Universidad de las Ciencias Informáticas

Facultad 3

Personalización del Sistema de Gestión para la Atención a la Población en el Ministerio del Comercio Interior

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

Autores:

Ramsés González Toledo

Leonel Domínguez Ávila

Tutores:

MSc. YordanisGarcía Leiva

Ing. Yeleny Almora Galvez

Septiembre, 2020

"Año 62 del Triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de laUniversidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

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

_____ _____

Firma del autor **Firma del autor**

Ramsés González Toledo Leonel Domínguez Ávila

_____ _____

Firma del tutor **Firma del tutor**

MSc. Yordanis García Leiva Ing. Yeleny Almora Galvez

AGRADECIMIENTOS

De Leonel:

Primeramente, quiero empezar agradecer a mi familia en especial a mis padres Ana Belkis, Leonardo y a mi hermano Yasnier ya que ellos son el motor impulsor de mi vida y las personas que siempre estuvieron conmigo en todo momento.

A mi novia Yanara que también estuvo junto a mí en todo este proceso de tesis.

A mis tutores Yordanis García Leiva y Yeleny Almora Gálvez por todo su apoyo y dedicación durante este procesó de tesis

A mi Profesor y amigo Hugo por apoyarme y siempre darme esa palmada en la espada que necesitaba.

A todos los especialistas del centro CEGEL por aportar su granito de arena.

A mis amigos de la UCI y compañeros de aula.

A todas las personas en general que estuvieron vinculadas a mi durante este proceso.

También quiero darle gracias a Dios por permitir que mi sueño de ser informático se hiciera realidad.

DeRamsés:

Quiero agradecer a toda mi familia, ya que esta es la que me trajo al mundo y desde ese momento el suyo cambió para dedicarse al mío, gracias a eso pude cumplir mi sueño y deseo de ser informático, agradezco a mi madre Martha, mi padre Raudel y mis hermanas Johana y Laurita.

Agradezco muchísimo a mi novia Gabriela por estar a mi lado siempre y apoyarme en cada decisión, gracias por ser especial.

Agradezco a mi padrastro Raudel por apoyarme en mis viajes a la escuela.

Agradezco a mis dos grandes profesores y tutores de esta tesis Yordanis García Leyva y Yeleny Almora Gálvez, por apoyarme tanto en este proceso de tesis y por enseñarme tanto en mi etapa de estudiante.

Agradezco a todos los trabajadores de CEGEL en general, especialmente a Rafa, Robin, Rolando, Yamila, Hernán, Morciego, Arnel, por apoyarme tanto en la programación de la solución propuesta en la tesis.

A todos los compañeros de clases, y cada profesor que me enseño algo útil.

Agradezco mucho a dios por permitirme vivir una vida buena y agradable, con personas que me aman, y por permitir que uno de mis sueños, el de ser informático se cumpliera.

DEDICATORIA

De Leonel:

A todas las que aportaron su granito en este procesó en especial a mis padres, mi hermano, mi novia, a mis tutores y a todas esas personas q siempre estuvieron hay cuando las necesité.

De Ramsés:

Dedicado a mi familia, mi novia, mis profesores, mis tutores, mis amigos, a todas esas personas que siempre me han apoyado y confiado en mí.

RESUMEN

El Sistema de Gestión para la Atención a la Población constituye una aplicación web capaz de gestionar el flujo básico para la atención a la población en una organización o empresa. La aplicación permite procesar información relacionada con quejas, reclamaciones, denuncias y sugerencias, brindando la posibilidad de crear un expediente único para cada ciudadano que presente un escrito de queja y a partir de este se desencadena todo el flujo de gestión del proceso de atención a la población. La aplicación cuenta con un conjunto de funcionalidades que viabilizan el trabajo de los especialistas, pero aún no cubre la totalidad de las características de los procesos de atención a la población en cada ministerio.La presente investigación tiene como objetivo desarrollar una personalización del Sistema de Gestión para la Atención a la Población para el Ministerio del Comercio Interior, adaptando este software a las características del flujo de trabajo de la Dirección de Protección al Consumidor en este ministerio. El desarrollo de la solución está guiado por el uso de la metodología de desarrollo de software Proceso Unificado Ágil en su variación para la Universidad de Ciencias Informáticas y la utilización de tecnologías de código abierto. El resultado de la investigación fue validado aplicando pruebas de software en sus cuatro niveles. Además, se verifica el cumplimiento del objetivo general de la investigación a partir de la definición de un conjunto de indicadores que permiten evaluar la relación causa-efecto de la variable independiente sobre las variables dependientes de la investigación.

PALABRAS CLAVES: atención, comercio interior, ministerio, población

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La atención a la población está integradapor vías, procedimientos y acciones de interacción sistemática entre cuadros y funcionarios,dirigidas fundamentalmente, al control, tramitación, orientación y atención de quienes acuden o se dirigen al organismo para plantear sus insatisfacciones(Ecured, 2020).En la actualidad se desarrollan software que informatizan el proceso de atención a la población en las entidades. Un ejemplo de estasaplicaciones informáticas lo constituye el Sistema de Gestión para la Atención a la Población (SIGAP) desarrollado por especialistas del Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI).

A pesar de las funcionalidades y ventajas que brinda el SIGAP, este no cubre en su totalidad las características de los procesos de atención a la población en cada ministerio. En los cuales existen áreas, oficinas o departamentos encargados de gestionar la atención a la población en función del objeto social al cual estén destinados. Un ejemplo de esto lo constituye el Ministerio del Comercio Interior (MINCIN).

El MINCIN es el organismo encargado de dirigir, ejecutar, y controlar la Política del Estado y del Gobierno cubano en cuanto al comercio interior mayorista y minorista. Este se encarga de controlar también los servicios personales y técnicos, la economía de almacenes y la protección al consumidor. Entre sus prioridades se encuentra la atención a la canasta básica, el consumo social priorizado, el saneamiento de las finanzas internas con calidad y honestidad, así como el almacenamiento, conservación, y rotación de las reservas estatales(MINCIN, 2020).

Una de las direcciones que forman parte de este ministerio es la dirección de Protección al consumidor. La cualse divide en dos áreas: Atención a la población y Protección al consumidor. El área de Atención a la población controla las reclamaciones, quejas, denuncias y sugerencias presentadas por los consumidores ante proveedores de productos o servicios del sistema estatal o no estatal del comercio en Cuba. Cada una de las reclamaciones, quejas, denuncias o sugerencias que presentan los consumidores pueden estar relacionadas con las siguientes temáticas: (Garbey Rivera, 2019).

- Los derechos de los consumidores.
- La educación, información y divulgación en materia de consumo.

- La actuación ética y la responsabilidad social de los proveedores.
- La transparencia y la profesionalidad en su actuar.
- La prevención y mejora continua de los procedimientos.

Por otra parte, el área de Protección al consumidor gestiona planteamientos, y las respuestas dadasa la población.Losplanteamientos están relacionados con temas de la canasta básica u otras actividades de comercio vinculadas con la venta de mercancías, servicios técnicos y personales, así como gastronomía y alojamiento, independientemente de su subordinación o no al MINCIN. Cada planteamiento se origina en las reuniones de rendición de cuenta correspondientes a los períodos de mandato de las Asambleas Municipales del Poder Popular. Las respuestas gestionadas en esta área, son las respuestas dadas a las reclamaciones o quejas realizadas por personas naturales o jurídicas(Garbey Rivera, 2019).

En la actualidad el área de Atención a la población del MINCIN cuenta con una solución informática para la gestión de reclamaciones, quejas, denuncias y sugerencias de la población. Esta se encuentra desarrollada en*acces,* tecnología propietaria, perteneciente a la *suit*[1]de*Microsoft Office*, por lo que responde a la política de privacidad de la empresa *Microsoft,* y no contribuye ala soberanía tecnológica por la cual vela el país. La solución permite realizar búsquedas bajo diferentes criterios, pero aplicando solamente un criterio a la vez, lo cual provoca dificultad y pérdida de tiempo en el proceso. Por ejemplo: si se desean buscar las denuncias realizadas por una persona, teniendo únicamente los datosde la provincia de la persona y los seis primeros dígitos de su carnet de identidad, el flujo a seguir es el siguiente:primeramente, se debe realizar la búsqueda aplicando un filtro con la provincia, el resultado de esta búsqueda puede ser bastante extenso, luego se debe realizar una nueva búsqueda filtrando por los 6 primeros dígitos del carnet de identidad.

En el caso del área de Protección al consumidor los planteamientos se gestionan a través de una hoja electrónica de cálculo (excel). Por tanto, en temas de soberanía tecnológica tiene la misma limitante de la solución utilizada en el área de Atención a la población, teniendo en cuenta que el excel también pertenece a la *suit* de *Microsoft Office*. Otra de las características de la solución utilizada en el área de Protección al consumidor, que a la

[1] En informática es un conjunto de aplicaciones o programas desarrollados bajo un mismo ambiente de desarrollo con distintas funcionalidades entre ellas.

vez limita el trabajo en el área, es que para dar respuesta a los planteamientos de la población se debe consultar manualmente la base de datos en *acces* del área de Atención a la poblaciónpara obtener el identificador de los mismos y luego darle respuestas en el área de Protección al consumidor, respuestas que estaránasociadas a los identificadores en la solución implementada en excel, el cual marca la relación de la respuesta con la reclamación, queja, denuncia o sugerencia. Teniendo en cuenta que lo anterior no está integrado en un mismo sistema informático,el proceso se encuentra expuesto a errores humanos que causen alteración en la información.

A partir de la problemática antes descrita se define el siguiente **problema de investigación**:¿Cómo gestionar las informaciones relacionadas con las reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas procesadas en la dirección de Protección al consumidor en el MINCIN, logrando celeridad en el proceso y reduciendo la ocurrencia de errores?

Tomando en cuenta el problema antes propuesto se define como**objeto de estudio**:aplicaciones informáticas de gestión.

Determinándose como**objetivo general:**desarrollar una personalización del SIGAP en el MINCIN, que permita gestionar las informaciones relacionadas con las reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas procesadas en la dirección de Protección al consumidor de este ministerio, de forma tal que se logre celeridad en el procesoy reduciendo la ocurrencia de errores.

Desglosándose del objetivo general los**objetivos específicos:**

- Elaborar el marco teórico de la investigación mediante el estudio de referentes teóricos sobre el desarrollo de aplicaciones informáticas dirigidas a la gestión de información relacionada conreclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas.
- Realizarlaidentificación de requisitos, análisis, diseño e implementaciónde la solución propuesta.
- Validar el diseño y el funcionamiento de la solución propuesta, mediante la aplicación de métricas y pruebas de software.
- Verificar el cumplimiento de la relación causa efecto de la variable independiente sobre la variable dependiente de la investigación.

Para ello se tiene como**campo de acción:**la gestión de datos relacionados con reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas

Definiéndose como **idea a defender:**si se desarrolla una personalización del SIGAP en el MINCIN, que permita gestionar las informaciones relacionadas con las reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas procesadas en la dirección de Protección al consumidor de este ministerio, se logra celeridad en el proceso y se reduce la ocurrencia de errores.

Para la presente investigación se aplicaron los siguientes métodos científicos

Métodos Teóricos:

Histórico-Lógico:permitió realizar un estudio sobre la evolución de soluciones web destinadas al procesamiento de la información relacionada con la gestión de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas, en el ámbito nacional e internacional.

Analítico-Sintético:permitió la realización del diseño teórico de la investigación, extrayéndose los elementos más importantes relacionada con la gestión de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas, así como de las herramientas, tecnologías y buenas prácticas utilizadas en la propuesta de solución durante el análisis de la bibliografía estudiada.

Modelación:permitió la confección de prototipos funcionales, la representación de los requisitos y el diseño de la base de datos.

Métodos Empíricos:

Entrevista:se le realizó al cliente una entrevista estructurada con anterioridad con el propósitodecomprender el proceso de negocio, recolectar datos como la estructura y funcionamiento de la Dirección de protección al consumidor en el MINCIN, así como las deficiencias existentes que permitieron definir el problema a resolver y establecer el objeto de estudio. En el Anexo 1, se encuentra la guía de preguntas utilizada en el desarrollo de la entrevista.

El presente documento consta de tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. Los capítulos han sido definidos de la siguiente forma:

Capítulo 1. Fundamentación teórica:en este capítulo se describen los conceptos relacionados con el objeto de estudio de la presente investigación, así como las características fundamentales de los sistemas para la gestión de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas que existen en diversas regiones del mundo incluyendo Cuba. Además, se describe la metodología seleccionada para guiar el proceso de desarrollo de la presente investigación, junto a los lenguajes y herramientas utilizadas, así como los patrones de diseño, y las pruebas de software a aplicar.

Capítulo 2.Análisis, diseño e implementación de la solución propuesta:en este capítulo se realiza una descripción de las principales características de la personalización del SIGAP en el MINCIN. En el mismo se describe el diseño de la solución propuesta, a partir de las disciplinas de la metodología definida para guiar el desarrollo de esta. Entre los contenidos analizados se encuentra el proceso de captura de los requisitos funcionales (RF) y no funcionales (RnF) con los que debe cumplir el componente propuesto. Además, se describe la arquitectura de la solución, el diagrama de clases del diseño, el modelo de datos y los patrones de diseño utilizados. Por otra parte, se exponen los estándares de codificación empleados en la disciplina de implementación y se realiza una descripción de las principales interfaces de la solución obtenida.

Capítulo 3.Validación de la solución propuesta: en el capítulo se muestran los resultados obtenidos luego de aplicar las técnicas de validación de requisitos, así como las métricas para la validación del diseño de la solución propuesta. En cada caso se documentan los resultados obtenidos. Por otra parte, se aplican las pruebas de software organizadas a partir de las disciplinas establecidas por la metodología AUP variación UCI para esta etapa (Pruebas internas, Pruebas de liberación y Pruebas de Aceptación). En cada disciplina se realizan pruebas a nivel de unidad, integración y sistema, con la aplicación de sus respectivos métodos y técnicas. El capítulo concluye con la verificación del cumplimiento del objetivo general de la investigación a partir de la definición de un conjunto de indicadores que permiten evaluar a través de un antes y un después al desarrollo de lapersonalización, la relación causa-efecto de la variable independiente sobre las variables dependientes de la investigación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se describen conceptos relacionados con el objeto de estudio de la presente investigación, los sistemas homólogos para la gestión de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestasque existen en el ámbito nacional e internacional. Se describe la metodología seleccionada para guiar el proceso de desarrollo de la presente investigación, junto a los lenguajes y herramientas utilizadas, los patrones de diseño y las pruebas de software a aplicar.

1.2. Conceptos fundamentales asociados al dominio del problema

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

CEGEL: es un centro de desarrollo de software, perteneciente a la Facultad 3 de la UCI. Está orientado a la informatización de empresas cubanas, mediante el desarrollo de productos, servicios y soluciones integrales de alta confiabilidad, calidad, competitividad, fidelidad y eficiencia(UCI, 2019).

Ministerio del Comercio Interior (MINCIN):es un organismo de la administración central del estado cubano, encargado de proponer al Gobierno las políticas relacionadas con el comercio mayorista y minorista, la logística de almacenes y la protección al consumidor que son realizadas por actores de la economía estatales y no estatales(MINCIN, 2020).

Protección al consumidor:es un medio que las leyes nacionales e internacionales otorgan a quienes compran una mercadería (personas físicas o jurídicas) o utilizan un servicio, para que puedan reclamar si sienten que han sido engañados o perjudicados por los vendedores o prestadores del servicio (DeConceptos, 2014).

Reclamación: es una petición, una demanda que una persona plantea ante otra con el objetivo de resolver un problema concreto. Una reclamación también puede realizarse ante una institución específica o ante un comercio. Los clientes tienen derechos como consumidores de defender esos derechos ante el establecimiento(Definicion ABC, 2007).

