente	Media
RF10	Crear tipo de cliente	Alta
RF11	Modificar tipo de cliente	Alta
RF12	Ver detalle de tipo de cliente	Media
RF13	Mostrar tipos de casos excepcionales	Media
RF14	Crear tipo de caso excepcional	Alta
RF15	Modificar tipo de caso excepcional	Alta
RF16	Ver detalle de tipo de caso excepcional	Media
RF17	Mostrar clientes institucionales	Media
RF18	Crear cliente institucional	Media
RF19	Modificar cliente institucional	Media
RF20	Ver detalles de cliente institucional	Media
RF21	Mostrar cilindros	Media
RF22	Crear cilindro	Media
RF23	Modificar cilindro	Media
RF24	Ver detalles de cilindro	Media
RF25	Mostrar casos excepcionales	Media

RF26	Crear caso excepcional	Alta
RF27	Modificar caso excepcional	Alta
RF28	Ver detalles de caso excepcional	Media
RF29	Mostrar apartamentos	Media
RF30	Ver detalles de apartamentos	Media
RF31	Registrar recogida de cilindro vacío	Alta
RF32	Modificar recogida de cilindro vacío	Media
RF33	Registrar entrega de cilindro lleno	Alta
RF34	Modificar entrega de cilindro lleno	Media
RF35	Mostrar configuración de caso excepcional	Alta
RF36	Crear configuración de caso excepcional	Alta
RF37	Modificar configuración de caso excepcional	Media
RF38	Ver detalles configuración de caso excepcional	Media
RF39	Mostrar configuración de tipo cliente	Media
RF40	Crear configuración de tipo cliente	Media
RF41	Modificar configuración de tipo cliente	Media
RF42	Ver detalles configuración de tipo cliente	Media
RF43	Generar reporte de consumo	Media
RF44	Generar reporte de cantidad de cilindros distribuido	Media
RF45	Generar reporte de cantidad de cilindros sin distribuir	Media
RF46	Generar reporte de cantidad de cilindros vacíos	Media

2.4.3 Requisitos no funcionales

Los requisitos no funcionales, como su nombre sugiere, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento (Sommerville, 2005).

El componente propuesto contará con los siguientes requisitos no funcionales:

Soporte

RNF1: El componente de software contará con toda la documentación asociada a sus procesos para las actividades de soporte.
RNF2: El sistema debe dar la posibilidad de ser mejorado, así como de incorporarle nuevas funcionalidades en caso de ser necesario.

Usabilidad

RNF3: Debe poseer una interfaz intuitiva y agradable al usuario.
RNF4: Los íconos deben estar representados por una imagen acorde a la acción que se realiza sobre los mismos.

Seguridad

RNF5: Se define el acceso al componente y sus funcionalidades mediante la asignación de permisos por roles de usuarios.
RNF6: Los errores deben mostrar la menor cantidad de detalles posible, para evitar brindar información que comprometa la seguridad e integridad del sistema.

Apariencia o interfaz externa

RNF7: El componente debe ser compatible con los navegadores Chrome, Firefox e Internet Explorer.
RNF8: El componente debe poseer un diseño en correspondencia con las pautas definidas en la DIN.

Rendimiento

RNF9: El componente debe responder en un máximo de 10 segundos las solicitudes de los usuarios.
RNF10: El componente debe soportar la conexión simultánea de 500 usuarios.

Hardware

RNF11: Para la ejecución del componente se requiere que la PC cliente cumpla con los siguientes requerimientos: Pentium 4 o superior, 512 MB de RAM y 80 de disco duro disponible como mínimo.
RNF12: Para alojar el componente desarrollado se requiere de un servidor que cumpla con los siguientes requerimientos: 4 GB RAM, CPU de 3 núcleos y al menos 200 GB de Disco Duro.

Software

RNF13: Servidor Web: Apache 2.4.7.
RNF14: Para el despliegue del componente se debe contar con el servidor de bases de datos PostgreSQL en su versión 9.4.
RNF15: Sistema operativo bajo la licencia Libre.

Restricciones de diseño

RNF16: Las herramientas a utilizar para la construcción de prototipos y diagramas son las definidas por la DIN.

Interfaz de comunicación

RNF17: La comunicación entre el servidor de aplicaciones y el cliente se lleva a través del protocolo HTTPS[4].
RNF18: La autenticación del sistema se podrá realizar mediante el protocolo LDAP para la consulta del directorio activo.
RNF19: La comunicación entre la base de datos y el servidor de aplicaciones se lleva a cabo a través del protocolo TCP.

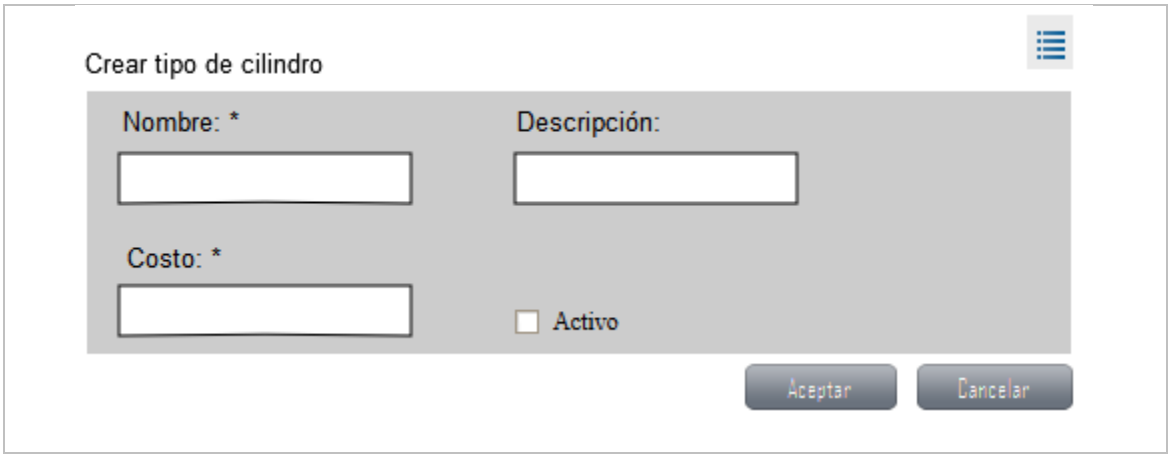
[4] HTTPS: (*Hypertext Transfer Protocol Secure*, Protocolo Seguro de Transferencia de Hipertexto) es un protocolo de aplicación basado en HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

2.4.4 Descripción de requisitos

Los requisitos se describieron haciendo uso de escenarios donde se pudiera mostrar el flujo de las funcionalidades y prototipos. Seguidamente se presenta como ejemplo la descripción del requisito “Crear tipo de cilindro”, el resto de las descripciones correspondiente a los otros requerimientos se encuentran en el documento SGC_RG_ERS del sistema propuesto.

Tabla 4. Descripción del requisito "Crear tipo de cilindro".

Código	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF2	Crear tipo de cilindro	La funcionalidad permite crear un tipo de cilindro. La persona que brinda el servicio selecciona el componente de “Configuración”, luego en la parte lateral derecha, en el ícono que indica la opción “Crear tipo de cilindro” da clic. Se muestran los campos a llenar además de la opción Listar y Detalles.	Alta	Alta
Actor		Precondición	Poscondición	
Administrador				
Prototipo: Crear tipo de cilindro				



En el **Anexo 4** se muestra una serie de prototipos de interfaz correspondientes a la propuesta de solución. En el expediente del proyecto se encuentran los demás prototipos.