Queja:es un reclamo, lamento o llamado de atención ante lo quedisgusta, apena o provoca malestar. Puede hacerse contra el sujeto o entidad que la ha provocado o contra quienes ejercen autoridad (Pérez Porto, y otros, 2010).

Denuncia:es la acción y efecto de denunciar (avisar, noticiar, declarar la irregularidad o ilegalidad de algo, delatar). La denuncia puede realizarse ante las autoridades correspondientes (lo que implica la puesta en marcha de un mecanismo judicial) o de forma pública (sólo con valor testimonial) (Pérez Porto, y otros, 2010).

Sugerencia:es algo que se propone, insinúa o sugiere. El término suele emplearse como equivalente a consejo o recomendación con el objetivo de tener mejoría en cuanto a lo sugerido (Pérez Porto, y otros, 2010).

Planteamiento: es la acción de proponer, exponer o suscitar un tema, una duda o un problema; poner en ejecución una reforma o un sistema; o enfocar la solución de un problema, aunque no se llegue a obtenerla(Pérez Porto, y otros, 2010).

Celeridad:se trata de un término que hace referencia a la velocidad, la premura, la rapidez o la prisa(Pérez Porto, y otros, 2010).

En el caso de la presente investigación, el término celeridad se utiliza como variable dependiente con el propósito de indicar el breve espacio de tiempo en el cual debe procesarse la información.

1.3. Sistemas homólogos

En búsqueda de una solución para la gestión de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas, se realizó un estudio de sistemas homólogos utilizados en diferentes esferas a nivel nacional e internacional.

En el ámbito internacional, se estudiaron las siguientes soluciones:

ISOTools:esuna solución desarrollada el año 1998, como un proyecto europeo de inteligencia Artificial.En el año 2002 nace la plataforma para sistemas de gestión, la cual incorpora sistema de gestión de quejas y reclamaciones. En el año 2006 se convierte en el primer cliente de América. Desde el año 2011 hasta el 2013, fue expandiendo sus oficinas a Chile, Perú, México y Colombia. Es utilizado en más de 20 países(IsoTools, 2019).

El software ISOTools permite a las organizaciones establecer un sistema de gestión de quejas y reclamaciones, ya sea a través de la plataforma web o móvil. La solución permitepersonalizar los campos que se desean incluir en las encuestas de quejas y reclamaciones. Por otra parte, con esta herramienta se logra automatizar de principio a fin todo el proceso a través de flujos de trabajo o procedimientos. Esto permite una gestión rápida y eficiente de los procesos, posibilitando que los clientes realicen sus trámites en un menor plazo de tiempo (IsoTools, 2019).

Givertex: es una herramienta que permite gestionar las incidencias y las reclamaciones de los clientes. Está desarrollada sobre la plataforma *Microsoft Office SharePointServer*, diseñada para automatizar el control y la gestión de incidencias y reclamaciones de una empresa. Con esta herramienta se podrá disponer de un control exhaustivo de cada incidencia y reclamación, evitando así la pérdida de información relevante para la correcta gestión de su negocio. La aplicación se caracteriza por realizar el flujo de trabajo de forma sencilla y automatizada. Permitegenerar informes estadísticos que mejoren los procesos internos de la empresa u organización y brindar mayor calidad en losservicios al cliente (Givertex, 2019).

En el ámbito nacionalse estudiaron el Sistema de Gestión para la Atención a la Población (SIGAP)y el Portal delBanco Metropolitano. A continuación, se describenlos mismos.

SIGAP:esta solución se caracteriza por ser un sistema genérico, capaz de gestionar el flujo básico para la atención a la población en una organización o empresa. A través del mismo se puede procesar información relacionada con quejas, reclamaciones, denuncias y sugerencias. El sistema brinda la posibilidad de crear un expediente único para cada promovente que presente un escrito de queja y a partir de este se desencadena todo el flujo de gestión del proceso de atención a la población.

Portal del Banco Metropolitano:el Banco Metropolitano S.A. cuenta con un portal web, el cual permite la gestión de quejas y reclamaciones, relacionadas con los servicios que el banco brinda. El procesose desarrolla a través de un formulario de quejas y reclamaciones presente en el módulo de atención a la población. Por medio de este formulario los usuarios pueden ingresar sus quejas o reclamaciones, para luego ser revisadas por los especialistas del banco(WebSite4Cuba, 2016).

Para llevar a cabo una comparación entre las soluciones informáticas estudiadas y las necesidades de la dirección de Protección al consumidordel MINCIN, los autores de la

presente investigación definieron un conjunto de características que debe cumplir la solución propuesta. A continuación, se muestra una tabla en la cual a través de las filas se lista las características definidas y en las columnas los softwares estudiados.

Tabla 1 Comparación de sistemas homólogos

Característica \ Software	ISOTools	Givertex	Portal del Banco Metropolitano	SIGAP
Gestión de reclamaciones	Sí	Sí	Sí	Sí
Gestión de quejas	Sí	No	Sí	Sí
Gestión de denuncias	No	No	No	Sí
Gestión de sugerencias	No	No	No	Sí
Solicitar y disponer prórroga	No	No	No	No
Gestión de planteamientos	No	No	No	No
Gestión de respuestas	No	No	No	No
Tecnología libre	No	No	No	Sí

Fuente: elaboración propia

Teniendo en cuenta el estudio de cada uno de los sistemas anteriormente descritos se demuestra que ninguno presenta la totalidad de las características que requiere el MINCIN en una solución para la dirección de Protección al consumidor. Pues la mayoría de estos sistemas solo se centran en registrar reclamaciones, sin llegar a clasificar las mismas y darle seguimiento a todo el proceso de gestión. Por otra parte, prácticamente la totalidad de estos están desarrollados en tecnologías propietarias. El SIGAP es la solución que más se aproxima al sistema deseado por el MINCIN, sin embargo, tiene como limitante que no es capaz de gestionar planteamientos, ni brindar respuestas a estos.

Todo lo anterior demuestra la necesidad de desarrollarla personalización propuesta,con el propósito de obtener una solución que independiza al país de las soluciones propietarias y contribuya a la política de soberanía tecnológica por la cual Cuba apuesta. El estudio de soluciones homólogas también ha permitido a los autores de la presente investigación

obtener conocimientos en cuanto al proceso de gestión de quejas y reclamaciones a través de software nacionales y foráneos empleados en diferentes esferas. Cada uno de estos conocimientos se tuvieron en cuenta en el desarrollo de la solución propuesta.

1.4. Metodología de desarrollo de software

Una metodología de desarrollo de software consiste en múltiples herramientas, modelos y métodos para asistir en el proceso de desarrollo de software. En ella se define con precisión los artefactos, roles y actividades involucradas, junto con prácticas y técnicas recomendadas, las guías de adaptación de la metodología al proyecto y las guías para el uso de herramientas de apoyo(Association of Modern Technologies Professionals, 2019).

En la presente investigación se decide utilizar como metodología de desarrollo de software, una variación del Proceso Unificado Ágil (AUP, por sus siglas en inglés *Agile Unified Process*) definida para los proyectos de la UCI, teniendo en cuenta que es la más acorde con las características del negocio a desarrollar, así como con las habilidades y experiencias del equipo de desarrollo. Además, la solución propuesta responde a uno de los proyectos de la Red de Centros de Desarrollo de Software de la UCI, la cual tiene establecido el uso de esta metodología en todos los proyectos de la Universidad. A continuación, se describen los principales elementos de esta metodología.

1.4.1. Metodología AUP-UCI

Dentro de las metodologías ágiles se encuentra el Proceso Unificado Ágil (*AUP*, por sus siglas en inglés*Agile Unified Process*), la cual consiste en una versión simplificada de la metodología de desarrollo tradicional Proceso Racional Unificado (RUP). *AUP* describe la forma de desarrollar aplicaciones de software de manera fácil de entender, usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Con el objetivo de estandarizar el proceso de desarrollo de software, la dirección de producción de la UCI tiene definida como metodología a utilizar, una variación de *AUP* en unión con el modelo *CMMI-DEV v1.3*[2], denominada AUP-UCI. La misma establece prácticas centradas en el desarrollo de productos y servicios de calidad (Sánchez, 2015).

La metodología *AUP* propone organizar el proceso de desarrollo de software en cuatro fases (Inicio, Elaboración, Construcción, Transición). En el caso de la adaptación de esta

[2]Modelo de Madurez de Capacidades Integrado para Desarrollo (CMMI-DEV) proporciona buenas prácticas para el desarrollo y mantenimiento de productos y servicios.

metodología para los proyectos de la UCI se decide mantener la fase de inicio, pero modificando su objetivo, las tres restantes fases se unifican, quedando una sola denominada ejecución y se agrega una fase de cierre (Sánchez, 2015). A continuación, se describen cada una de las fases de la variación *AUP* para la UCI(Sánchez, 2015):

- **Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación de este. En esta fase se realiza un estudio inicial de la organización cliente, obteniéndose información acerca del alcance del proyecto, estimaciones de tiempo, esfuerzo y costo. Todo este estudio permite decidir si se ejecuta o no el proyecto.
- **Ejecución:** en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y el diseño. Además, se implementa el software y se le aplican pruebas. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Asimismo, en la transición se capacita a los usuarios finales sobre la utilización del software.
- **Cierre:** en esta fase se analizan los resultados del proyecto relacionados con su ejecución y se realizan las actividades formales de cierre del proyecto.

Disciplinas de variación de AUP-UCI

AUP propone siete disciplinas: Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno. Para el ciclo de vida de los proyectos de la UCI se decide utilizar igual número de disciplinas. Los flujos de trabajos: modelado de negocio, requisitos y análisis y diseño en *AUP* están unidos en la disciplina modelo. En la variación *AUP* para la UCI estos flujos de trabajo se consideran disciplinas independientes, además se mantiene la disciplina implementación. En el caso de las pruebas, estas se desagregan en tres disciplinas: pruebas internas, pruebas de liberación y pruebas de aceptación. Las disciplinas de *AUP* asociadas a la gestión de proyecto, en la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2 (gestión de la configuración, planeación de proyecto y monitoreo y control) (Sánchez, 2015)**.**

Escenarios para la disciplina Requisitos

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Caso de Uso del Negocio (CUN), Diagrama de Proceso del Negocio (DPN) y Modelo Conceptual (MC)). Existen tres formas de encapsular los requisitos (Caso de Uso del Sistema (CUS), Historias de Usuarios (HU), Diagrama de Requisitos por Proceso (DRP)), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

- **Escenario No 1:** proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS. Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización.
- **Escenario No 2:** proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS. Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no sea necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelan exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.
- **Escenario No 3:** proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP. Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.
- **Escenario No 4:** proyectos que no modelen negocio solo pueden modelar el sistema con historias de usuario (HU). Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y

validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información (Sánchez, 2015).

La aplicación de la variación de *AUP* en la UCI permite estandarizar el proceso de desarrollo de software en la universidad, dando cumplimiento a las buenas prácticas que define *CMMI-DEV v1.3*.

A partir del análisis realizado sobre la variación de la metodología *AUP* para la UCI, los autores dela presente investigación deciden organizar el proceso de desarrollo dela personalización a partir de las fases y disciplinas que propone esta variación de la metodología. La decisión se adopta teniendo en cuenta que el desarrollo delosmódulos forma parte al proyecto “Personalización del XABAL SIGAP 1.0 para el MINCIN”, el cual responde la red de centros de la universidad. Por tanto, los autores de la investigación son integrantes de este equipo de desarrollo. El proyecto está planificado a través del cronograma de proyectos versión 5.0 definido en la UCI para proyectos de desarrollos, clasificación sobre la cual se tipifica la presente solución. Este cronograma está estructurado a partir de las fases y disciplinas definidas por AUP-UCI.

El escenario a utilizar es el No.4, ya que aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. Estas características se adaptan a la solución propuesta.

1.5. Herramientas y Tecnologías

Los autores de la presente investigación deciden utilizar en el desarrollo de la solución propuesta las herramientas y tecnologías empleadas en el proyecto de desarrollo de software al cual responde esta tesis, con sus mismas versiones. Esta decisión es adoptada con el propósito de lograr compatibilidad de la solución con otros componentes ya desarrollados en el proyecto. Así como, permitir que la curva de aprendizaje de los autores sea baja, al estar vinculados con personal de experiencia en el uso de estas herramientas y tecnologías,que les pueda aclarar rápidamente cualquier duda o error que surja en el proceso. Además, el uso de un entorno de desarrollo ya probado agiliza los tiempos fijados en cronograma. En los próximos subepígrafes se describen cada una de las herramientas y tecnologías utilizadas en el desarrollo de la tesis.

1.5.1. Herramientas de Ingeniería del Software Asistidas por Computadoras

Las herramientas *CASE* (por sus siglas en inglés *Computer Aided Software Engineering)* están destinadas a aumentar la productividad en el desarrollo del software reduciendo el

coste del mismo en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costos, implementación de parte del código a partir del diseño dado, compilación automática y documentación o detección de errores (Visual-Paradigm, 2019). A continuación, se describen las herramientas*CASE* utilizadas en las tareas ingenieriles de la presente investigación.

Visual Paradigm for UML v8.0 Enterprise Edition

Es una herramienta queutiliza lenguaje unificado de modelado (*UML*, por sus siglas en inglés) quesoportael ciclo devidacompletodel desarrollo desoftware:análisisydiseño orientadosa objetos,construcción,pruebasydespliegue. Permitemodelar todoslos tiposde diagramas de clases, generar código desde diagramas y generar documentación. La herramienta agiliza la construcción de aplicaciones con calidad y a un menor coste de tiempo. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como obtener ingeniería inversa de bases de datos (Visual-Paradigm, 2019).

Balsamiq Mockups v3.5.17

Balsamiq Wireframes es una aplicación de creación de estructuras alámbricas de sitios web de interfaz gráfica de usuario. Permite al diseñador organizar *widgets* prediseñados utilizando un editor *WYSIWYG* de arrastrar y soltar. La aplicación se ofrece en una versión de escritorio, así como en un complemento para *Google Drive*, *Confluence* y JIRA(Balsamiq Studios, 2016). En el caso de la presente investigación ha sido utilizado en la creación de los prototipos de interfaz de usuario de la solución propuesta.

1.5.2. Marcos de trabajo

Desde el punto de vista del desarrollo de software, un marco de trabajo es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado (Alegsa, 2019).

Para el desarrollo de la solución propuesta se hace necesario utilizar los marcos de trabajo *Angular* (como marco de trabajo para el lado del cliente) y *SpringBoot*(como marco de trabajo para el lado del servidor) en sus versiones 5.0 y 1.5 respectivamente,teniendo en cuenta que el SIGAPestá implementado sobre estas tecnologías.

Angular v.5.0

Es un marco de trabajo de *JavaScript* de código abierto desarrollado por Google**.** Contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. Además, contiene todas las herramientas que los creadores ofrecen para que los desarrolladores sean capaces de crear un *HTML*(Lenguaje de Marcados para Hipertextos, por sus siglas en inglés *HyperText Markup Language)* enriquecido (Google, 2019).

Spring Boot v.1.5

Spring Boot es una infraestructura ligera que elimina la mayor parte del trabajo de configurar las aplicaciones basadas en *Spring*, el mismo facilita la creación de proyectos con *frameworkSpring* eliminando la necesidad de crear largos archivos de configuración *XML* (Lenguaje de Marcado Extensible, por sussiglas en inglés*eXtensible Markup Language*). Además, provee configuraciones por defecto para Spring y otras librerías. También provee un modelo de programación parecido a las aplicaciones *java* tradicionales que se inician en el método principal "*main*" (Pivotal, 2019).