2.5 Descripción del sistema propuesto

Se propone la implementación del CGSGL[5] en la Universidad de las Ciencias Informáticas que se integre al Sistema de Gestión del Ciudadano que se encuentra en desarrollo en la UCI. El componente que se pretende desarrollar cuenta con cuatro módulos: Gestión, Configuración, Servicio al cliente y Reporte.

La solución propuesta garantiza una adecuada gestión del servicio del gas en la UCI donde la Dirección de Servicios Generales es la encargada de gestionar dicho servicio. Los usuarios que acceden al sistema pueden ejecutar distintas funciones según el rol asignado, en dependencia de si es un usuario que consume el servicio o es quien brinda dicho servicio.

El módulo de Gestión permite registrar la información de los cilindros ingresados en el punto de despacho en el momento que CUPET hace la entrega. Por otra parte, Configuración contiene las funcionalidades necesarias para garantizar los trámites de creación de tipos de clientes, tipos de cilindros y tipos de casos excepcionales. En el módulo Servicio al cliente, se almacena la información necesaria de la recogida y entrega de los cilindros a los usuarios que se nutren del gas, posibilitando llevar un control de los recursos y de lo consumido en la Universidad. Aquí también es donde se

[5] **CGSGL:** Componente de Gestión del Servicio del Gas Licuado.

almacena la información para los tipos de clientes institucionales. El módulo Reporte facilitará la generación de documentos esenciales para llevar un control del consumo de gas en la UCI.

2.6 Descripción del patrón de arquitectura y diseño

El diseño de un sistema, juega un papel clave en el desarrollo de software porque permite a los ingenieros producir diversos modelos que pueden ser analizados y evaluados con el fin de determinar si se satisfacen los requisitos (Ruiz, 2013). La solución informática propuesta se concibe como un sistema más dentro del SGC[6], razones por la cual se ajustará a la arquitectura MVC[7] por la flexibilidad y ventajas que presenta.

2.6.1 Patrón de arquitectura

El patrón utilizado para desarrollar la propuesta de solución es el Modelo Vista Controlador (MVC). Este es un tipo de diseño que separa en capas bien definidas el desarrollo de una aplicación, esas partes mencionadas anteriormente son tres (Salvador, 2015).

El Modelo maneja los datos y controla todas sus transformaciones. La Vista genera una representación visual del Modelo y muestra los datos al usuario y el Controlador es el objeto que proporciona significado a las órdenes del usuario (Fernández, 2012).

[6] **SGC:** Sistema de Gestión del Ciudadano
[7] **MVC:** Modelo Vista Controlador

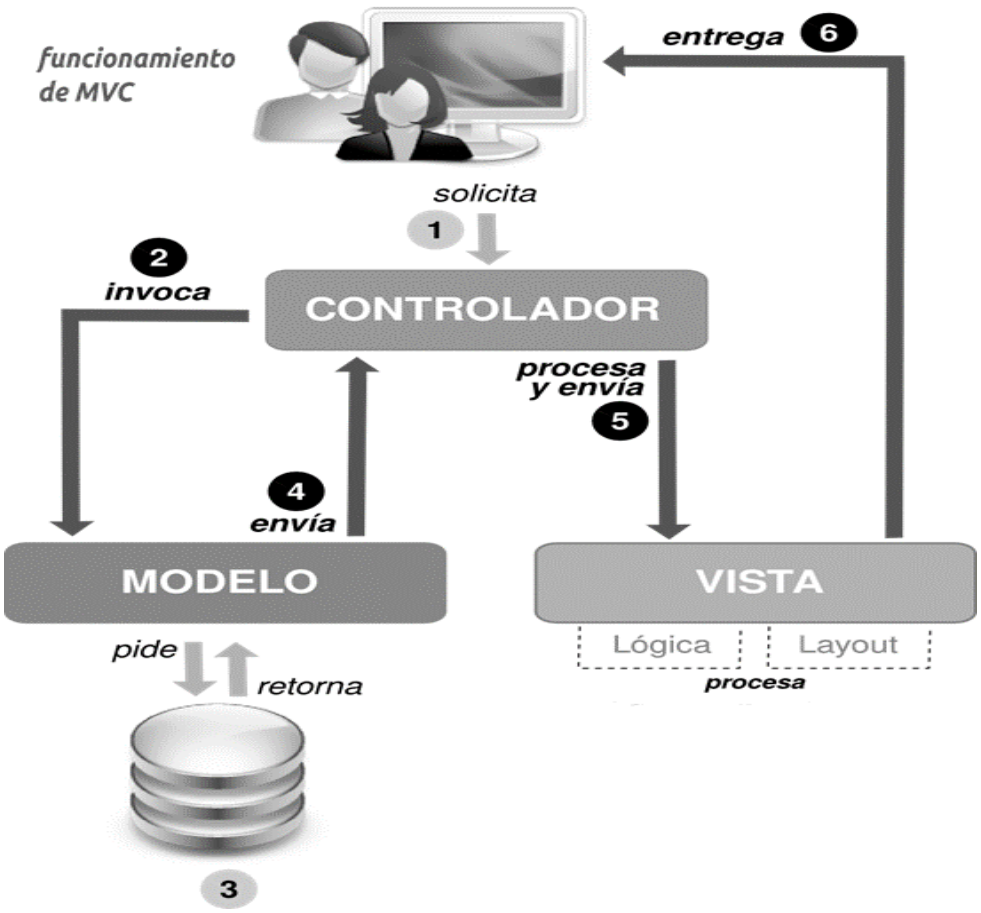


Figura 2. Patrón Modelo-Vista-Controlador.

2.6.2 Patrones de diseño

Los patrones de diseño son una herramienta muy útil en el proceso de desarrollo de software ya que mediante estos se expresan esquemas para definir estructuras de diseño de un sistema informático determinado.

Patrones de diseño GRASP: los patrones GRASP[8] son guías o principios que sirven para asignar responsabilidades a las clases (Maia, 2011). El uso de estos patrones es fundamental en el diseño del software ya que brindan soluciones a posibles problemas en la programación.

[8] **GRASP:** Responsabilidad general de Patrones de Software de asignación.

A continuación, se realiza una descripción de los patrones utilizados en el desarrollo de la propuesta de solución:

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. Este es frecuentemente usado en las clases controladoras, es aquí donde se hace instancia de una entidad determinada para obtener sus valores.
Ejemplo: El uso de este patrón se evidencia (Ver Figura 3) en las acciones de la clase controladora TipoCilindroController.