1.5.3. Lenguajes de programación

De los lenguajes de programación que existen para desarrollar aplicaciones orientadas a la web se encuentran dos grupos fundamentales, la programación del lado del servidor y la programación del lado del cliente. Esta última incluye aquellos lenguajes que son interpretados por una aplicación cliente como el navegador web, entre ellos se encuentra *HTML***.** Los lenguajes de programación del lado servidor son reconocidos, ejecutados e interpretados por el propio servidor, el que se encarga de brindar información al cliente en un formato comprensible para él. A continuación, se describenlos lenguajes de programación utilizados en el desarrollo de la solución propuesta:

Java SE v.11

Java(desarrollado por *Sun Microsystems* y posteriormente adquirida por la compañía *Oracle*) es un lenguaje de programación de propósito general, concurrente, orientado a objetos y multiplataforma que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. El mismo permite desarrollar e implementar aplicaciones Java en equipos de escritorio y servidores. Proporciona una colección de clases para su uso en aplicaciones de red, que permite establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas (ORACLE, 2019).

JavaScript v.ES6

Es un lenguaje de programación ligero, interpretado por la mayoría de los navegadores y que les proporciona a las páginas web, efectos y funciones complementarias a las consideradas como estándar *HTML*. Este tipo de lenguaje de programación es de código abierto, por lo que cualquier persona puede utilizarlo sin comprar una licencia. Con frecuencia son empleados en los sitios web, para realizar acciones en el lado del cliente, estando centrado en el código fuente de la página web (Mozilla, 2019).

TypeScript v.2.9

Es un lenguaje de programación de código abierto desarrollado por Microsoft, el cual cuenta con herramientas de programación orientada a objetos, muy favorable si se tienen proyectos grandes. *TypeScript* convierte su código en *javascript*. Es llamado también *superset*[3] de *javascript*, lo que significa que, si el navegador está basado en *javascript*, este nunca llegará a comprender que el código original fue realizado con *typescript* y ejecutará el *javascript* como lenguaje original (Typescript, 2019).

HTML v.5.0

*HTML*es el lenguaje que se utiliza para crear las páginas web de internet. Es reconocido universalmente y permite publicar información de forma global. Define una estructura básica y un código *HTML* para la definición de contenido de una página web, como texto, imágenes, entre otros y se basa en la referenciación por hipertextos o enlaces entre páginas (Mozilla, 2019).

CSS v.3.0

CCS (Hojas de Estilo en Cascada, por sus siglas en inglés *Cascading Style Sheets)* es un lenguaje para controlar la presentación de los documentos electrónicos definidos con *HTML* y *XHTML* (Lenguaje de Marcado para Hipertextos Extensible, por sus siglas en inglés *eXtensible HyperText Markup Language)*. *CSS* es la mejor forma de separar los contenidos de una presentación y es imprescindible para la creación de páginas web complejas.Mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en dispositivos diferentes(Mozilla, 2019).

[3] Se trata de un lenguaje escrito sobre otro lenguaje.

Se utiliza para definir el aspecto de todos los contenidos, como: el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. Funciona a base de reglas para las declaraciones sobre el estilo de uno o más elementos (Mozilla, 2019)

1.5.4. Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado, (IDE por sus siglas en inglés*Integrated Development Environment*), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los *IDE* proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (Rouse, 2019).
Para el desarrollo de la solución propuesta, se utilizan como *IDE* las herramientas *IntelliJ IDEA*en su versión 2018.3.5 para la comunidad y *WebStorm* en su versión 2018.3.5 para la comunidad, teniendo en cuenta que ambas son libres, de código abiertoy se integran perfectamente alos lenguajes de programación a utilizar.

IntelliJ IDEA v.2018.3.5

Es un *IDE* desarrollado por *JetBrains*[4] y está disponible en dos ediciones: una edición para la comunidad (código abierto) y una edición comercial (de pago). Es soportado por varios sistemas operativos: *Windows*, *Linux* o *MacOS*. Su versión gratuita soporta lenguajes como *Java*, *Groovy*, *XML/XSL*, *Kotlin* y*Python*, *Clojure*, *Dart*, *Erlang*, *Go*, *Haxe*, *Perl*, *Scala*, *Haskell*, *Lua*, estos últimos vía *plugin*. La versión de pago además de soportar todos los lenguajes antes mencionados soporta *JavaScript*, *HTML*, *CSS*, *SQL*, *Ruby*, así como *PHP* vía *plugin*(JetBrains, 2019).

WebStorm v.2018.3.5

Es un entorno de desarrollo inteligente, igualmente desarrollado por *JetBrains,* entiende el proyecto y ayuda a producir código de alta calidad de manera más eficiente, gracias al completamiento de código, se detectan errores sobre la marcha, la navegación y refactorizaciones automatizadas. Da soporte a las recientes tecnologías, funcionando de

[4] Empresa informática de nacionalidad checa, desarrolladora de IDEs y productos de software en general.

la mejor manera con las más modernas y populares para el desarrollo web, así como *AngularJS*, *ECMAScript 6* y *Compass*. Es multiplataforma, funciona en *Windows*, *MacOS* o *Linux* con una única clave de licencia. *WebStorm* agiliza el flujo de trabajo mediante la integración con todo lo necesario para el desarrollo productivo. Además, desde el *IDE* se pueden utilizar varias herramientas como el depurador y terminales (JetBrains, 2019).

1.5.5. Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es un conjunto de programas no visibles que administran y gestionan la información que contiene una base de datos. A través de él se maneja todo acceso a la base de datos con el objetivo de servir de interfaz entre ésta, el usuario y las aplicaciones (Rouse, 2019). Para el desarrollo de la solución propuesta se utilizael sistema gestor de bases de datos PostgreSQL en su versión 9.6.

PostgreSQL v.9.6

Es un potente sistema de base de datos objeto-relacional de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada con una alta fiabilidad e integridad de datos. Se ejecuta en los principales sistemas operativos que existen en la actualidad como *Linux*, *UNIX* (*AIX*, *BSD*, *HP-UX*, *SGIIRIX*, *MacOSX*, *Solaris*, *Tru64*) y *Windows*(The PostgreSQL Global Development Group, 2019).

1.6. Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características, una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Rojas, 2010).

Objetivos de estos patrones:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas de software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.

- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimientos ya existentes.

Así mismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.
- No es obligatorio utilizar los patrones siempre, sólo en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error (Rouse, 2019).

Se pueden agrupar en dos grandes grupos; los GRASP (*GeneralResponsibilityAssignmentSoftwarePatterns*, en inglés), que son patrones generales de software para asignación de responsabilidades y los GOF (*GangofFour*, en inglés), encargados de la inicialización, agrupación y comunicación de los objetos. En el Capítulo 2 de la presente investigación, se describen los patrones de diseño empleados en el desarrollo dela soluciónpropuesta.

1.7. Pruebas de software

Las pruebas de software son un conjunto de actividades que pueden ser planificadas con antelación y ejecutarse sistemáticamente durante la implementación o al finalizar el desarrollo del software. Estas se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación. Las pruebas de software se ejecutan a partir de la aplicación de métodos y técnicas. La organización del proceso de pruebas se realiza a través de cuatro niveles: unidad, integración, sistema y aceptación (Pressman, 2010).A continuación, se describen las disciplinas definidas en la metodología AUP-UCI para la etapa de pruebas de software y la relación de estas con los niveles de prueba.

Pruebas internas

En la disciplina pruebas internas se verifica a nivel de equipo de desarrollo el resultado de la implementación. Para la ejecución de estas pruebas se deben desarrollar artefactos de apoyo, tales como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar el proceso (Sánchez, 2015).

Para la validación de la solución propuesta las pruebas internas se ejecutan a nivel de unidad, integración y sistema. En el capítulo 3 del presente documento se detalla cómo se realizó el desarrollo de las mismas.

Pruebas a nivel de unidad

Las pruebas a nivel de unidad se centran en el esfuerzo de verificación de la unidad más pequeña del diseño: el componente o módulo de software. Tomando como guía la descripción del diseño a nivel de componente, se prueban importantes caminos de control para describir errores dentro de los límites del módulo. El alcance restringido que se ha determinado para las pruebas de unidad limita la relativa complejidad de las pruebas y los errores que éstas descubren (Pressman, 2010).

En el epígrafe 3.4 del presente documento se explican los métodos de pruebas utilizados a nivel de unidad.

Pruebas en los niveles de integración y de sistema

Las pruebas a nivel de integración tienen como objetivo identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente. Se diseñan para descubrir errores o completitud en las especificaciones de las interfaces (Pressman, 2010).

Las pruebas a nivel de sistema tienen como objetivo verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las operaciones apropiadas funcionando como un todo. Es similar a la prueba de integración, pero con un alcance más amplio(Pressman, 2010).

En el capítulo 3 del presente documento se pueden encontrar las técnicas, métodos y resultados obtenidos al aplicar este nivel de prueba a la solución propuesta.

Pruebas de liberación

Las pruebas de liberación son diseñadas y ejecutadas por una entidad certificadora de la calidad externa al equipo de desarrollo. Estas se realizan a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (Sánchez, 2015).

En el capítulo 3 del presente documento se describe cómo se ejecutaronlas pruebas de liberación sobre la solución propuesta.

Pruebas de aceptación

Las pruebas de aceptación se realizan para establecer el grado de confianza en un sistema,partes del mismo o en sus características no funcionales.La confianza en el sistema está determinada por su grado de adherencia a las necesidades, requerimientos y procesos de negocio solicitados por el usuario o cliente. Es en función a estos que el usuario debe decidir si acepta o no el sistema que le está siendo entregado.Por lo tanto, las pruebas de aceptación suelen ser responsabilidad de los clientes o usuarios del sistema. Otros interesados del proyecto pueden involucrarse también.

En el capítulo 3 se describe cómo se ejecutaron las pruebas de aceptación sobre la solución propuesta.

1.8. Conclusiones parciales

- El desarrollo del marco teórico referencial relacionado con los términos descritos en el epígrafe 1.2, facilita una mejor comprensión del objeto de estudio de la presente investigación.
- El estudio de soluciones nacionales y extranjeras destinadas a la gestión de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas, demostró la necesidad de desarrollar la solución propuesta.
- El análisis de las características de la variación de la metodología *AUP* para la UCI, permitió definir las disciplinas a utilizar en la organización del desarrollo de la solución propuesta, en correspondencia con las características de esta.
- El estudio de los patrones de diseño posibilitó identificar cómo y cuándo utilizarlos en el desarrollo de la solución propuesta.
- El estudio de las pruebas de software permitió definir la estrategia de prueba a aplicar en la validación de la solución propuesta.

CAPÍTULO 2.ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓNPROPUESTA

2.1. Introducción

En este capítulo se realiza una descripción de los requisitos implementados en la personalización del SIGAP para el MINCIN. En el mismo se describe el diseño de la solución propuesta, a partir de las fases de la metodología definida para guiar el desarrollo de esta. Entre los contenidos analizados se encuentran el proceso de captura y validación de los requisitos funcionales (RF) y no funcionales (RnF). Además, se describe la arquitectura de la solución, el diagrama de clases del diseño, el modelo de datos y los patrones de diseño utilizados. Por otra parte, se exponen los estándares de codificación empleados en la disciplina de implementación y se realiza una descripción de las principales interfaces de la solución obtenida.

2.2. Descripción del negocio

El MINCIN presenta la dirección de Protección al consumidor, la cual se divide en dos áreas: Atención a la población y Protección al consumidor. El área de Atención a la población es la encargada de gestionar cada uno de los escritos de queja presentados por los consumidores ante proveedores de productos o servicios del sistema estatal o no estatal. Cuando un escrito llega al MINCIN vía telefónica, correo postal, correo electrónico o de manos de la persona que lo promueve, este es recepcionado y asignado a un especialista, quien se encarga de clasificarlo en reclamación, queja, denuncia o sugerencias. Luego cada escrito es desglosado en asuntos, los cuales posteriormente son dispuestos en función de la temática que aborden. La acción de disponer permite definir quién es el responsable de tramitar cada asunto, si el propio MINICIN u otro ministerio u organismo. Una vez que todos los asuntos son tramitados, se procede a gestionar las respuestas de cada proceso de tramitación en caso que las lleven. Cada respuesta tiene un tiempo determinado para ser gestionada, en los casos que se necesite más tiempo para esta gestión, se procede a la solicitud de una prórroga. Cuando todos los asuntos de un escrito son procesados y las respuestas son registradas, entonces se puede proceder a cerrar el escrito.

Por otra parte, el área de Protección al consumidor es la responsable de gestionar las soluciones a los escritos procesados desde el área de Atención a la población; de igual forma se gestionan los planteamientos originados en las reuniones de rendición de cuenta correspondientes a cada período de mandato de las asambleas municipales del poder popular. Cada planteamiento está relacionado con temas de la canasta básica u otras actividades de comercio vinculadas con la venta de mercancías, servicios técnicos y personales, así como gastronomía y alojamiento, independientemente de su subordinación o no al MINCIN. La gestión de las soluciones se realiza obteniéndose las causas y condiciones, la acción tomada, así como la conformidad del promovente con la respuesta al escrito y el tipo de solución que finalmente se adopta. Por otra parte, la gestión de los planteamientos se realiza a partir del análisis de las intervenciones de la población en las asambleas de rendición de cuentas. De esta forma se tiene un control de los principales datos de los planteamientos, tales como: tipo de planteamiento, proceso y mandato al que corresponde la asamblea en la cual se origina el planteamiento, texto del planteamiento y la respuesta que recibe.

2.3. Disciplina de requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir y comprende la administración de los requisitos funcionales y no funcionales del producto (Sánchez, 2015). Teniendo en cuenta que la personalizaciónresponde al cumplimiento del objetivo general de la presente investigación, tiene un negocio bien definido y fácil de comprender, sin necesidad de realizar un modelado de este, se decide, según la metodología seleccionada, utilizar en la disciplina de requisitos el escenario número cuatro, historias de usuario (HU). En este epígrafe se presentan las técnicas para la identificación de los requisitos, la especificación de los mismos, así como las historias de usuario obtenidas.

2.3.1. Técnicas para la identificación de requisitos

Para identificar las necesidades del cliente se utilizan técnicas que permiten determinar y documentar los requisitos para el desarrollo de la solución. Esta actividad es continua durante el ciclo de desarrollo y combina, en diferentes puntos, diversas técnicas de identificación para obtener la visión más completa de las necesidades del usuario final (Pressman, 2010). Para la captura de los requisitos dela personalización propuesta se utilizan las siguientes técnicas:

- **Entrevista:** es de gran utilidad para obtener información cualitativa, requiere seleccionar bien a los entrevistados para obtener la mayor cantidad de información en el menor tiempo posible. Es muy aceptada y permite acercarse al problema de una manera natural (Sommerville, 2011). Esta técnica fue aplicada a los trabajadores delMINCIN. En el Anexo 1, se encuentra la guía de preguntas utilizada en el desarrollo de la entrevista.
- **Tormenta de ideas:** es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios (Pressman, 2010). Esta técnica se evidencia durante las reuniones con el cliente donde, luego de un diálogo entre ambas partes, se obtuvo como resultado un conjunto de acuerdos con el fin de refinar las necesidades del cliente. En el Anexo 2 del presente documento se relacionan los acuerdos tomados durante la tormenta de ideas.

2.3.2. Especificación de requisitos

Los requisitos constituyen un punto clave en el desarrollo de aplicaciones informáticas. Un gran número de proyectos de software fracasan debido a una mala definición, especificación o administración de requisitos. Factores tales como requisitos incompletos o mal manejo de los cambios de los requisitos, llevan a proyectos completos al fracaso total. Los requisitos se enfocan en las especificaciones de lo que se desea desarrollar y tienen dos clasificaciones: requisitos funcionales y no funcionales. Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particularidades y de cómo se debe comportar en distintas situaciones y los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema (Sommerville, 2011).

En la presente investigación, una vez realizado el levantamiento de información e identificadas las necesidades del cliente, se identificaron un total de 44 requisitos funcionales (RF). A continuación, en la tabla 2 se presentan cada uno de estos y sus respectivas descripciones.