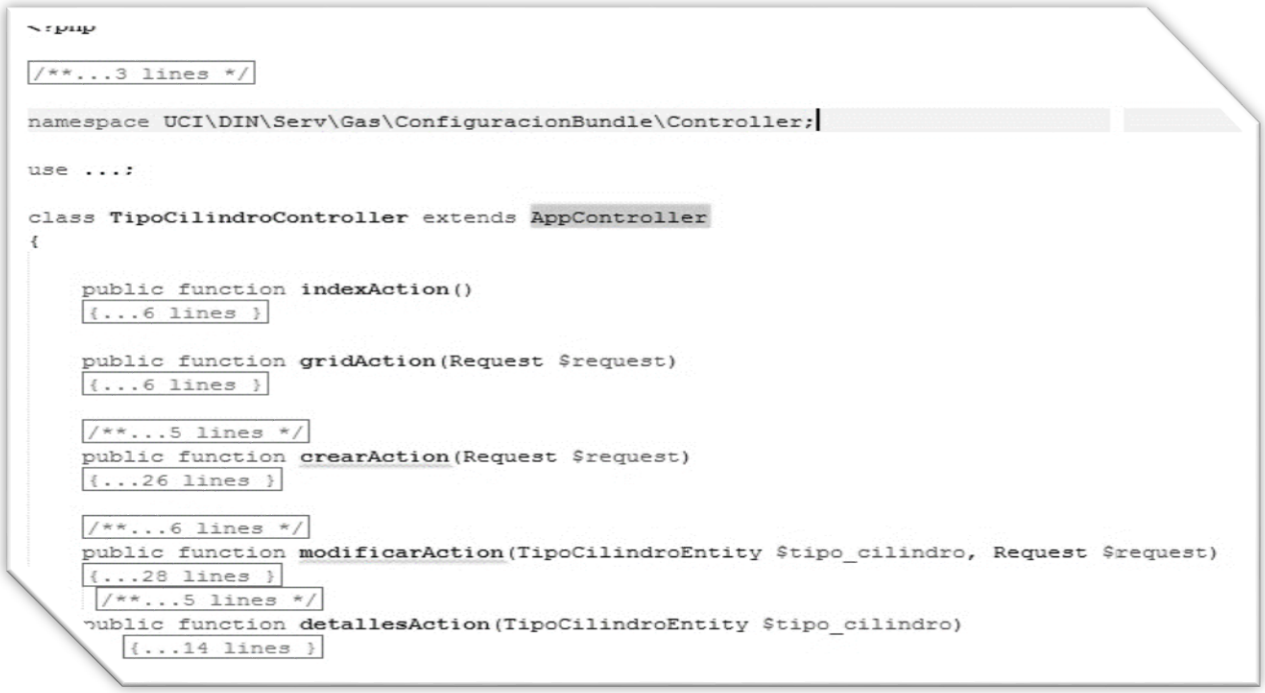


Figura 3. Patrón Creador.

Experto: este patrón plantea que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. En Symfony este patrón se evidencia con la inclusión de la librería Doctrine para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las

entidades, las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan. En la Figura 4 se muestra un ejemplo de la utilización del patrón donde se maneja información de los tipos de cilindros referente a la tabla *sq_gas.tb_ntipo_cilindro*.

Figura 4. Patrón Experto.

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. Este patrón se manifiesta en todo momento en el archivo *routing.yml*, es aquí donde están contenidas todas las rutas para atender las peticiones realizadas

por los usuarios. En la siguiente figura se muestra un ejemplo del uso de este patrón en el archivo *tipoCilindro.yml* de la propuesta de solución.

```
gas_configuracion_tipo_cilindro_index:
    path: /index
    defaults:
        _controller: GasConfiguracionBundle:TipoCilindro:index
    options:
        expose: true
gas_configuracion_tipo_cilindro_grid:
    path: /grid
    defaults:
        _controller: GasConfiguracionBundle:TipoCilindro:grid
    options:
        expose: true
gas_configuracion_tipo_cilindro_crear:
    path: /crear
    defaults:
        _controller: GasConfiguracionBundle:TipoCilindro:crear
    options:
        expose: true
gas_configuracion_tipo_cilindro_modificar:
    path: /modificar/{id}
    defaults:
        _controller: GasConfiguracionBundle:TipoCilindro:modificar
    options:
        expose: true
gas_configuracion_tipo_cilindro_detail:
    path: /detalles/{id}
    defaults:
        _controller: GasConfiguracionBundle:TipoCilindro:detalles
    options:
        expose: true
```

Figura 5. Patrón Controlador.

Patrones de diseño GoF: los patrones GoF[9] se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento. Los patrones creacionales abstraen el

[9] **GoF:** Gang of Four (conocido como pandilla de los cuatro)

proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados, mientras que los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades. Por otra parte, los patrones de comportamiento están relacionados con la asignación de responsabilidades entre objetos (Marquez, 2015).

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 que son instancias únicas donde en estas a través del "*use*" se proporciona la creación de un objeto para acceder a los valores de una clase determinada. En la **Figura 6** se muestra un ejemplo del uso del "*use*" en la clase controladora "*TipoCilindroController*".

```
namespace UCI\DIN\Serv\Gas\ConfiguracionBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use UCI\DIN\Nucleo\BaseBundle\Controller\AppController;
use UCI\DIN\Serv\Gas\ConfiguracionBundle\Form\TipoCilindroType;
use UCI\DIN\Serv\Gas\ConfiguracionBundle\Entity\TipoCilindroEntity;

class TipoCilindroController extends AppController
```

Figura 6. Patrón instancia única.

Observador (*Observer*): este patrón se encarga de definir dependencias entre objetos, de forma que, si alguno cambia su estado automáticamente se notifica y actualizan todos los objetos que dependen de él. En la propuesta de solución se evidencia su utilización en el momento que un cilindro es distribuido automáticamente este cambia su estado a "*en uso*".

2.7 Modelo de datos

Un buen modelo de datos es el punto de partida en el diseño de una aplicación, ya que a través de estos se pueden concebir las relaciones necesarias para manejar grandes cantidades de datos. La propuesta de solución cuenta con un modelo de datos compuesto por 17 tablas (Ver Figura 7).

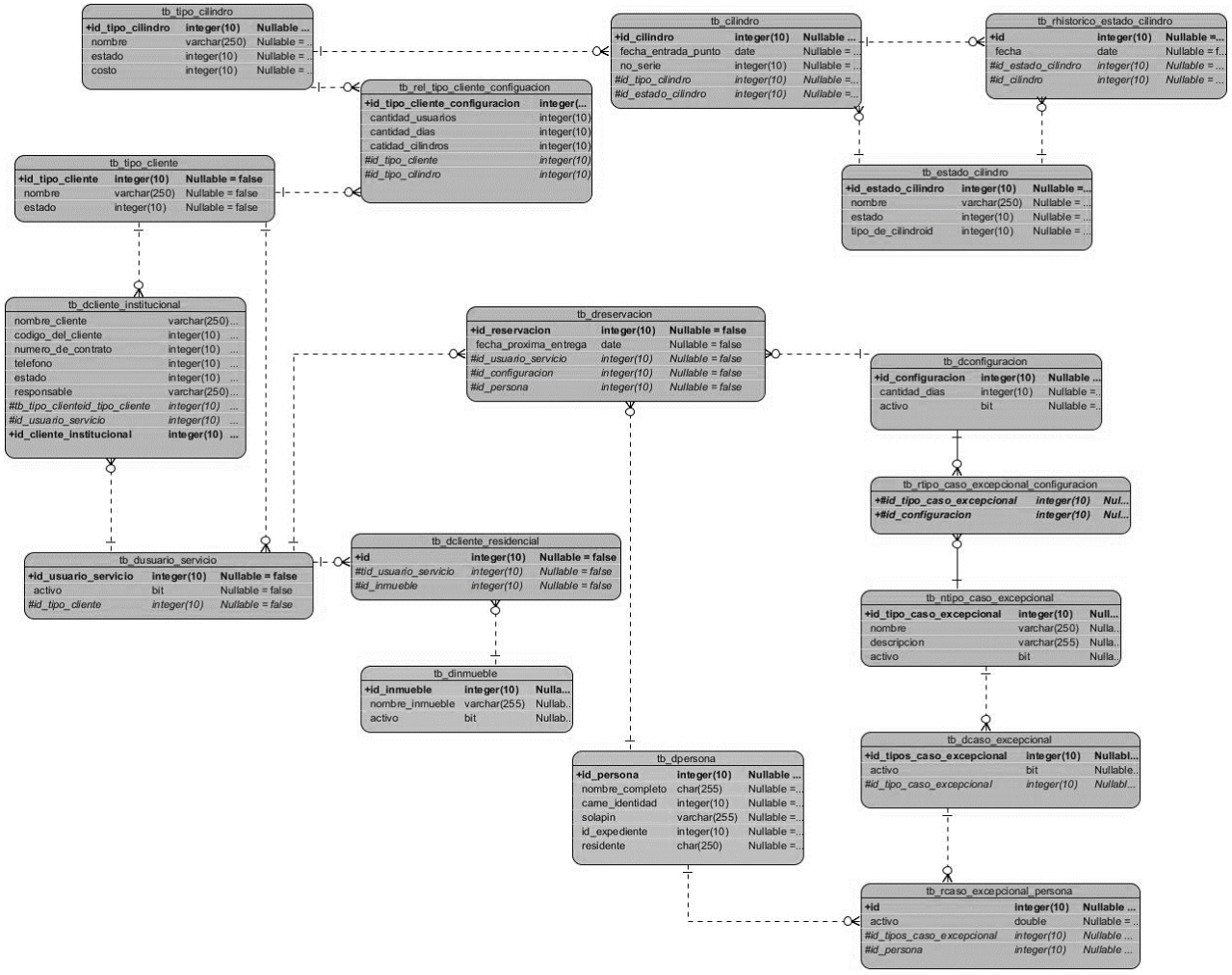


Figura 7. Modelo de datos.

2.8 Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se relacionan en esos nodos (Sparks, 2014).

Como se puede apreciar en la Figura 8 el nodo “PC-Cliente” representa un dispositivo utilizado por el usuario desde el cual se podrá realizar la comunicación a través del protocolo HTTPS, haciendo uso de un navegador web. El nodo “Servidor de Aplicaciones” es el encargado de atender y ofrecer respuesta a cada una de las solicitudes de los usuarios, es decir que aquí es donde se gestiona todo el contenido de la aplicación. Dicho nodo se comunicará con los nodos “Servidor de bases de datos” y “LADP” haciendo uso de los protocolos TCP y LDAP respectivamente. La comunicación con el “LDAP” se encargará de la validación de los usuarios para posteriormente gestionar la información requerida por el usuario en el “Servidor de Bases de Datos”.

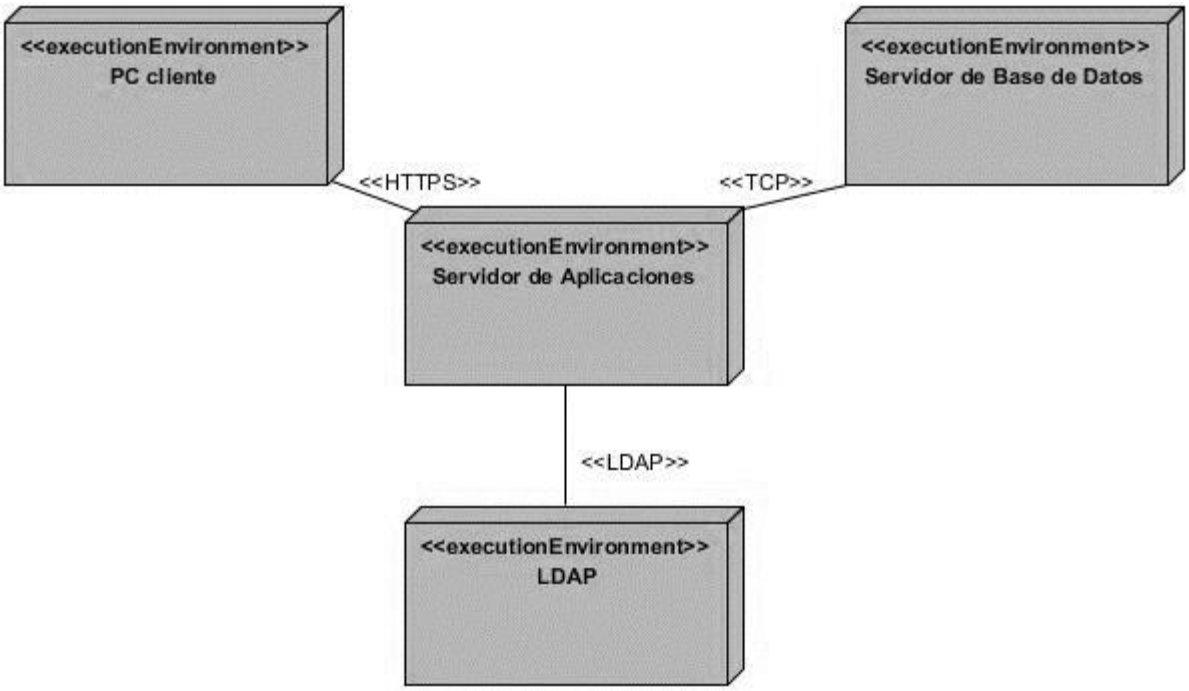


Figura 8. Diagrama de despliegue.

2.9 Conclusiones del capítulo

En este capítulo se trataron una serie de aspectos correspondientes al análisis y diseño del componente de software propuesto llegándose a la siguiente conclusión:

- ✓ La representación y descripción de los artefactos generados garantizaron un mejor entendimiento de los procesos de negocio presentes en el servicio de gas y propició la identificación y obtención de las principales funcionalidades con las que debe cumplir el componente.
- ✓ La caracterización del patrón arquitectónico, patrones de diseño, así como la obtención del modelo de datos y diagrama de despliegue del componente propuesto, permitió una mayor compresión de la arquitectura para el desarrollo de la solución.

Capítulo 3: Implementación y validación del Componente para la gestión del servicio del gas licuado en la UCI

3.1 Introducción

En el presente capítulo se describen los estándares de codificación empleados en el desarrollo de la propuesta de solución posibilitando el mantenimiento y la legibilidad del código de dicha solución. Además, se describen las pruebas realizadas a la solución desarrollada y se analizan los resultados arrojados.

3.2 Estándar de codificación

En el desarrollo de la solución informática se aplican estándares de codificación con el propósito de estandarizar las nomenclaturas y hacer que el producto sea entendible para posteriores modificaciones.