Tabla 2 Requisitos funcionales

N°	Nombre	Descripción
RF1.	Registrar expediente	Permite registrar un expediente en el sistema, con los datos introducidos del mismo.
RF2.	Modificar expediente	Permite modificar un expediente registrado en el sistema

RF3.	Visualizar expediente	Permite visualizar la información de un expediente registrado en el sistema.
RF4.	Buscar expediente	Permite realizar una búsqueda de los expedientes atendiendo a ocho criterios de búsqueda. Como resultado se muestra un listado de los expedientes según coincidencia con los criterios de búsqueda introducidos en caso de introducir alguno.
RF5.	Cerrar expediente	Permite cerrar el expediente seleccionado. Se le pregunta al usuario que confirme está acción antes de realizarse. Cuando se cierra el expediente el estado debe actualizarse a Cerrado.
RF6.	Registrar promovente	Permite registrar un promovente en el sistema, llenando los campos que muestra la interfaz.
RF7.	Modificar promovente	Permite modificar los datos de un promovente en el sistema.
RF8.	Visualizar promovente	Permite visualizar la información de un promovente registrado en el sistema.
RF9.	Buscar promovente	Permite realizar una búsqueda de los promoventes atendiendo a nueve criterios de búsqueda. Como resultado se muestra un listado de los promoventes según coincidencia con los criterios de búsqueda introducidos en caso de introducir alguno.
RF10.	Asociar promovente	Permite asociar un promovente a un expediente determinado.
RF11.	Registrar escrito	Permite registrar un escrito en el sistema.
RF12.	Modificar escrito	Permite modificar la información de un escrito registrado en el sistema.
RF13.	Visualizar escrito	Permite visualizar la información de un escrito registrado en el sistema.
RF14.	Buscar escrito	Permite realizar una búsqueda de los escritos atendiendo a ocho criterios de búsqueda. Como resultado se muestra un listado de los escritos según coincidencia con los criterios de búsqueda

		introducidos en caso de introducir alguno.
RF15.	Clasificar escrito	Permite clasificar un escrito en los diferentes asuntos que puede contener.
RF16.	Cerrar escrito	Permite cerrar un escrito determinado.
RF17.	Registrar asunto	Permite registrar un asunto de un escrito registrado en el sistema.
RF18.	Modificar asunto	Permite modificar un asunto de un escrito registrado en el sistema.
RF19.	Visualizar asunto	Permite visualizar la información de un asunto registrado en el sistema.
RF20.	Buscar asunto	Permite realizar una búsqueda de los asuntos atendiendo a siete criterios de búsqueda. Como resultado se muestra un listado de los asuntos según coincidencia con los criterios de búsqueda introducidos en caso de introducir alguno.
RF21.	Disponer asunto	Permite disponer sobre un asunto seleccionado. Para ello el sistema muestra la información del asunto y permite seleccionar el tipo de disposición, así como la información necesaria asociada cada tipo de disposición.
RF22.	Modificar disposición de asunto	Permite modificar una disposición registrada en el sistema.
RF23.	Visualizar disposición de asunto	Permite visualizar la información de unadisposición registrada en el sistema.
RF24.	Responder disposición de asunto	Permite registrar otras observaciones de la disposición, además de una breve síntesis de la respuesta.
RF25.	Registrar nomenclador	Permite adicionar un nomenclador en el sistema.
RF26.	Modificar nomenclador	Permite modificar los datos de un nomenclador registrado en el sistema.
RF27.	Buscar nomenclador	Permite realizar una búsqueda de los nomencladores atendiendo a dos criterios de

		búsqueda. Como resultado se muestra un listado de los nomencladores según coincidencia con los criterios de búsqueda introducidos en caso de introducir alguno.
RF28.	Desactivar nomenclador	Permite desactivar el nomenclador seleccionado.
RF29.	Modificar informe de respuesta al promovente	Permite modificar los datos del informe de respuesta al promovente registrado en el sistema.
RF30.	Visualizar informe de respuesta al promovente	Permite visualizar la información del informe de respuesta al promovente registrado en el sistema.
RF31.	Solicitud de prórroga	Permite solicitar prórroga al escrito seleccionado.
RF32.	Crear informe de respuesta al promovente	Permite crear un informe de respuesta al promovente en el sistema.
RF33.	Buscar informe de respuesta al promovente	Permite realizar una búsqueda de los informes de respuesta al promovente atendiendo a cinco criterios de búsqueda. Como resultado se muestra un listado de los informes de respuesta al promovente según coincidencia con los criterios de búsqueda introducidos en caso de introducir alguno.
RF34.	Reasignar escritos	Permite reasignar un escrito seleccionado.
RF35.	Buscar expediente de protección al consumidor	Permite realizar una búsqueda de los expedientes atendiendo a cinco criterios de búsqueda. Como resultado se muestra un listado de los expedientes según coincidencia con los criterios de búsqueda introducidos en caso de introducir alguno.
RF36.	Registrar solución al problema	Permite registrar la solución al problema en el sistema.
RF37.	Registrar planteamiento	Permite registrar un planteamiento en el sistema.
RF38.	Modificar planteamiento	Permite modificar un planteamiento en el sistema.
RF39.	Visualizar planteamiento	Permite visualizar un planteamiento en el sistema.
RF40.	Buscar planteamiento	Permite realizar una búsqueda de los planteamientos atendiendo a nueve criterios de búsqueda. Como resultado se muestra un listado de

		los planteamientos de según coincidencia con los criterios de búsqueda introducidos en caso de introducir alguno.
RF41.	Registrar anónimo	Permite registrar un expediente en el sistema sin promovente (Anónimo).
RF42.	Registrar escrito con preservación de identidad	Permite registrar un escrito en el sistema con promovente que solicita preservación de su identidad.
RF43.	Disponer sobre prórroga	Permite disponer sobre la prórroga solicitada.
RF44.	Dictaminar sobre escrito	Permite dictaminar el escrito en el sistema.

Fuente: elaboración propia

En el caso de los requisitos no funcionales (RnF) se definieron un total de nueve, los cuales son clasificados y descritos a continuación:

Usabilidad

RnF1. La interfaz del sistema a desarrollar debe ser sencilla.
RnF2. El sistema debe mostrar facilidad para interactuar y entender las actividades que realiza el usuario.
RnF3. La cantidad de clic para acceder a una funcionalidad debe ser como máximo cinco.

Portabilidad

RnF4. El sistema debe ser multiplataforma, principalmente debe funcionar sobre las plataformas Windows y Linux.

Confiabilidad

RnF5. El sistema debe contar con campos obligatorios para garantizar un manejo adecuado de la información introducida por el usuario.
RnF6. El sistema no debe permitir la entrada de datos incorrectos y debe notificar los errores.

Mantenibilidad

RnF7. Todo el código del sistema debe estar comentado haciéndolo fácil de analizar.

Eficiencia

RnF 8. El sistema en condiciones de carga máxima, cuando se ejecuten tareas que realizan consultas a las bases de datos y mecanismos de comunicación entre módulos debe ser capaz de ofrecer tiempos de repuestas mínimos o iguales a cinco segundos.

Restricciones del diseño y la implementación

RnF 9. Para la implementación del sistema se utilizan las mismas herramientas y tecnologías empleadas en el desarrollo del proyecto al cual tributa la solución propuesta.

2.3.3. Historias de usuario

Las Historias de Usuario (HU) son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes, por lo que una historia de usuario puede tener varios cambios a lo largo de un desarrollo sin afectarse el tiempo (Rouse, 2019).
Para el desarrollo delapersonalización propuesta se definieron un total de 44 HU, una por cada RF. A continuación, se describe la HU correspondiente al RF37Registrar planteamiento, teniendo en cuenta que este es uno de los RF bases para el desarrollo delapersonalización y de mayor prioridad para el cliente. El resto de las HU se pueden consultar en los objetos entregables de la presente investigación.

Tabla 3HU Registrar planteamiento

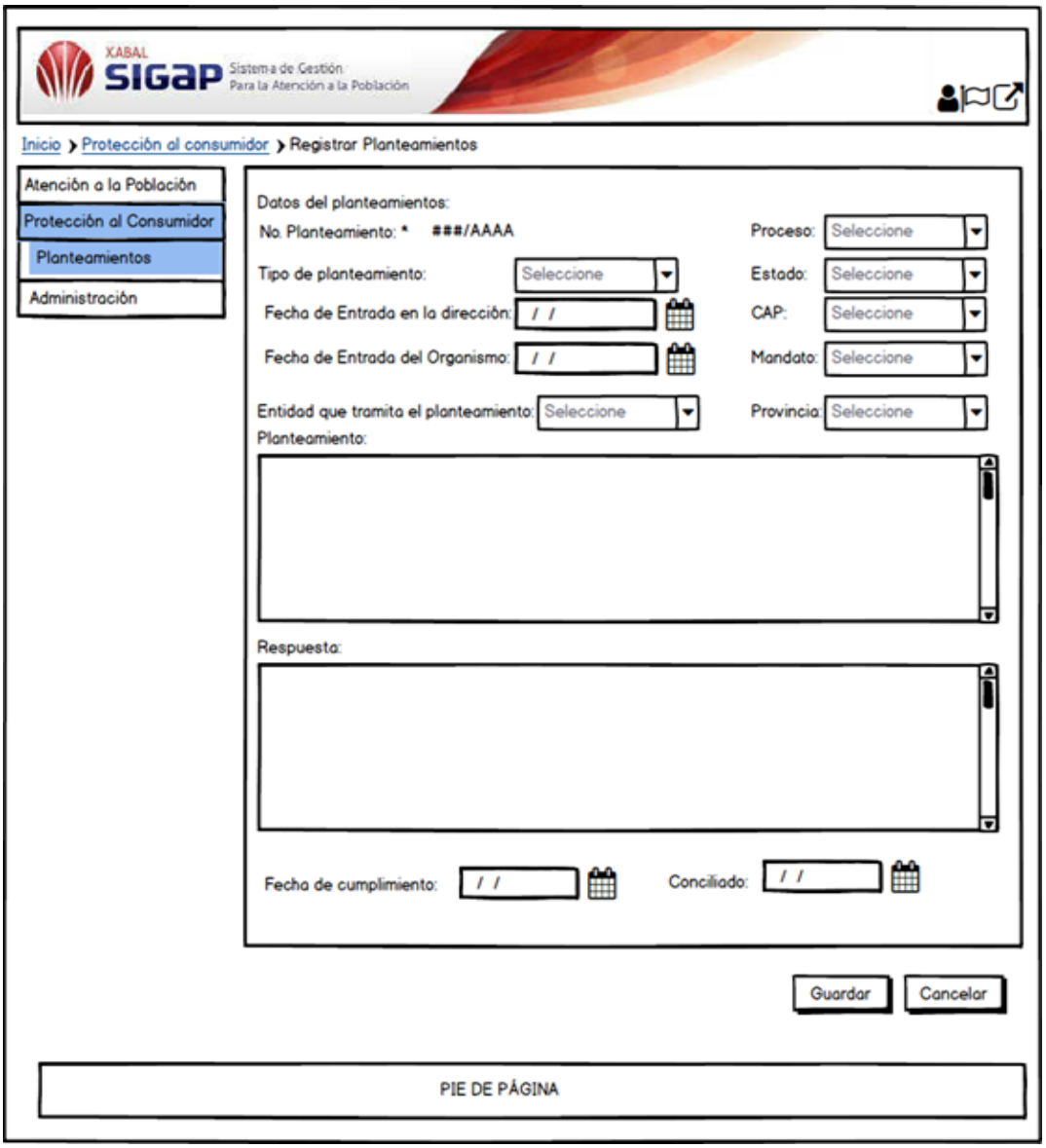
Historia de usuario	
Número:37	**Nombre del requisito:** Registrar planteamiento
Programadores: Leonel y Ramsés	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:**40h
Riesgo en Desarrollo:	**Tiempo Real:** 40h
Descripción:permite al usuario registrar un planteamiento en el sistema, llenando los campos que muestra la interfaz.Todos los campos son obligatorios.A continuación, se describen cada uno de estos campos: • Tipo de planteamiento: campo de tipo nomenclador • Proceso: campo de tipo nomenclador	

- Estado: campo de tipo nomenclador
- CAP: campo de tipo nomenclador
- Fecha de Entrada en la dirección: campo de tipo calendario
- Fecha de Entrada del Organismo: campo de tipo calendario
- Mandato: campo de tipo nomenclador
- Entidad que tramita el Planteamiento: campo de tipo nomenclador
- Provincia: campo de tipo nomenclador
- Planteamiento: campo de tipo texto, el cual permite todo tipo de caracteres
- Respuesta: campo de tipo texto, el cual permite todo tipo de caracteres
- Fecha de cumplimiento: campo de tipo calendario
- Conciliado: campo de tipo calendario

Una vez introducidos los datos el usuario selecciona la opción *Guardar* para guardar los datos, y el sistema muestra un mensaje de éxito al crear planteamiento.

La opción Cancelar permiteal usuario cerrar la interfaz: Registrar planteamiento.

Prototipo de interfaz:



Fuente: elaboración propia

2.4. Disciplina de análisis y diseño

En esta disciplina los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa, así como una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo la arquitectura) (Sánchez, 2015). Además, en esta disciplina se modela el sistema para que soporte todos los requisitos, incluyendo los requisitos no funcionales.

2.4.1. Diseño arquitectónico

El diseño arquitectónico garantiza cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global del mismo. En el modeladode proceso de desarrollo de software, el diseño arquitectónico es la primera etapa en el proceso de diseño del software. Es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. La salida del proceso de diseño arquitectónico consiste en un modelo que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación (Sommerville, 2011).

La solución propuesta tiene como base la arquitectura del SIGAP. En la figura 1 se representan los elementos de esta arquitectura y las partes donde incide lapersonalización propuesta. Es importante especificar que el SIGAP utiliza una arquitectura separada en *frontend* y *backend*[5]. El *frontend* es la parte del software que interactúa con los usuarios y el *backend* es la parte que procesa la entrada desde el *frontend*. La idea general es que el *frontend* sea el responsable de recolectar los datos de entrada del usuario, transformándolos y ajustándolos a las especificaciones que demanda el *backend* para ser procesados, devolviendo generalmente una respuesta que el *frontend* recibe y expone al usuario de una forma entendible para este.

Para la solución propuesta la conexión entre *frontend* y *backend*se realiza a través del protocolo *HTTP* (Protocolo de transferencia de hipertexto, por sus siglas en inglés *Hypertext Transfer Protocol*) intercambiando datos en formato *JSON* (Notación de objetos *JavaScript*, por sus siglas en inglés *JavaScript ObjectNotation*). En la figura 1 se muestra cómo está compuesta la arquitectura de la solución propuesta

[5]Frontend es la parte de un sitio web que interactúa con los usuarios, es la que está del lado del cliente. Backend es la parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web, es la que corre del lado del servidor.

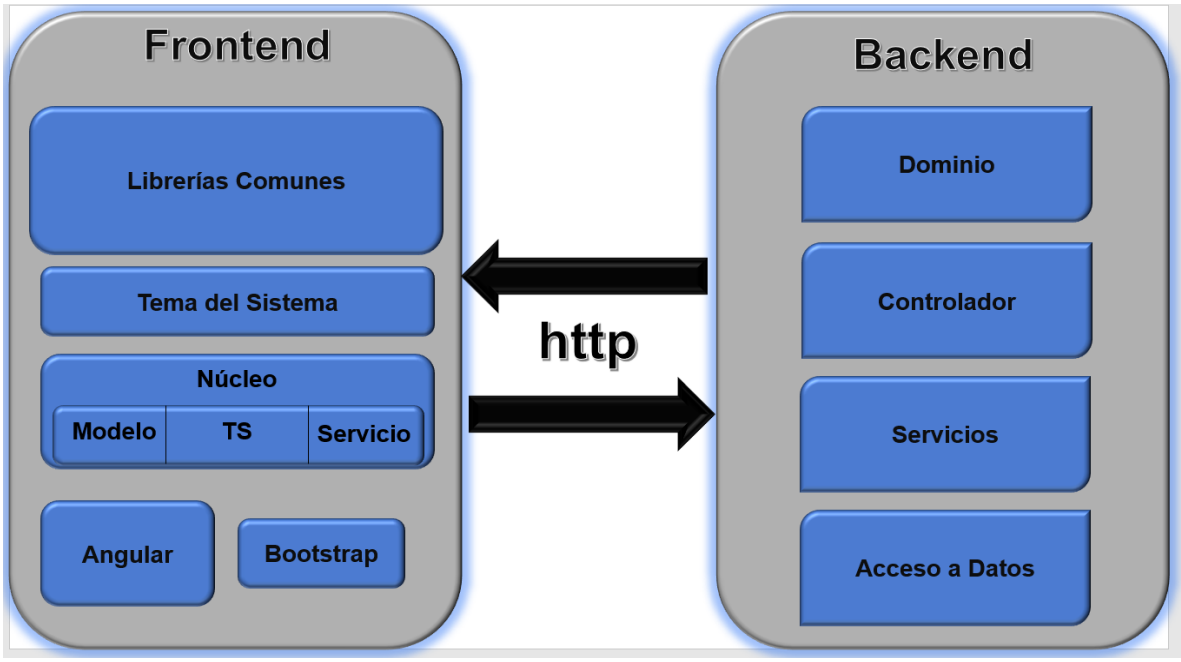


Figura 1 Arquitectura del SIGAP
Fuente: elaboración propia

2.4.2. Patrones de diseño

Para diseñar la solución propuesta se emplearon un conjunto de patrones de diseño, los cuales son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. A continuación, se describen los patrones empleados:

- **Patrones GRASP**[6]

Los Patrones Generales de Software para la Asignación de Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2016). En el caso de la presente investigación se aplicaron los siguientes patrones GRASP:

Experto: el patrón experto en información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Este permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para llevar a cabo las tareas (Larman, 2016).

[6]General Responsibility Assignment Software Patterns (Patrones de software de asignación de responsabilidad general)

En la presente investigación el uso de este patrón se evidencia en las clases entidades. Estas clases son las expertas en información y las encargadas de manejar la lógica del negocio. En la siguiente figura se muestra un fragmento del código de la clase *Planteamiento.java*, la cual es la experta en información para el tratamiento de los planteamientos.

```java
public class Planteamiento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @NotNull
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "dplanteamiento_seq")
    @SequenceGenerator(name = "dplanteamiento_seq", sequenceName = "dplanteamiento_id_seq", allocationSize = 1)
    @Basic(optional = false)
    @Setter

    @Column(name = "id")
    protected Long id;
    @Setter

    @Column(name = "no_planteamiento")
    private String noPlanteamiento;

    @Column(name = "fecha_entrada_direccion")
    @Convert(converter = FechaConverter.class)
    private LocalDate fechaEntradaDireccion;
```

Figura 2 Aplicación del patrón Experto en la clase *Planteamiento.java*
Fuente: elaboración propia

Creador: este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas desarrollados a partir de un paradigma orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización (Larman, 2016).

En la presente investigación el uso de este patrón se evidencia en la clase *PlanteamientoServiceImpl.java.* El patrón se aplica en el momento de crear nuevas instancias de la clase Planteamiento, tal y como ilustra en el área señalada en rojo en la figura 3.

```java
@Override
public Planteamiento CrearPlanteamiento(PlanteamientoDTO planteamientoDTO){

    Planteamiento planteamiento = planteamientoFactory.CrearPlanteamiento(planteamientoDTO);
    planteamiento.setNoPlanteamiento(generarConsecutivo());
    return planteamientoRepository.save(planteamiento);
}
```

Figura3 Aplicación del patrón Creador en la clase *PlanteamientoServiceImpl.java*
Fuente: elaboración propia

Controlador: el objetivo de este patrón es asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Este patrón delega la responsabilidad en otras clases con las que mantiene un modelo de alta cohesión (Larman, 2016).

En la presente investigación el uso del patrón se evidencia en las clases controladoras presentes en la arquitectura del *backend*. En la figura 4 se muestra la clase *PlanteamientoController.java*.

```
/**
 * Registrar un planteamiento
 *
 * @param planteamientoDTO
 * @return
 */
@RequestMapping(path = ENTITY_URI, method = RequestMethod.POST, consumes = "application/json")
public ResponseEntity<PlanteamientoResource> registrarPlanteamiento(@RequestBody PlanteamientoDTO planteamientoDTO) {

    try {
        Planteamiento planteamiento = planteamientoService.CrearPlanteamiento(planteamientoDTO);

        return ResponseEntity.created(new URI(ENTITY_API + ENTITY_URI + "/" + planteamiento.getId()))
                .headers(HeaderUtil.createEntityCreationAlert(ENTITY_NAME, planteamiento.getId().toString()))
                .body(planteamientoResourceAssembler.toResource(planteamiento));

    } catch (Exception ex) {
        return ResponseEntity.badRequest()
                .headers(HeaderUtil.badRequestAlert("El " + ENTITY_NAME + " no se puede registrar: " + ex.getMessage())).build();
    }
}
```

Figura 4 Aplicación del patrón Controlador en clases del *backend, método registrar un planteamiento.*
Fuente: elaboración propia

Bajo acoplamiento:este permite potenciar la reutilización y disminuye la dependencia entre las clases. Su principal característica es mantener las clases más independientes entre sí y con la menor cantidad de relaciones; la cual posibilita que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases (Larman, 2016).

En la presente investigación el uso de este patrón se evidencia en el 67 % de las clases del diseño, las cuales tienen una baja dependencia una de otras, tal y como se demuestra en el Capítulo 3 con los resultados de la validación del diseño a partir de la aplicación de métricas.

Alta cohesión: en la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una

alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (Larman, 2016). La principal función de la aplicación de este patrón es asignar responsabilidades de modo que la cohesión siga siendo alta. La información que almacena una clase debe de ser coherente y debe estar en la medida de lo posible relacionada con la clase.

En la presente investigación el uso de este patrón se evidencia en el 59 % de las clases del diseño, las cuales tienen una baja sobrecarga de responsabilidades, tal y como se demuestra en el Capítulo 3 con los resultados de la validación del diseño a partir de la aplicación de métricas.

- **Patrones GoF**[7]

Patrones Banda de los Cuatro, por sus siglas en inglés de *Gang of Four (*GoF*),* describen soluciones simples y eficaces a problemas específicos en el diseño de software orientado a objetos (Larman, 2016). En el caso de la presente investigación de aplicaron los siguientes patrones GoF:

Patrón Decorador: permite añadir responsabilidades a objetos concretos de manera dinámica y transparente sin afectar a otros objetos. Este patrón brinda más flexibilidad que la herencia estática y evita que las clases más altas en la jerarquía estén demasiado cargadas de funcionalidad y sean complejas (Larman, 2016).

En la presente investigación el uso de este patrón se evidencia en la clase *planteamiento-add.component.ts.* El patrón se aplica en el momento de hacer uso del decorador *@Component*, el cual es típico de las clases *Component* en Angular. En el área señalada en rojo en la figura 5 se especifica el uso de este patrón.

```
@Component({
  selector: 'app-planteamiento-add',
  templateUrl: './planteamiento-add.component.html',
  styleUrls: ['./planteamiento-add.component.scss']
})
```

Figura 5 Aplicación del patrón decorador en la clase *planteamiento-add.component.ts*
Fuente: elaboración propia

[7]Gang Of Four (Pandilla de cuatro)

Patrón Observador: permite observar los cambios producidos por un objeto, de esta forma, cada cambio que afecte el estado del objeto observado lanza una notificación a los observadores; a esto se le conoce como Publicador-Suscriptor. Es uno de los principales patrones de diseño utilizados en interfaces gráficas de usuario (GUI), ya que permite desacoplar al componente gráfico de la acción a realizar (Google, 2019).

En la presente investigación el uso de este patrón se evidencia en varias partes de la implementación de los componentes en el *frontend.* Porejemplo, al enviar los datos de la vista al controlador, este último hace la petición de un servicio a través de una inyección y se queda a la espera de la respuesta que este servicio debe proveer para así emitir o no un evento. En la figura 6 se especifica un ejemplo de la aplicación de este patrón en la clase *planteamiento.service.ts.*

```
obtenerplanteamientos(): Observable<any> {
  return this.apiService.use('API_RESOURCE_URL').post(this.urlResource + '/list');
}
```

Figura 6 Aplicación del patrón observador en la clase *planteamiento.service.ts*
Fuente: elaboración propia

- **Otros patrones**

Inyección de dependencias (ID): este es un patrón de diseño orientado a objetos, que permite el paso de objetos como dependencias, los cuales a su vez pueden pasar a componentes y estar disponibles en toda la aplicación. El propósito de este patrón es contener la lógica del negocio, clases para acceso a datos o utilidades de infraestructura.

Angular tiene su propio patrón ID, que se utiliza normalmente en el diseño de aplicaciones angulares para aumentar su eficiencia y modularidad. Las dependencias son servicios u objetos que una clase necesita para realizar su función. ID es un patrón de codificación en el que una clase solicita dependencias de fuentes externas en lugar de crearlas. En Angular, el patrón ID proporciona dependencias declaradas a una clase cuando se crea una instancia de esa clase (Google, 2019).

Teniendo en cuenta que la solución propuesta está desarrollada utilizando este marco de trabajo, el patrón ID se aplica en correspondencia con las adecuaciones para Angular. En la figura 7y 8se ilustran aplicaciones de este patrón.

```
@Injectable()
export class PlanteamientoResolver implements Resolve<Planteamiento> {

  constructor(private planteamientoService: PlanteamientoService) {
  }
```

Figura 7 Aplicación de *@Injectable*
Fuente: elaboración propia

```
@Component
public class PlanteamientoServiceImpl implements PlanteamientoService {

    @Autowired
    private PlanteamientoRepository planteamientoRepository;
    @Autowired
    private PlanteamientoFactory planteamientoFactory;
    @Autowired
    private ConsecutivoPService consecutivoPService;
```

Figura 8 Aplicación de *@Autowired*
Fuente: elaboración propia

Repositorio (*Repository*): el patrón repositorio está estrechamente relacionado con el acceso a datos y permite tener una abstracción de la implementación del acceso a datos en cualquier aplicación, de modo que la lógica de negocio no conozca ni esté acoplada a la fuente de datos. En pocas palabras, el repositorio actúa como un intermediario entre la lógica del negocio y la lógica de acceso a datos para que se centralice en un solo punto, y de esta forma se logre evitar redundancia de código. Al ser este una abstracción del acceso a datos permite desacoplar y testear de una forma más sencilla el código, ya que al estar desacoplado se pueden generar pruebas unitarias con mayor facilidad (Janssen, 2019).

Un ejemplo de la aplicación de este patrón en la propuesta de solución se logra cuando se obtiene un planteamiento del repositorio como se muestra en la siguiente figura.

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;

public interface PlanteamientoRepository extends JpaRepository<Planteamiento, Long>, JpaSpecificationExecutor<Planteamiento> {


    Planteamiento getById(Long aLong);