3.2.1 Tamaño de líneas, llaves de apertura y cierre e Identación.

Se debe usar una identación equivalente a 4 espacios. Las llaves "{}" se utilizarán en una línea nueva. La longitud de cada línea de código debe ser de 75 a 80 caracteres como máximo para mantener la legibilidad del código. En la siguiente figura se muestra un ejemplo:

```
public function setActivo($activo)
{
    $this->activo = $activo;

    return $this;
}
```

Figura 9. Identación, llaves de apertura y cierre y tamaño de líneas.

3.2.2 Convención de nomenclatura

La convención de nomenclaturas se utiliza con el objetivo de estandarizar la forma en que se nombran las variables, clases y funciones dentro del código del componente propuesto.

Variables: están regidas por la nomenclatura **camelCase**[10]**.** Estas empiezan con letra inicial minúscula. En caso de ser palabras compuestas la primera letra de cada palabra comienza con minúscula y las demás con mayúscula. A continuación, se muestra un ejemplo:

```
$formName = 'base_nomenclador';
$formulario = $this->createForm(new TipoCilindroType($this->container));
```

Figura 10. Variables.

Clases: cada palabra se escribe con letra inicial mayúscula como se muestra a continuación:

```
class ClienteInstitucionalController extends AppController

class Controller extends ContainerAware
```

Figura 11. Clases.

Funciones: están regidas por la nomenclatura **camelCase.** El nombre de cada función comienza con minúscula, para el caso de nombres compuestos la primera letra de cada palabra comienza con minúscula. Las funciones que lleven parámetros son separados por una coma y espacio. Ver la siguiente figura.

```
public function forward($controller, array $path = array(), array $query = array())
{...6 lines }
public function crearAction(Request $request)
{...28 lines }
```

Figura 12. Funciones.

[10] **CamelCase:** es un estilo de escritura que se aplica a frases o palabras compuestas. En este caso se usa el *lowerCamelCase,* cuando la primera letra de cada una de las palabras es mayúscula con la excepción de que la primera letra es minúscula. Ejemplo: ejemploDeLowerCamelCase

3.2.3 Ficheros

El uso de los estándares de codificación, poseen gran importancia ya que aquí es donde se define las palabras empleadas para identificar a que parte pertenece un fichero determinado. Seguidamente se explica la nomenclatura a seguir en las clases controladoras, modelos y vistas.

Controladoras: todas las clases controladoras terminan con la palabra Controller.
Ejemplo: **EstadoSolicitudController.php**

Modelos: todas las clases modeladoras terminan con la palabra Manager.
Ejemplo: **ClienteInstitucionalManager.php**

Vistas: relacionado con el formulario y/o vista que representa. Para el nombre de estos ficheros se emplea el sufijo **"_view".**
Ejemplo: **nomenclador_view.html.twig**

3.2.4 Estructuras de control

Las estructuras de control incluyen sentencias if, for, foreach, while, etc. 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 es opcional su uso. Seguidamente se muestra un ejemplo de su uso en la clase **EstadoSolicitudController.**

```
if ($formulario->isValid())
{
    $em = $this->getDoctrine()->getManager();
    $em->persist($formulario->getData());
    $em->flush();
    $message = $this->get('translator')->trans('app.msg.inf_success');
```

Figura 13. Estructuras de control.

3.2.5 Buenas prácticas

Para facilitar la legibilidad del código los valores lógicos se escriben con mayúscula y se emplea un salto de línea luego de definir una función y antes de una estructura de control. También se aplica el uso de comentarios para facilitar la comprensión del código.

3.2.6 Operadores

Todos los operadores binarios como: +, -, /, =, ==,>, entre otros, deben tener un espacio antes y después del operador, para facilitar la lectura. Ver la siguiente figura.

```
$em = $this->getDoctrine()->getManager();
```

Figura 14. Operadores.

3.3 Estrategias de validación de requisitos

La validación de requisitos examina las descripciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin omisiones, que los errores identificados hayan sido corregidos y que el resultado se ajusta a los estándares establecidos en el proyecto y el producto (Pressman, 2001).

3.3.1 Criterios de validación de requisitos

El desarrollo de software empleando la metodología DAC propone criterios a tener en cuenta en la validación de los requerimientos del cliente, donde respondiendo determinadas interrogantes para cada requisito se determina si es aprobado o no en el documento SGC-GAS_CVRC_A del expediente del proyecto. Entre dichas interrogantes figuran las siguientes:

- ✓ ¿El proveedor del requisito es un proveedor válido?
- ✓ ¿El requisito tiene un identificador único?
- ✓ ¿El requisito es modificable?
- ✓ ¿El requisito no es ambiguo?
- ✓ ¿El requisito está completo?
- ✓ ¿El requisito es congruente con otros requisitos relacionados?

- ✓ ¿El requisito puede ser implementado?
- ✓ ¿El requisito puede ser probado?
- ✓ ¿El resultado de la evaluación de impacto es positivo?
- ✓ ¿El requisito es correcto?

3.3.2 Validación de requisitos

Para validar los requisitos se emplearon las siguientes técnicas:

- ✓ **Construcción de prototipos:** la construcción de los prototipos de interfaz es una de las técnicas empleadas para la validación de los requerimientos. El uso de esta permitió que el cliente pudiera conocer como es el componente que los autores implementan, permitiéndole a este aprobar la idea o corregir elementos ajenos a los requisitos funcionales descritos.
- ✓ **Revisión técnica formal de requisitos:** en el uso de esta técnica se realizaron reuniones donde se examinó la descripción del sistema buscando errores en el contenido o en la interpretación, ideas donde hicieran falta aclaraciones, inconsistencias, así como poder identificar requisitos contradictorios.

3.3.3 Resultados de aplicar los criterios de validación

Al finalizar la revisión de los requisitos se identificaron algunas inconsistencias que fueron corregidas de inmediato. Entre las más comunes se encuentran:

- ✓ Errores ortográficos o tipográficos.
- ✓ La interpretación incorrecta de algunas funcionalidades y requerimientos solicitados por el cliente.

3.4 Validación de la aplicación

Las pruebas de software son actividades donde un sistema o uno de sus componentes se ejecuta en circunstancias previamente descritas, registrándose los resultados obtenidos (Pressman, 2001).

En la presente investigación se emplearon algunas de las pruebas definidas por Pressman[11] en su libro “Ingeniería de software: Un enfoque práctico”, donde se pretende que el producto se pruebe desde la parte más pequeña hasta la más grande en forma espiral haciendo uso de niveles, tipos y técnicas de pruebas.

Para comprobar que el sistema implementado es válido y que cumple con los requisitos acordados con el cliente, se propone la siguiente estrategia de pruebas:

3.4.1 Métodos de pruebas

Pruebas de caja blanca: Las pruebas de caja blanca están dirigidas a las funciones internas del sistema. Estas son una verificación técnica del software que los desarrolladores pueden usar para examinar si el código trabaja como se esperaba. Mediante este método de prueba se pueden obtener casos de prueba que garanticen que se ejercita por lo menos una vez las decisiones lógicas en sus vertientes verdadera y falsa (Pressman, 2001).

Pruebas de caja negra: Las pruebas de caja negra se desarrollan sobre la interfaz visual del software y se encargan de verificar que las funciones que debe desempeñar el sistema estén en correspondencia con la interfaz. Se centran en los requisitos funcionales de la aplicación, sin internarse en el funcionamiento interno de la misma. Mediante la realización de estas pruebas se pueden encontrar errores de interfaz, funciones incorrectas, errores de salida y problemas con el acceso a datos (Pressman, 2001).