}
```

Figura 9 Aplicación del patrón Repository. clase *PlanteamientoRepository.java*

Fuente: elaboración propia

Como se puede observar en la figura 9, la lógica de negocio del cliente no interactúa directamente con la fuente de datos, ya que con este patrón la lógica de negocio no tiene conocimiento de dónde provienen ni cómo se obtuvieron los datos. Además, se puede apreciar que se comunica directamente con el repositorio, el cual se encarga de hacer los mapeos necesarios y de comunicarse con la fuente de datos.

2.4.3. Diagrama de clases

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y las interfaces que participan en el sistema con sus relaciones estructurales y de herencia. Los diagramas de este tipo contienen las definiciones de las entidades del software en vez de conceptos del mundo real y son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea una vista lógica de la información que se maneja en el sistema, los componentes que se encargarán del funcionamiento y la relación entre uno y otro (Larman, 2016).

A continuación, en la figura 10 se muestra una parte del diagrama de clases del diseño, en el cual se encuentran las clases correspondientes a la HU Registrar planteamiento, teniendo en cuenta que la misma responde al requisito anteriormente seleccionado en todo el documento. El diagrama de clases del diseño con la totalidad de las clases se encuentra entre los artefactos entregables de la tesis.

En el diagrama de clases del diseño se evidencia cómo la vista interactúa con la controladora, esta a su vez manipula al modelo y se nutre de un servicio ejecutado medianteuna petición realizada al *backend* a través del protocolo HTTP. Esta petición llega al controlador *PlanteamientoController.java*.

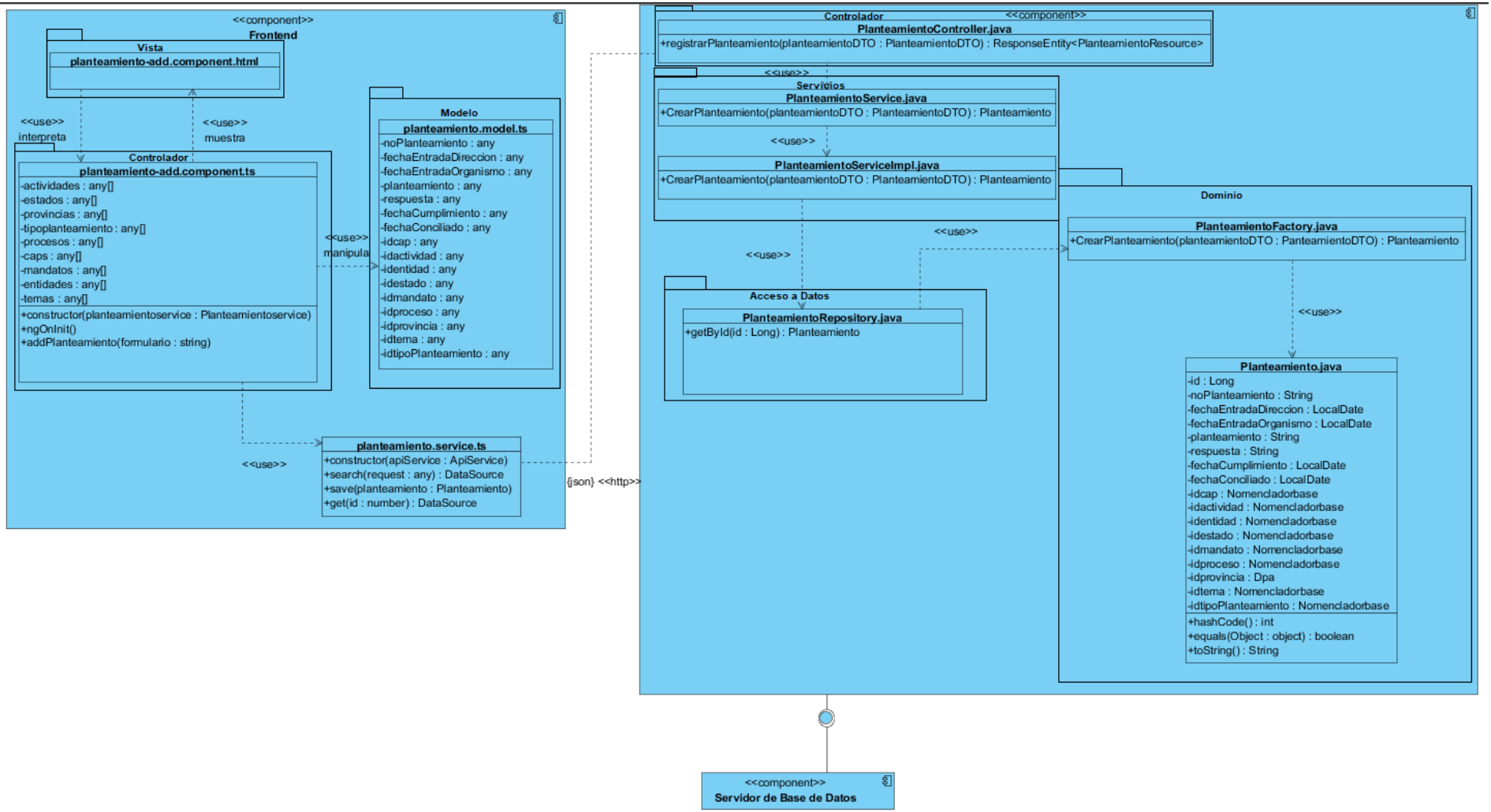


Figura 10 Diagrama de clases de diseño para la HU registrar planteamiento.
Fuente: elaboración propia

2.5. Disciplina de implementación

En esta disciplina a partir de los resultados del análisis y el diseño se construye la solución informática que da lugar a la presente investigación, para la cual se implementaron cada uno de los RF que permitieron obtener la personalización del SIGAP para el MINCIN.

2.5.1. Diagramas de componentes

Un diagrama de componentes es un diagrama tipo del Lenguaje Unificado de Modelado. Este representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema(Fakhroutdinov, 2015).

En la figura 11se muestra,a través de un diagrama de componentes, una descripción más detallada de cómo se representa en el *frontend* del SIGAP laarquitectura de la solución propuesta, tomando como ejemplo el RF37 Registrar planteamiento, a partir del uso del patrón arquitectónico MVC. En la figura el modelo está compuesto por el archivo *planteamiento.model.ts*, el cual contiene la información que se le solicita al usuario para crear el planteamiento, ejemplo: tipo de planteamiento y mandato. Por otra parte, la vista se compone por el archivo *planteamiento-add.component.html*. Este representa la información visible al usuario e interactúa con funciones específicas del controlador. El controlador está compuesto por el archivo *planteamiento-add.component.ts*, encargado de manipular el modelo y mostrar la información a la vista.

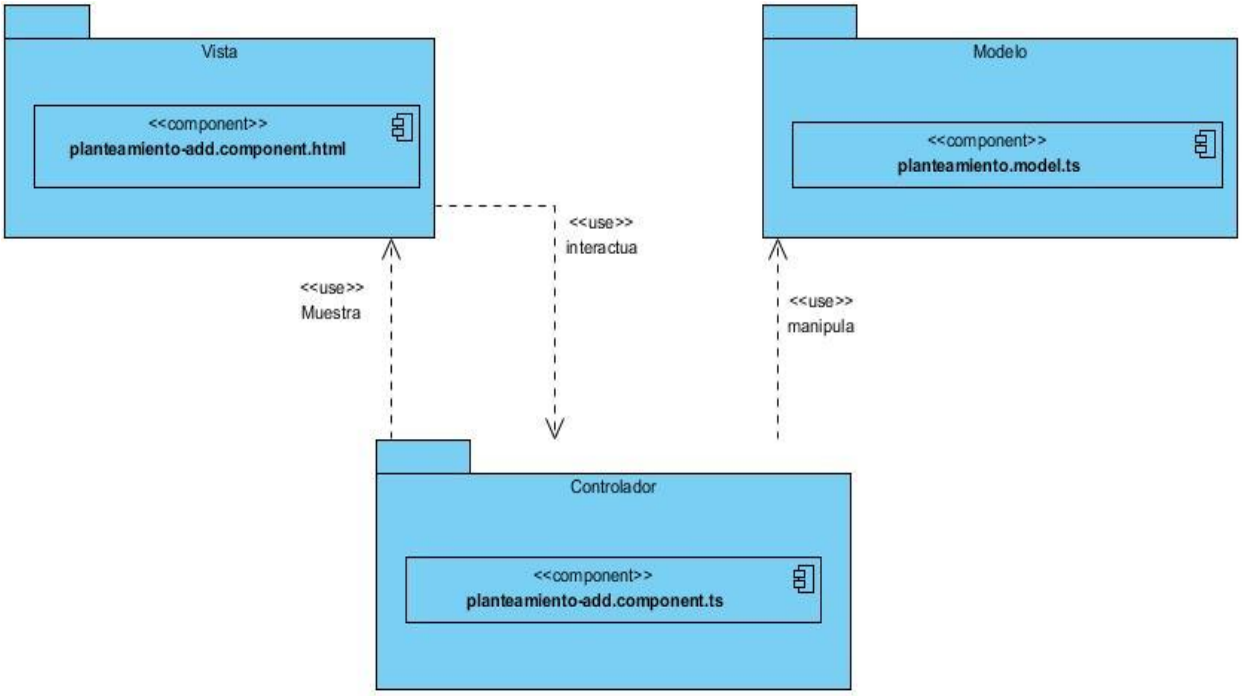


Figura 11 Diagrama de componentes en la arquitectura *frontend* del RF37.
Fuente: elaboración propia

En la figura 12 se muestra el diagrama de componentes que describe cómo se representa en el*backend* del SIGAP la arquitectura de lasolución propuesta de manera genérica, a partirdel uso del patrón arquitectónico N Capas.

En la figura el componente*PlanteamientoController.java* de la capa Controlador usa al componente *PlanteamientoService.java* de la capa Servicios, el cual usa al componente de la capa de Acceso a Datos *PlanteamientoRepository.java*y los tres a la vez están orientados a la capa Dominio, la cual es el núcleo de la aplicación. La misma y su lógica de negocio definen el comportamiento y las restricciones de la aplicación, en cuanto a la forma en que se van a comunicar las demás capas con esta. Lo anterior hace que una aplicación sea diferente de otras en cuanto a la función que realiza.

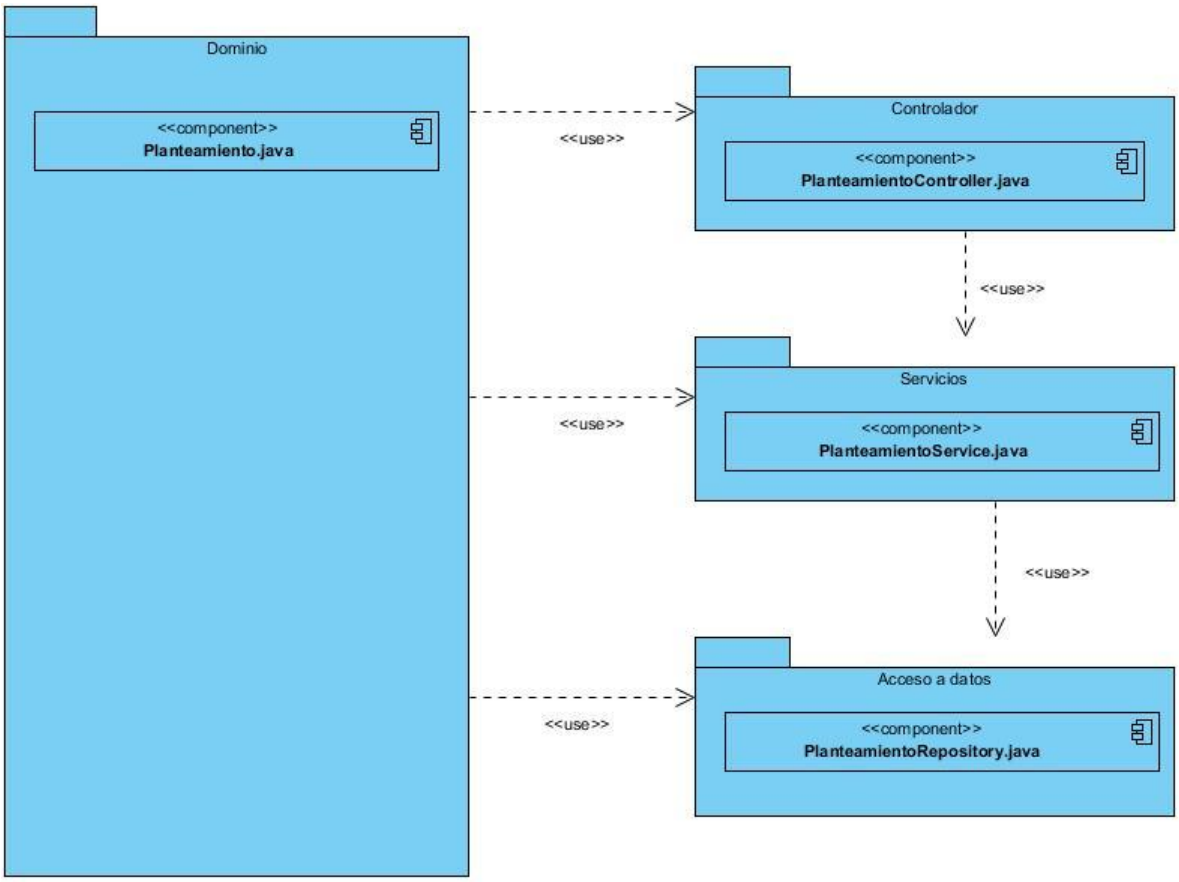


Figura 12 Diagrama de componentes en la arquitectura *backend* del RF37.
Fuente: elaboración propia

2.5.2. Descripción del sistema

En la figura 13 se muestra la vista de inicio de la solución, desde la cual se acceden a los diferentes módulos de gestión. Por otra parte, la figura 14 muestra la vista de la funcionalidad que da solución al RF37Registrar planteamiento.

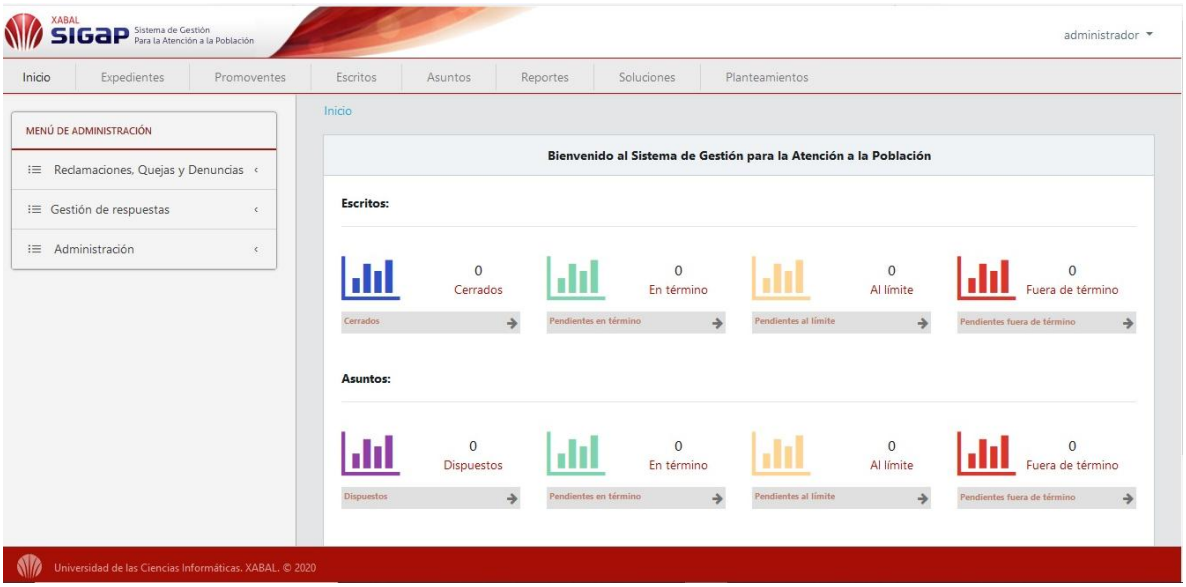


Figura13 Vista inicial de la solución.
Fuente: elaboración propia.

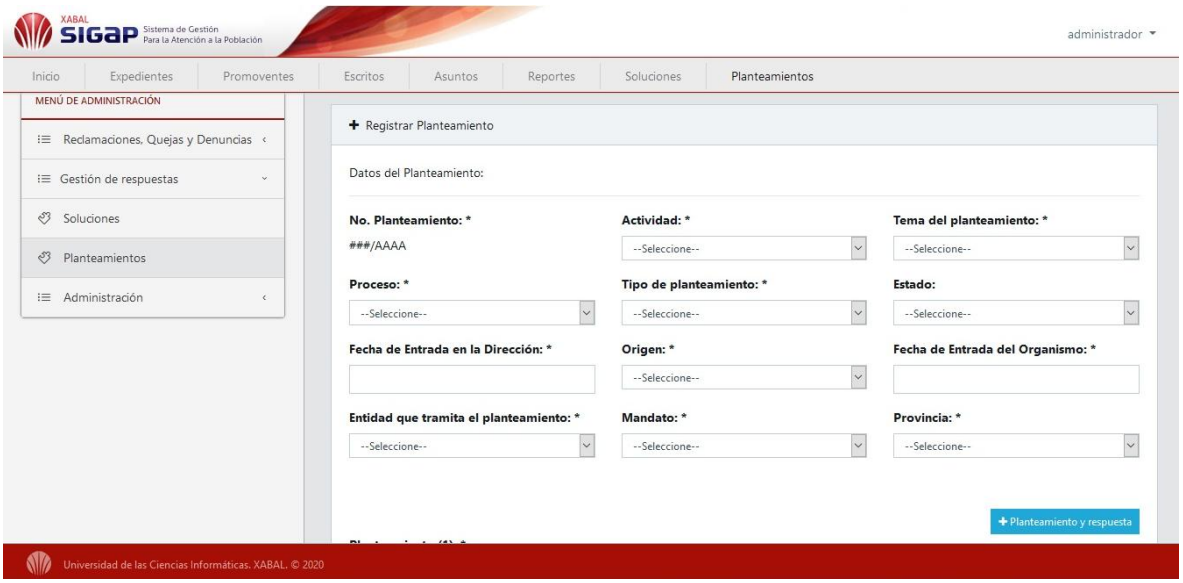


Figura 14 Vista para registrar un planteamiento.
Fuente: elaboración propia

En la figura 15 se muestra una imagen de la funcionalidadque responde al RF39 Visualizar planteamiento.

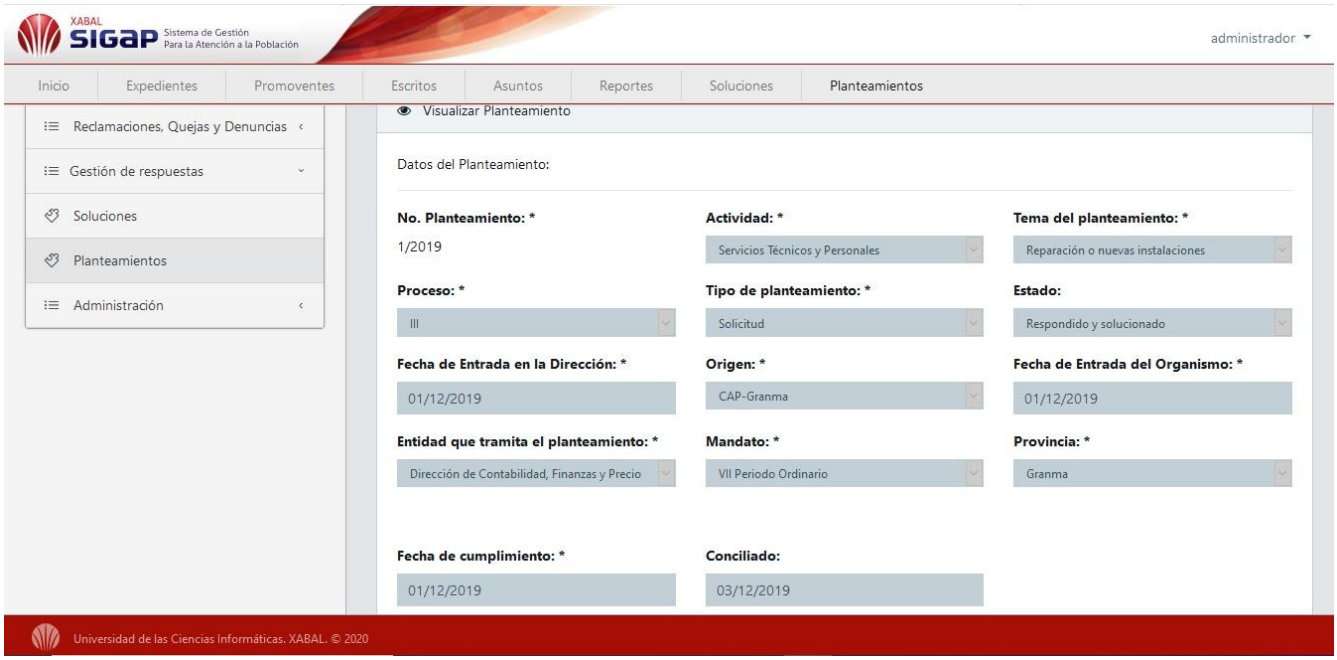


Figura 15 Visualización de un planteamiento.
Fuente: elaboración propia

2.6. Conclusiones parciales

- El empleo de la metodología *AUP* en su variación para la UCI en función de describir el proceso de desarrollo de la solución propuesta, permitió organizar el desarrollo de la misma y generar los artefactos necesarios.
- La aplicación de los patrones arquitectónicos MVC para la arquitectura del *frontend* y N-Capas para la arquitectura del *backend*, así como el uso de patrones de diseño, permitieron obtener una solución con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. Introducción

En el presente capítulo se muestran los resultados obtenidos luego de aplicar las técnicas de validación de requisitos y las métricas para la validación del diseño de la solución propuesta, documentándose los resultados obtenidosen cada caso. Por otra parte, se aplican las pruebas de software organizadas a partir de las disciplinas establecidas por la metodología *AUP* variación UCI para esta etapa (Pruebas internas, Pruebas de liberación y Pruebas de Aceptación). En el caso de las pruebas internas y las pruebas de liberación se decide aplicar los niveles de unidad, integración y sistema. Para las pruebas de aceptación se aplica el nivel de aceptación.El capítulo concluye con la verificación del cumplimiento del objetivo general de la investigación a partir de la definición de un conjunto de indicadores que permiten evaluar a través de un antes y un después al desarrollo de la personalización, la relación causa-efecto de la variable independiente sobre las variables dependientes de la investigación.

3.2. Validación de los requisitos

Con el objetivo de asegurar que la personalización a desarrollar se corresponde con las necesidades del cliente, cada uno de los requisitos identificados fueron validados antes de llegar a la disciplina de análisis y diseño. Para la validación de estos se utilizaron las siguientes técnicas:

- **Generación de casos de prueba:** los casos de pruebas que responden a los RF del componente propuesto fueron fáciles de diseñar, lo que significa que cada uno de los RF se interpretan de forma correcta y pueden ser comprendidos satisfactoriamente por los desarrolladores en la disciplina de implementación. En el caso de la presente investigación se generaron un total de 44 descripciones de casos de prueba,los cuales se encuentran como artefactos entregables de la tesis.

- **Revisiones formales de los requisitos:** se realizaron revisiones formales de cada requisito, por parte del cliente y el equipo de desarrollo. Estas revisiones demostraron que la interpretación de cada una de las descripciones no es ambigua, ni presenta omisiones o errores que hagan que la descripción del requisito no se corresponda con las necesidades del cliente.

- **Construcción de prototipos de interfaz de usuario:** esta técnica permite hacer simulaciones delapersonalización a implementar y brinda la posibilidad a los especialistas de tener una idea de

cómo serán las interfaces del sistema una vez desarrollado. En la tabla 2 del epígrafe 2.3.3 se muestra el prototipo no funcional correspondiente al RF 37.Registrar planteamiento.

3.3. Validación del diseño

Con el objetivo de comprobar la calidad del diseño se emplearon las métricas de diseño tamaño operacional de clases (TOC) y relaciones entre clases (RC).

3.3.1. Tamaño operacional de clases

La métrica TOC se aplica a cada una de las clases del diseño con el objetivo de medir la calidad de las mismas con respecto a su grado de responsabilidad, complejidad de implementación y reutilización (Pressman, 2010).
A continuación, se describen los pasos que se llevaron a cabo para aplicar la métrica:

- Cálculo del umbral. El umbral se toma del tamaño general de una clase que se determina sumando todas las operaciones que posee el diseño.
- Calcular el promedio de los umbrales.
- Teniendo en cuenta los valores antes obtenidos, se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla 4.

Tabla 4 Rango de valores para medir la afectación de los atributos de calidad (TOC).

Atributos de calidad	Clasificación	Criterio
Responsabilidad	Baja	Umbral<= Promedio