3.4.2 Pruebas de integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los componentes probados anteriormente y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño (Pressman, 2001).

[11] **Pressman:** es una autoridad internacionalmente reconocida en la mejora del proceso de software y en tecnologías de ingeniería de software.

Integración incremental

Al culminar el desarrollo de un sistema se tiende a una integración no incremental; es decir, a combinar todos los componentes y probar todo el programa en su conjunto. El resultado puede ser caótico con un gran conjunto de fallos y la consiguiente dificultad para identificar el componente (o componentes) que los provocó. Por otra parte, se puede aplicar la integración incremental en la que el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir (Pressman, 2001). Por esta razón se decidió escoger un enfoque incremental para la realización de las pruebas de integración de la solución.

Resultados de las pruebas de integración

Para la validación de la solución se probó la integración de la misma con el resto de los componentes del Sistema de Gestión del Ciudadano. Los resultados de las pruebas que se describen están relacionados con los componentes Seguridad y Base. La siguiente tabla muestra el caso de prueba de integración realizado al componente Base. Para consultar el resultado del resto de las pruebas ver el **Anexo #5.** Luego de haber aplicado las pruebas de integración, no se detectaron errores asociados a la interrelación entre el Componente del Servicio del Gas con los componentes del SGC.

Tabla 5. Prueba de integración componente Base.

Caso de prueba: Integración de Servicio del Gas	
Componente al que se integra	Base.
Condiciones de ejecución	En el componente Base se debe haber introducido la información en la base de datos central y que la misma esté disponible.
Descripción de la prueba	Verificar que el Servicio del Gas es capaz de comunicarse con la base de datos central y hacer uso de la información del personal involucrado en el proceso del gas.
Entradas/Pasos de ejecución	Desde el Servicio del Gas se gestiona la información del personal involucrado en el proceso directamente con el componente Base.
Resultado esperado	Se gestiona la información del personal involucrado.
Evaluación	Satisfactoria.

3.4.3 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 (Pressman, 2001). 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 (Pressman, 2001). Lo que posibilita determinar cuan rápida es la respuesta del sistema ante un conjunto de peticiones concurrentes.

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 para comprobar su capacidad de respuesta (Pressman, 2001).

Resultados de las pruebas de rendimiento y resistencia

Una prueba de rendimiento se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas. Para llevar a cabo las pruebas de rendimiento y resistencia se utilizó la herramienta JMeter 2.12 descrita en el acápite 1.4.7 y se utilizó una copia local del componente en una computadora con sistema operativo Linux Mint 17.2, un microprocesador AMD con 4 GB de RAM a 1.30 GHz de velocidad y 500 GB de almacenamiento. La prueba realizada consistió en definir tres pruebas de 25, 50 y 75 hilos cada una, los cuales simulan 25, 50 y 75 accesos de usuarios respectivamente.

Para un mejor entendimiento de la tabla que a continuación se presenta, se explica cada parámetro que la compone:

#Muestras: cantidad de hilos utilizados para la URL.
Min: tiempo mínimo que demora un hilo en acceder a una página.
Max: tiempo máximo que demora un hilo en acceder a una página.
Media: tiempo promedio en milisegundos para un conjunto de resultados.
Error: indica la relación entre el total de peticiones y el número de peticiones que originaron errores.

Pet/seg: hace referencia al número de peticiones que el servidor puede procesar en un segundo.
Kb/seg: rendimiento medido en Kilobytes por segundo.

En la tabla siguiente solo se muestran los valores totales obtenidos para hacer uso de la funcionalidad "Crear tipo de cliente" con 50 y 75 hilos respectivamente. Para consultar el resto de los casos de prueba ver **Anexo 6**.

Tabla 6. Resultados de la prueba de rendimiento con el acceso de 50 usuarios.

Aplicado a	#Muestras	Tiempo de ejecución (ms)			Rendimiento		
		Mín.	Máx.	Media	% Error	Pet/seg	Kb/seg
Crear tipo de cilindro	50	425	6179	3351	0.0%	7.0	220.2

Como se muestra en la Tabla 6, el componente desarrollado para 50 usuarios conectados de forma concurrente, es capaz de procesar 220.2 Kb/seg en una media 3.351 segundos, lo que equivale a 7 peticiones por segundo.

Los valores totales obtenidos para 75 hilos se muestran en la Tabla 7.

Tabla 7. Resultados de la prueba de rendimiento con el acceso de 75 usuarios.

Aplicado a	#Muestras	Tiempo de ejecución (ms)			Rendimiento		
		Mín.	Máx.	Media	% Error	Pet/seg	Kb/seg
Crear tipo de cilindro	75	848	7229	3877	0.0%	6.2	204.7

Como se muestra en la Tabla 7, el componente desarrollado para 75 usuarios conectados de forma concurrente, es capaz de procesar 204.7 Kb/seg en una media 3.877 segundos, lo que equivale a 6.2 peticiones por segundo.

De manera general estos resultados son favorables teniendo en cuenta que cumplen con el requisito no funcional de rendimiento RNF9.

3.4.4 Pruebas funcionales

Las pruebas funcionales tienen por objetivo probar que los sistemas desarrollados cumplen con los requisitos funcionales del software. Los probadores o analistas de prueba no enfocan su atención en cómo se generan las respuestas del sistema, sino en el funcionamiento de la interfaz del sistema (Pressman, 2001).

Partición equivalente

La partición equivalente es una técnica de prueba de caja negra que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman, 2001).

Para la realización de las pruebas funcionales se generaron los artefactos “Diseño de casos de pruebas basado en requisitos”. Por cada requisito funcional del sistema se generó un documento donde se recogen todos los datos necesarios para probar la interfaz. Seguidamente se muestra el diseño de caso de prueba que responde al requisito funcional “Mostrar cilindro”.

Tabla 8. Diseño de caso de prueba "Mostrar cilindro".

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Mostrar datos correctamente.	Mediante este escenario se muestra un listado de cilindros con las opciones: Modificar, Ver detalle y Crear por cada cilindro.	El sistema muestra un listado de cilindros.	1. El usuario con rol de administrador se autentica. 2. Una vez autenticado, selecciona la opción Cilindro en el menú de agrupaciones en el módulo de Gestión. 3. Se muestra una interfaz con un grid listando los cilindros.
EC 1.2 No existen elementos creados.	Mediante este escenario en caso de que no exista	El sistema muestra un listado vacío.	1. El usuario con rol de administrador se autentica.

	creado ningún cilindro se muestra un listado vacío.		2. Una vez autenticado, selecciona la opción Cilindro en el menú de agrupaciones en el módulo de Gestión. 3. Se muestra una interfaz con un listado vacío.
EC 1.3 Seleccionar una cantidad de elementos por página mayor que la cantidad de elementos que existen en el sistema.	Mediante este escenario el usuario selecciona una cantidad de elementos por página mayor que la cantidad de elementos que existen en el sistema. Para ello debe seleccionar los valores (10, 20 ó 40) porque el 5 se muestra por defecto.	El sistema muestra en el listado la cantidad de elementos que tiene indicando en la parte inferior del listado el número de elementos que existen. Muestra la cantidad de elementos que están registrados que en este caso es menor que la cantidad seleccionada.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Servicio y luego el módulo de Gestión. El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Cilindro" la funcionalidad "Listar Cilindro". El sistema muestra los cilindros registrados. El usuario selecciona un número de página mayor que la cantidad de los elementos registrados en el sistema.
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.	El sistema muestra la página solicitada por el usuario.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Servicio y luego el módulo de Gestión. El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Cilindro" la funcionalidad "Listar Cilindro". El sistema muestra los cilindros registrados.

			El usuario selecciona un número de página mayor que la cantidad de los elementos registrados en el sistema.
EC 1.5 Escribir 0 en el número de página.	Mediante este escenario el usuario escribe debajo del listado el número cero.	El sistema muestra la página en la que se encontraba el usuario.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Servicio y luego el módulo de Gestión. El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Cilindro" la funcionalidad "Listar Cilindro". El sistema muestra los cilindros registrados. El usuario escribe el número cero (0) y presiona Enter en el teclado.
EC 1.6 Dejar vacío el número de página.	Mediante este escenario el usuario deja vacío el número de la página a la que desea acceder.	El sistema muestra la primera página.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Servicio y luego el módulo de Gestión. El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Cilindro" la funcionalidad "Listar Cilindro". El sistema muestra los cilindros registrados. El usuario deja vacío el campo del 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.	El sistema muestra la última página del listado.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Servicio y luego el módulo de Gestión. El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Cilindro" la funcionalidad "Listar Cilindro". El sistema muestra los cilindros registrados El usuario escribe el número de la página 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 un valor numérico.	Mediante este escenario el usuario escribe en el campo de texto debajo del listado algún carácter que sea diferente a un valor numérico (carácter extraño, letras, etc.).	El sistema se mantiene en la misma página.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Servicio y luego el módulo de Gestión. El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Cilindro" la funcionalidad "Listar Cilindro". El sistema muestra los cilindros registrados. El usuario escribe en el campo del número de la página un carácter diferente 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.	El sistema muestra el listado vacío.	El usuario una vez autenticado en el sistema selecciona el Subsistema de Servicio y luego el módulo de Gestión. El sistema muestra las opciones de menú y el usuario selecciona en la agrupación funcional "Cilindro" la funcionalidad "Listar Cilindro".

Resultados obtenidos

A continuación, se muestra la relación de no conformidades por iteración detectadas durante la realización de las pruebas funcionales a los requisitos implementados.

Tabla 9. Relación de no conformidades por iteración.

Iteraciones	Cantidad de no conformidades	Asociadas a
1ra	43	Errores de interfaz, funciones incorrectas o ausentes.
2da	21	Errores de interfaz, de validación y ortografía.
3ra	0	-

A continuación, en la siguiente figura se puede apreciar de una manera más sencilla el comportamiento de las no conformidades detectadas durante cada iteración de las pruebas funcionales aplicadas a la propuesta de solución.

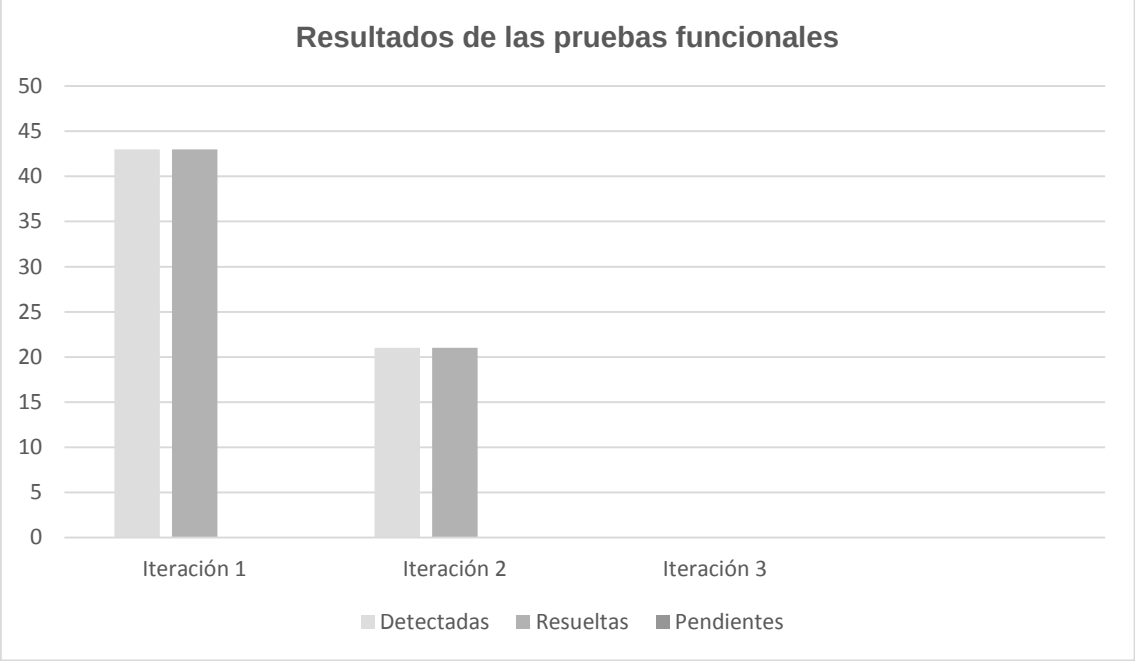


Figura 15. Relación de no conformidades por iteración.

3.5 Conclusiones del capítulo

En este capítulo se abordaron aspectos correspondientes a la implementación y validación del componente para el servicio de gas licuado en la UCI llegándose a las siguientes conclusiones:

- ✓ La descripción de los estándares de codificación permitió una mejor organización y comprensión del código.
- ✓ La validación de los requisitos identificados posibilitó obtener un listado de requisitos correctamente redactados.
- ✓ La ejecución de pruebas al componente permitió detectar las deficiencias presentes, subsanarlas en el menor tiempo posible y ofrecer una solución con mayor calidad.

Conclusiones

Al término de la presente investigación se concluye lo siguiente:

- ✓ El estudio de los sistemas similares permitió identificar características que sirvieron como base para elaborar la propuesta de solución.
- ✓ El estudio del proceso del servicio del gas licuado en la UCI propició un mejor dominio del funcionamiento en la ejecución de dicho proceso.
- ✓ La obtención, descripción y validación de los requisitos definieron las características técnicas y funcionales de la propuesta de solución.
- ✓ La aplicación de pruebas de software permitió detectar y corregir no conformidades identificadas durante la etapa de desarrollo, garantizando el cumplimiento de los requisitos de software.
- ✓ El componente desarrollado permite al encargado de brindar el servicio del gas, la generación de reportes en correspondencia con lo consumido en la UCI, así como llevar un control de la entrega y recogida de los cilindros por parte de los usuarios consumidores del servicio.

Recomendaciones

1. Implementar un flujo de gestión de solicitudes del servicio del gas licuado para clientes residenciales.

Bibliografía

ABASTIBLE. Abastible. [En línea] 2014. [Citado el: 28 de noviembre de 2015.] http://www.abastible.cl. 2014.

ACOSTA, Julio. *Medición de atributos POO en frameworks de desarrollo PHP.* 2012.