	Media	Promedio < Umbral < = 2* Promedio
	Alta	Umbral > 2* promedio
Complejidad de implementación	Baja	Umbral <= Promedio
	Media	Promedio < Umbral < = 2* Promedio
	Alta	Umbral > 2* promedio
Reutilización	Baja	Umbral > 2* promedio
	Media	Promedio < Umbral < = 2* Promedio
	Alta	Umbral <= Promedio

Fuente:(Lorenz, 1994)

En las tablas 5 y 6 se muestran los datos obtenidos una vez aplicada la métrica TOC sobre cada una de las clases del diseño de lasolución propuesta. En la tabla 6 se utilizan las siguientes abreviaturas:

- **Resp:** para el atributo de Responsabilidad
- **Comp. Imp:** para el atributo de Complejidad de implementación
- **Reut:** para el atributo de Reutilización

Tabla 5 Total de clases y promedio de procedimientos con la aplicación de la métrica TOC.

Total de clases	27
Promedio de procedimientos	3.481

Fuente: elaboración propia

En la figura 16 se muestran los resultados en porcientos obtenidos a partir del análisis de los datos mostrados en las tablas 5 y 6, los cuales permiten evaluar los atributos: responsabilidad, complejidad de implementación y reutilización con la aplicación de la métrica TOC.

Tabla 6 Resultados obtenidos por clases luego de aplicada la métrica TOC.

No.	Clase	Cantidad de procedimientos	Resp	Comp. Imp	Reut
1	*ConsecutivoP.java*	4	Media	Media	Media
2	*ConsecutivoPRepository.java*	1	Baja	Baja	Alta
3	*ConsecutivoPService.java*	1	Baja	Baja	Alta
4	*FiltrarFecha.java*	2	Baja	Baja	Alta
5	*Planteamiento.java*	22	Alta	Alta	Baja
6	*PlanteamientoFactory.java*	3	Baja	Baja	Alta
7	*PlanteamientoRepository.java*	1	Baja	Baja	Alta
8	*PlanteamientoService.java*	4	Media	Media	Media
9	*PlanteamientoSpecs.java*	7	Alta	Alta	Baja
10	*Proteccion.java*	5	Media	Media	Media
11	*ProteccionFactory.java*	1	Baja	Baja	Alta

12	*ProteccionRepository.java*	2	Baja	Baja	Alta
13	*ProteccionService.java*	4	Media	Media	Media
14	*ProteccionSpecs.java*	4	Media	Media	Media
15	*ConsecutivoPlanteamientoPServiceImpl.java*	3	Baja	Baja	Alta
16	*PlanteamientoServiceImpl.java*	5	Media	Media	Media
17	*ProteccionServiceImpl.java*	4	Media	Media	Media
18	*PlanteamientoController.java*	4	Media	Media	Media
19	*PlanteamientoDTO.java*	3	Baja	Baja	Alta
20	*PlanteamientoForm.java*	2	Baja	Baja	Alta
21	*PlanteamientoResource.java*	1	Baja	Baja	Alta
22	*PlanteamientoResourceAssembler.java*	2	Baja	Baja	Alta
23	*ProteccionController.java*	4	Media	Media	Media
24	*ProteccionDTO.java*	1	Baja	Baja	Alta
25	*ProteccionForm.java*	1	Baja	Baja	Alta
26	*ProteccionResource.java*	1	Baja	Baja	Alta
27	*ProteccionResourceAssembler.java*	2	Baja	Baja	Alta

Fuente: elaboración propia

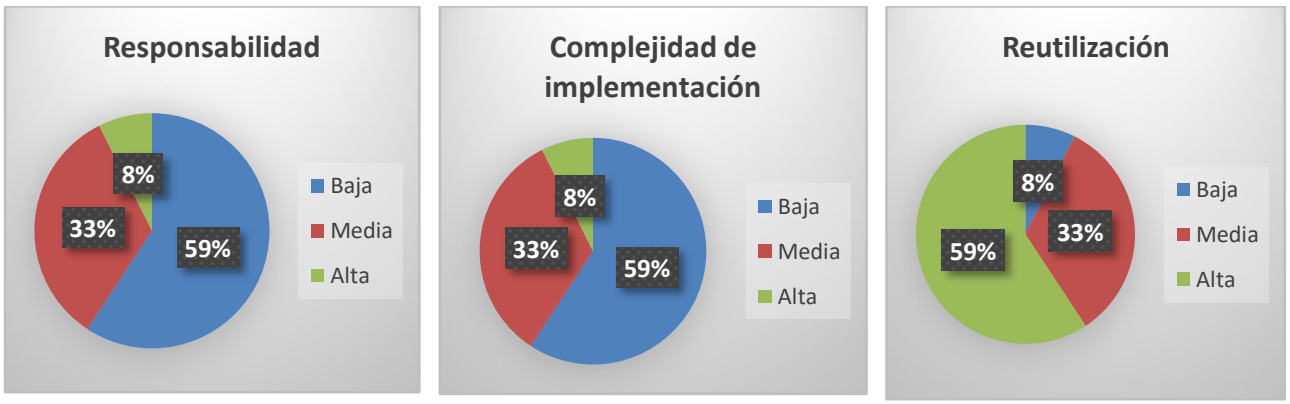


Figura 16 Representación en (%) de los resultados de la aplicación de la métrica TOC.
Fuente: elaboración propia

Con la aplicación de la métrica TOC se obtienen los siguientes resultados:

- Responsabilidad: los resultados fueron satisfactorios, teniendo en cuenta que el 59 % de las clases tienen una responsabilidad baja.

- Complejidad de implementación: los resultados fueron positivos, pues se demostró que el 59 % de las clases tienen una complejidad baja.

- Reutilización: los resultados obtenidos fueron satisfactorios, demostrándose que solo el 8% de las clases tiene una baja reutilización, mientras el 59 % tiene una reutilización alta.

El análisis de los resultados obtenidos con la aplicación de la métrica TOC demuestra que el diseño de lasolución propuesta tiene una baja responsabilidad y complejidad de implementación de las clases que lo componen, así como un alto grado de reutilización de estas. Todo lo anterior facilita la obtención de una solución informática con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios.

3.3.2. Relaciones entre clases

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso en su aplicación es evaluar los siguientes atributos de calidad: acoplamiento, complejidad de mantenimiento, reutilización de cada clase y la cantidad de pruebas que cada clase requiere (Pressman, 2010).
A continuación, se exponen los pasos que se llevaron a cabo para aplicar la métrica a todas las clases de la solución propuesta en la presente investigación:

- Determinar la Cantidad de Relaciones de Uso (CRU) que poseen las clases a medir.
- Calcular el promedio de las CRU.
- Determinar la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la tabla 7, teniendo en cuenta los valores antes obtenidos.

Tabla 7 Rango de valores para medir la afectación de los atributos de calidad (RC).

Atributos de calidad	Clasificación	Criterio
Acoplamiento	Ninguna	CRU=0
	Baja	CRU=1
	Media	CRU=2
	Alta	CRU>2

Complejidad de mantenimiento	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio
Reutilización	Baja	CRU > 2* promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU <= Promedio
Cantidad de pruebas	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio

Fuente:(Lorenz, 1994)

En las tablas 9 y 10 se muestran los datos obtenidos una vez aplicada la métrica RC sobre cada una de las clases del diseño de lasolución propuesta. En la tabla 10 se utilizan las siguientes abreviaturas:

- **Acop:** para el atributo de acoplamiento.
- **Comp. Mant:** para el atributo de complejidad de mantenimiento.
- **Reut:** para el atributo de reutilización.
- **Cant. Prueb:** para el atributo cantidad de pruebas.

Tabla 8 Total de clases y promedio de asociaciones de uso con la aplicación de la métrica RC.

Total de clases	27
Promedio de asociaciones de uso	1.703

Fuente: elaboración propia

Tabla 9 Resultados obtenidos por clases luego de aplicada la métrica RC.

No.	Clase	Cantidad de Relaciones de Uso	Acop.	Comp. Mant.	Reut.	Cant. Prueb.
1	*ConsecutivoP.java*	1	Bajo	Baja	Alta	Baja
2	*ConsecutivoPRepository.java*	1	Bajo	Baja	Alta	Baja
3	*ConsecutivoPService.java*	1	Bajo	Baja	Alta	Baja

4	*FiltrarFecha.java*	1	Bajo	Baja	Alta	Baja
5	*Planteamiento.java*	1	Bajo	Baja	Alta	Baja
6	*PlanteamientoFactory.java*	1	Bajo	Baja	Alta	Baja
7	*PlanteamientoRepository.java*	2	Medio	Media	Media	Media
8	*PlanteamientoService.java*	1	Bajo	Baja	Alta	Baja
9	*PlanteamientoSpecs.java*	3	Alta	Media	Media	Media
10	*Proteccion.java*	1	Bajo	Baja	Alta	Baja
11	*ProteccionFactory.java*	2	Medio	Media	Media	Media
12	*ProteccionRepository.java*	2	Medio	Media	Media	Media
13	*ProteccionService.java*	2	Medio	Media	Media	Media
14	*ProteccionSpecs.java*	4	Alta	Alta	Baja	Alta
15	*ConsecutivoPlanteamientoPServiceImpl.java*	1	Bajo	Baja	Alta	Baja
16	*PlanteamientoServiceImpl.java*	3	Alto	Media	Media	Media
17	*ProteccionServiceImpl.java*	3	Alta	Media	Media	Media
18	*PlanteamientoController.java*	3	Alta	Media	Media	Media
19	*PlanteamientoDTO.java*	1	Bajo	Baja	Alta	Baja
20	*PlanteamientoForm.java*	1	Bajo	Baja	Alta	Baja
21	*PlanteamientoResource.java*	1	Bajo	Baja	Alta	Baja
22	*PlanteamientoResourceAssembler.java*	2	Medio	Media	Media	Media
23	*ProteccionController.java*	3	Alta	Media	Medi	Media

					a	
24	*ProteccionDTO.java*	1	Bajo	Baja	Alta	Baja
25	*ProteccionForm.java*	1	Bajo	Baja	Alta	Baja
26	*ProteccionResource.java*	1	Bajo	Baja	Alta	Baja
27	*ProteccionResourceAssembler.java*	2	Medio	Media	Media	Media

Fuente: elaboración propia

En la figura 17 se muestran los resultados en por cientos obtenidos a partir del análisis de los datos mostrados en las tablas 9 y 10, los cuales permiten evaluar los atributos: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas, con la aplicación de la métrica RC.

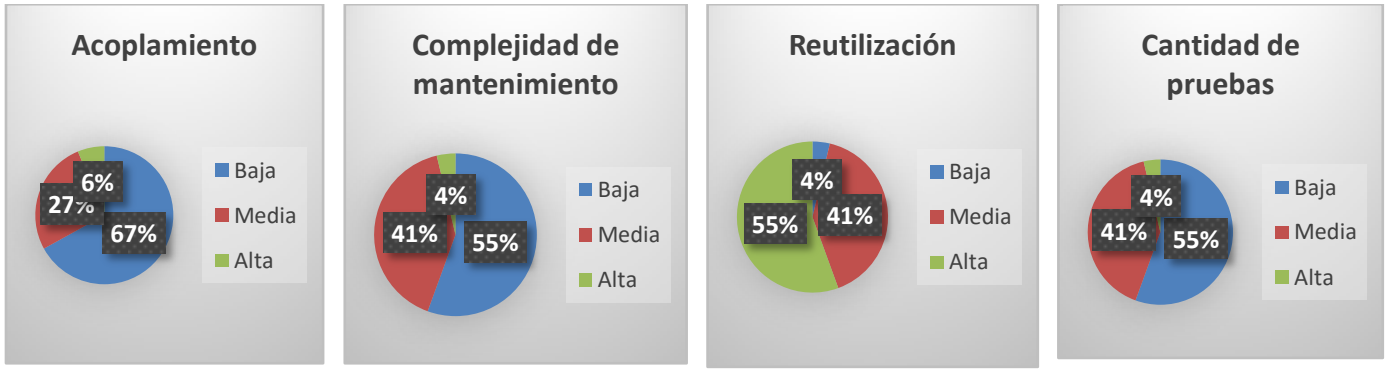


Figura17 Representación en (%) de los resultados de la aplicación de la técnica RC.
Fuente: elaboración propia.

Con la aplicación de la métrica RC se obtienen los siguientes resultados:

- Acoplamiento: los resultados obtenidos son positivos teniendo en cuenta que el 67 % de las clases poseen un bajo acoplamiento.
- Complejidad de mantenimiento: los resultados mostrados en la figura anterior, demuestran que el 55 % de las clases delasolución propuesta presentan una complejidad de mantenimiento baja, facilitando así las futuras actividades de soporte sobre el mismo.
- Reutilización: los resultados obtenidos fueron satisfactorios, demostrándose que el 55 % de las clases tienen una reutilización alta.

- Cantidad de pruebas: los resultados demuestran que el 55 % de las clases poseen un bajo grado de esfuerzo a la hora de realizar cambios, rectificaciones o pruebas de software sobre ellas.

Con los resultados obtenidos una vez aplicada la métrica RC se concluye que el diseño de lasolución propuesta tiene una baja complejidad de mantenimiento y acoplamiento entre sus clases. Además, se requiere de un bajo grado de esfuerzos para realizar cambios sobre la mayoría de las clases y estás a su vez presentan un elevado por ciento de reutilización. Todo lo anterior facilita la obtención de una solución informática escalable, con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios.

3.4. Pruebas de Software

Para la validación de la solución propuesta se aplican pruebas de software organizadas a partir de las disciplinas definidas para la etapa de pruebas en la metodología AUP variación UCI(Pruebas internas, Pruebas de liberación y Pruebas de Aceptación). En el caso de las disciplinas pruebas internas y pruebas de liberación se aplican los niveles de unidad, integración y sistema. Para la disciplina pruebas de aceptación se aplica el nivel de aceptación.