ACUNETIX. Audit your website security with Acunetix Web Vulnerability Scanner. *ACUNETIX.* [En línea] 2015. [Citado el: 10 de enero de 2016.] http://www.acunetix.com/. 2015.

BALLESTEROS, SOLER SANCHEZ Y. M., AGUILERA MONTERO M., LÓPEZ ÁREAS E., DELGADO GONZALEZ O. *Importancia de la gestión de la información y el conocimiento en el proceso de cambio organizacional.* 2008.

COBO A., GÓMEZ P. , PÉREZ D. y ROCHA R. *Tecnologías para el desarrollo de aplicaciones web.* 2005.

COMESAÑA, J. L.. *Entornos de Desarrollo del curso de "Desarrollo de Aplicaciones Web".* 2012.

CONDORI, S. F. T. [En línea] 2010. [Citado el: 10 de noviembre de 2015.] https://tallerinf281.wikispaces.com/file/view/METODOLOG%C3%8DAS+TRADICIONALES.pdf. 2010.

FOUNDATION-TIKA. Foundation-Tika. *Apache Tika.* [En línea] 2014. [Citado el:] 2. http://tika.apache.org. 2014.

GARCÍA LIMA L., MONTERO REYES I. *Sistema para el registro y control de los procesos de la vida interna del PCC en la UCI.* La Habana : s.n., 2015.

GARZÓN, T. *SISTEMAS GESTORES DE BASES DE DATOS.* 2010.

GASCO. Gasco. [En línea] 2014. [Citado el: 30 de noviembre de 2015.] http://www.gasco.cl. 2014.

GASENLINEA. Precios de Cilindros de Gas Licuado de Petróleo en Línea. [En línea] 2014. [Citado el: 29 de noviembre de 2015.] http://www.gasenlinea.gob.cl. 2014.

GAUCHAT, J. D. *El gran libro de HTML5, CSS3 y Javascript.* Primera edición. Barcelona : MARCOMBO, S.A, 2012.

GAUCHAT, J. D. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona : Marcombo, 2012.

GERTRUDYS. Buenas Tareas. [En línea] 17 de noviembre de 2010. [Citado el: 25 de noviembre de 2015.] http://www.buenastareas.com/ensayos/Sistema-De-Reservaciones/1131441.html. 2010.

RUÍZ, F. *Guión Visual Paradigm for UML.* 2013.

JOHANSEN, O. B. Introducción a la Teoría General de Sistemas. [En línea] 2013. [Citado el: 5 de noviembre de 2015.] http://www.manuelugarte.org/modulos/teoria_sistemica/introduccion_a_la_teoria_general_de_sistemas_bertoglio.pdf. 2013.

LARRAINVIAL . *Emisión de Bonos Corporativos.* 2015.

LEMAY, L. 2012. *Teach Yourself Web Publishing with HTML 4 in a Week.* USA : Sams Indianapolis, 2012. 1575213362.

LIPIGAS. Lipigas. [En línea] 2014. [Citado el: 30 de noviembre de 2015.] http://www.lipigas.cl. 2014.

LÓPEZ, L. F. *Tutorial de PostgreSQL.* 2014.

MAIA, K. TuxNots. [En línea] 3 de Diciembre de 2011. [Citado el: 2 de febrero de 2016.] https://sites.google.com/site/tuxnots/materias-de-la-facu/metodologia-de-sistemas/patronesgrasppatronesgofdiferenciaentregraspygof. 2011.

RUIZ MARTÍNEZ, M. *FUNDAMENTOS DE SISTEMAS.* 2014.

MARQUEZ, J. Patrones de diseño(GoF). [En línea] 2015. [Citado el: 3 de Febrero de 2016.] https://infow.wordpress.com/category/patrones-de-disenogof/. 2015.

MERIFONT. Pencil Project: Crea prototipos para aplicaciones y sitios web. [En línea] 2012. [Citado el: 11 de Enero de 2016.] http://www.neoteo.com/pencil-project-crea-prototipos-para-aplicaciones-y. 2012.

NEIL, M. *Beginning Databases with PostgreSQL.* 2005.

NETCRAFT. [En línea] Noviembre de 2015. [Citado el: 2 de diciembre de 2015.] http://news.netcraft.com/. 2015.

FERNÁNDEZ, Y. *Patrón Modelo Vista controlador* 1729-3804, La Habana : Revista Telem@tica, 2012, Vol. 11. 2012.

PGPB. Pemex Gas y Petroquímica Básica. [En línea] 2015. [Citado el: 29 de noviembre de 2015.] http://www.gas.pemex.com. 2015.

PRESSMAN. 2001. *Ingeniería de software: Un enfoque práctico.* Quinta edición. Madrid : s.n., 2001.

RUIZ, F. INGENIERÍA DEL SOFTWARE I. [En línea] 2013. [Citado el: 2 de Febrero de 2016.] http://www.ctr.unican.es/asignaturas/is1/is1-t04-trans.pdf. 2013.

SALINA, Y. *Sistema de gestión del servicio del gas licuado en la Universidad de las Ciencias Informáticas.* 2012.

SALVADOR. *Propuesta de implementación simple del patrón de diseño Modelo Vista Controlador sobre PHP.* Quito : s.n., 2015.

SÁNCHEZ, A. *Proceso de Desarrollo de Software DAC.* 2013.

SÁNCHEZ, J. *Implantación de aplicaciones Web del Ciclo Administración de Sistemas Informáticos en Red.* 2012.

SOMMERVILLE. *Ingeniería de Software.* Madrid : Pearson Educación S.A, 2005. 84-7829-074-5. 2005.

SPARKS, G. 2014. *El Modelo de Proceso de Negocio.* Argentina : Solus S.A, 2014.

STANTON, E. y W. 2006. *Definición y Características de los Servicios.* 2006.

CERVANTES OJEDA, J. y GÓMEZ FUENTES, M. del C. *Taxonomía de los modelos y metodologías de desarrollo de software más utilizados.* Mexico : s.n., Universidades 2012 LXII(52). 2012.

THINKANDSELL. Sistemas de Gestión Normalizados. [En línea] 2013. [Citado el: 20 de noviembre de 2015.] http://thinkandsell.com/servicios/consultoria/software-y-sistemas/sistemas-de-gestion-normalizados/. 2013.

RODRÍGUEZ TELLO, E. 2011. Importancia de las pruebas de software. Tamaulipas : s.n., 2011.

REAL ACADEMIA ESPAÑOLA. Diccionario de la lengua española. [En línea]. Real Academia Española. [Citado el: 4 de diciembre de 2014.]. Disponible en: [http://lema.rae.es/drae/?val=informaci%C3%B3n]. 2014.

NUGRAHA, A. Indexing Bibliographic Database Content Using MariaDB and Sphinx Search Server. [En línea]. The Code4Lib Journal – Indexing Bibliographic Database Content Using MariaDB and Sphinx Search Server. [Citado el: 28 de noviembre de 2015.] Disponible en: [http://journal.code4lib.org/articles/9793]. 2014.

Consejo Económico Social. Tecnologías de la información y las comunicaciones para un desarrollo social y económico incluyente. [En línea] 2014. [Citado el: 5 de noviembre de 2015.] http://unctad.org/meetings/es/SessionalDocuments/CSTD2014_17th_Report_es.pdf. 2014**.**