3.4.1. Pruebas internas

A continuación, se detalla cómo el equipo de desarrollo ejecutó las pruebas internas a la solución propuesta, organizada en los niveles de unidad, integración y sistema, aplicando los correspondientes métodos y tipos de pruebas.

Pruebas a nivel de unidad

En el caso de la presente investigación se realizaron pruebas a nivel de unidad aplicando los métodos de Caja blanca y Caja negra. A continuación, se describen los resultados obtenidos con la aplicación de estos métodos.

Método de Caja blanca:

El método de Caja blanca posibilita el desarrollo de casos de prueba que garanticen la ejecución, al menos una vez, de las rutas independientes(Pressman, 2010). En el caso de la presente investigación el método fue ejecutado aplicando laTécnica de ruta básica.

La técnica de ruta básicatiene como objetivo comprobar que cada ruta se ejecute independiente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño. Esta

técnica debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, con el objetivo de asegurar que cada ruta independiente sea ejecutada por lo menos una vez en el sistema(Pressman, 2010).

En la validación delasolución propuesta la técnica de ruta básica se aplicó a todos los métodos de las clases controladoras, teniendo en cuenta que estos agrupan las principales funcionalidades de la solución. A continuación, se describe la aplicación del método de Caja blanca sobre la funcionalidad *obtenerPlanteamiento*(ver figura 18) de la clase *PlanteamientoController.java,* teniendo en cuenta que es una de las funcionalidades de mayor importancia en la solución propuesta.

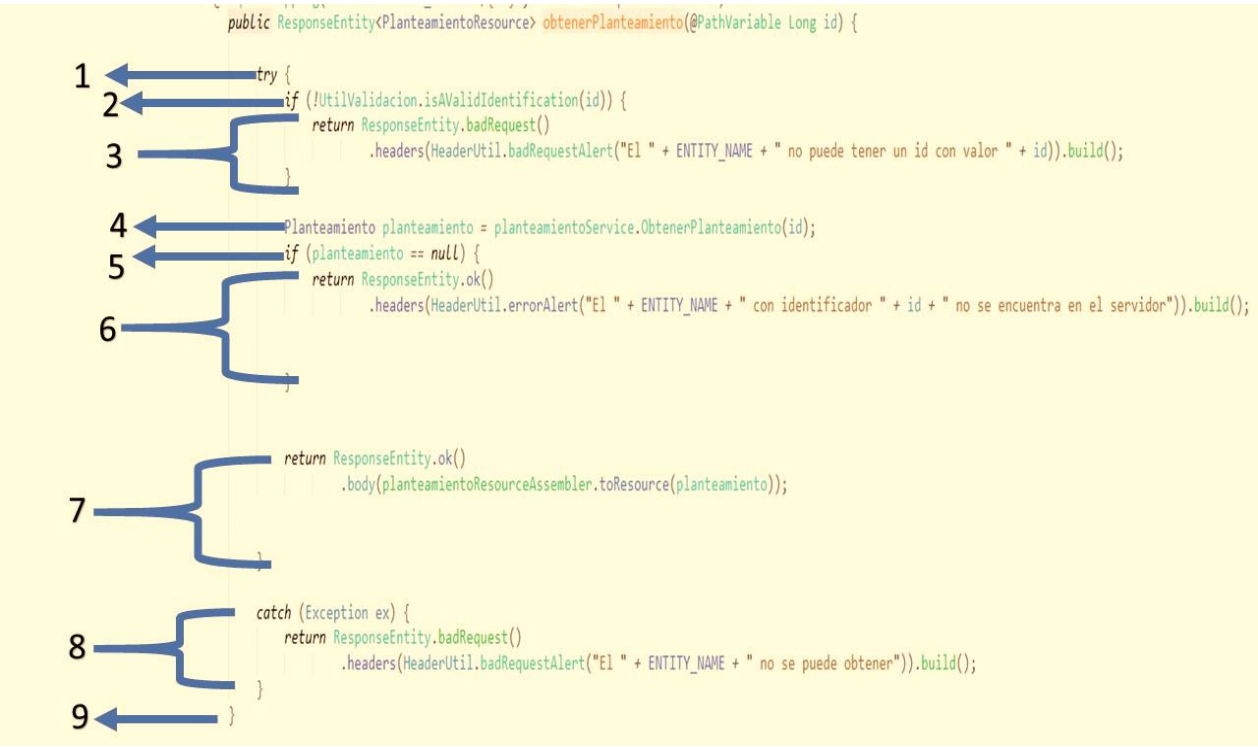


Figura 18método *obtenerPlanteamiento*
Fuente: elaboración propia

Una vez definido el código sobre el cual se aplica el método, los pasos a seguir para desarrollar la técnica de ruta básica son los siguientes:

1) Confeccionar el grafo de flujo: este muestra el flujo de control lógico (Pressman, 2010). Está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.

- Regiones: son las áreas delimitadas por aristas y nodos.

En la siguiente figura se muestra el grafo del flujo obtenido:

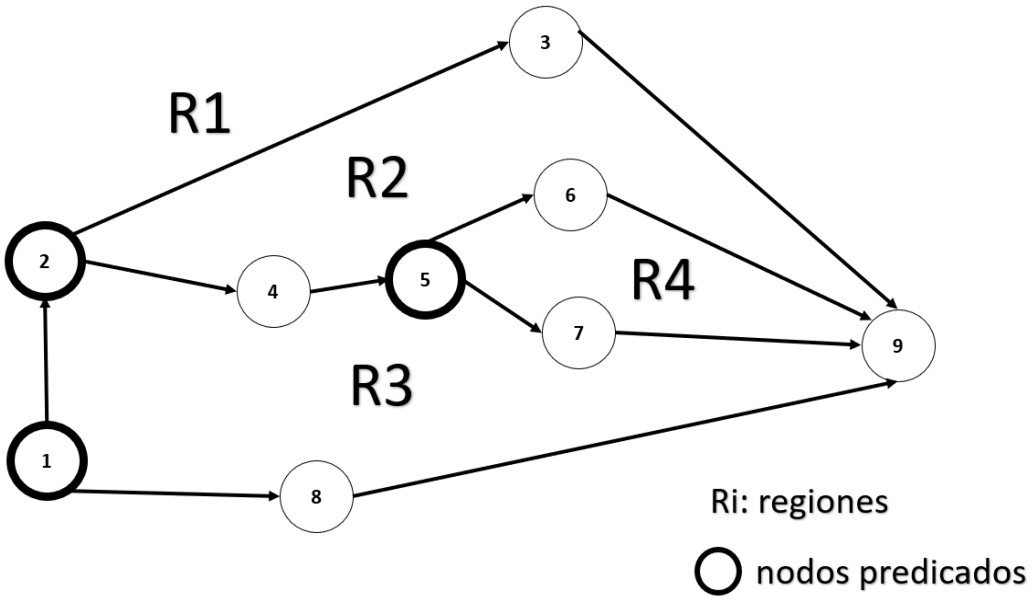


Figura 19 Grafo de flujo del método *obtenerPlanteamiento.*
Fuente: elaboración propia

2) Calcular la complejidad ciclomática:

El valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y brinda una cota superior para el número de pruebas que se deben realizar a fin de asegurar que todos los enunciados se ejecutaron al menos una vez (Pressman, 2010).

La complejidad ciclomática se calcula de tres formas diferentes, las cuales deben llegar al mismo resultado para comprobar que el cálculo es el correcto (Pressman, 2010).

- El número de regiones del grafo de flujo corresponde a la complejidad ciclomática.
- La complejidad ciclomática V(G) para un grafo de flujo G se define como:

V(G)=E-N+2

Donde E es el número de aristas del gráfico de flujo y N el número de nodos del gráfico de flujo.

- La complejidad ciclomática V(G) para un grafo de flujo G también se define como

V(G)=P+1

Donde P es el número de nodos predicado (nodos de donde parten al menos dos aristas) contenidos en el grafo de flujo G.

En el grafo de flujo de la figura 19, la complejidad ciclomática puede calcularse usando cada una de las vías anteriormente descritas:

1. El grafo de flujo tiene 4 regiones, por tanto, V(G) = 4
2. V(G) = (11 aristas - 9 nodos) +2 = 4
3. V(G) = 3 nodos predicado + 1= 4

Por tanto, la complejidad ciclomática del grafo de flujo de la figura 19 es 4.

3) Determinar un conjunto básico de rutas linealmente independientes:

El valor de V(G) proporciona la cuota superior sobre el número de rutas linealmente independientes a través de la estructura de control del programa (Pressman, 2010). En el caso de la funcionalidad *obtenerPlanteamiento*, se definen 4 rutas básicas:

Ruta básica # 1: 1, 8, 9
Ruta básica # 2: 1, 2, 3, 9
Ruta básica # 3: 1, 2, 4, 5, 6, 9
Ruta básica # 4: 1, 2, 4, 5, 7, 9

4) Obtención de casos de prueba (CP):

Una vez definidas las rutas, se procede a diseñar los casos de prueba para cada una de las rutas básicas obtenidas. A continuación, en la tabla 11 se describen los casos de prueba definidos para lascuatro rutas básicas obtenidas con la aplicación de la técnica.

Tabla 10 Casos de pruebas.

No.	Entrada	Resultado
1	Ocurre un error en el sistema.	Lanza una excepción.
2	Recibe un id incorrecto.	El sistema responde que no se puede obtener un planteamiento con ese id.
3	Recibe un id correcto, pero no se encuentra un planteamiento con ese id en la base de datos.	El sistema responde que no se encuentra un planteamiento con ese id en el servidor.
4	Recibe un id correcto, y encuentra	El sistema retorna el planteamiento

	el planteamiento con ese id en la base de datos.	con ese id.

Fuente: elaboración propia

Una vez ejecutados todos los casos de pruebas obtenidos con la técnica empleada, se concluye que los mismos fueron probados satisfactoriamente, corrigiéndose los hallazgos surgidos en una primera iteración y comprobándose su corrección en una segunda. Al concluir la prueba se demuestra que todas las funcionalidades delasolución se ejecutan satisfactoriamente, quedando libres de código repetido o innecesario.

Método de Caja negra:

El método de Caja negra se centra en los requisitos funcionales del software. Este permite al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. El método de Caja negra no es una opción frente a Caja blanca. Es, en cambio, un enfoque complementario que tiene probabilidades de describir una clase diferente de errores de los que se identifican con los métodos de Caja blanca (Pressman, 2010).

El método de Caja negra se ejecuta a partir del desarrollo de pruebas funcionales, con la intención de identificar errores en las siguientes categorías (Pressman, 2010):

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y término.

Para desarrollar el método de Caja negra se encuentra el uso de las técnicas (Pressman, 2010):

- Partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Análisis de valores límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Con el propósito de evaluar a nivel de equipo de desarrollo el correcto funcionamiento y diseño de la solución propuesta, se realizaron pruebas funcionales en los niveles de unidad, integración y sistema. Las pruebas funcionales a nivel de unidad permitieron comprobar que cada uno de los componentes y módulos de la solución funcionan correctamente de forma independiente. Luego con las pruebas a

nivel de integración se comprobó que cada uno de los módulos y componentes se integraron satisfactoriamente. Por último, se realizaron las pruebas a nivel de sistema, comprobándose que luego de la integración la solución funciona en correspondencia con cada uno de los requisitos funcionales que dieron lugar a su desarrollo. Las pruebas se ejecutaron aplicando el método de Caja negra con el uso de las técnicas partición de equivalencia y análisis de valores límites.Como herramientas de apoyo se utilizaron para el caso de las pruebas funcionales las descripciones de caso de pruebas (DCP). En el anexo 5 se puede consultar la DCP correspondiente al RF37. Registrar planteamiento. El resto de las DCP generadas para la validación del componente se encuentran entre los artefactos entregables de la tesis.A continuación, en la figura 20 se muestran los resultados de las pruebas funcionales a nivel de unidad, mostrándose la cantidad de No conformidades (NC) por iteración.

Figura 20NC detectadas al aplicar el método de Caja negra a nivel de unidad.
Fuente: elaboración propia

En la disciplina de pruebas internas, las pruebas funcionales a nivel de unidad se ejecutaron en un total de 5 iteraciones, como muestra la figura 20. Al terminar cada iteración todas las no conformidades (NC) identificadas fueron solucionadas por el equipo de desarrollo antes de iniciar la próxima iteración. Las 7 NC de la primera iteración fueron clasificadas de tipo funcionalidad.Las6NC de la segunda iteración se distribuyeron en 3 de redacción, 1 de funcionalidad y 2 ortografía. En la tercera iteración se detectaron 4 NC, clasificadas en 2 de funcionalidad y 2 de consistencia y estándares. En la cuarta iteración se encontraron 3 NC, clasificadas en 1 de redacción y 2 de

consistencia y estándares. En la quinta iteración los resultados fueron satisfactorios, obteniéndose cero NC. Este resultado demostróque cada uno de los módulos y componentes quedaron listos para ser integrados.

Pruebas a nivel de integración y de sistema

Con el propósito de evaluar a nivel de equipo de desarrollo la integración delos componentes y módulos desarrollados para la solución propuesta y su correcto funcionamiento una vez integrados, se realizaron pruebas funcionales a nivel de integración y de sistema, aplicando el método de Caja negra con el uso de las técnicas partición de equivalencia y análisis de valores límites, utilizando las mismas DCP usadas en las pruebas a nivel de unidad. A continuación, en la figura 21 se muestran los resultados de la aplicación del método.

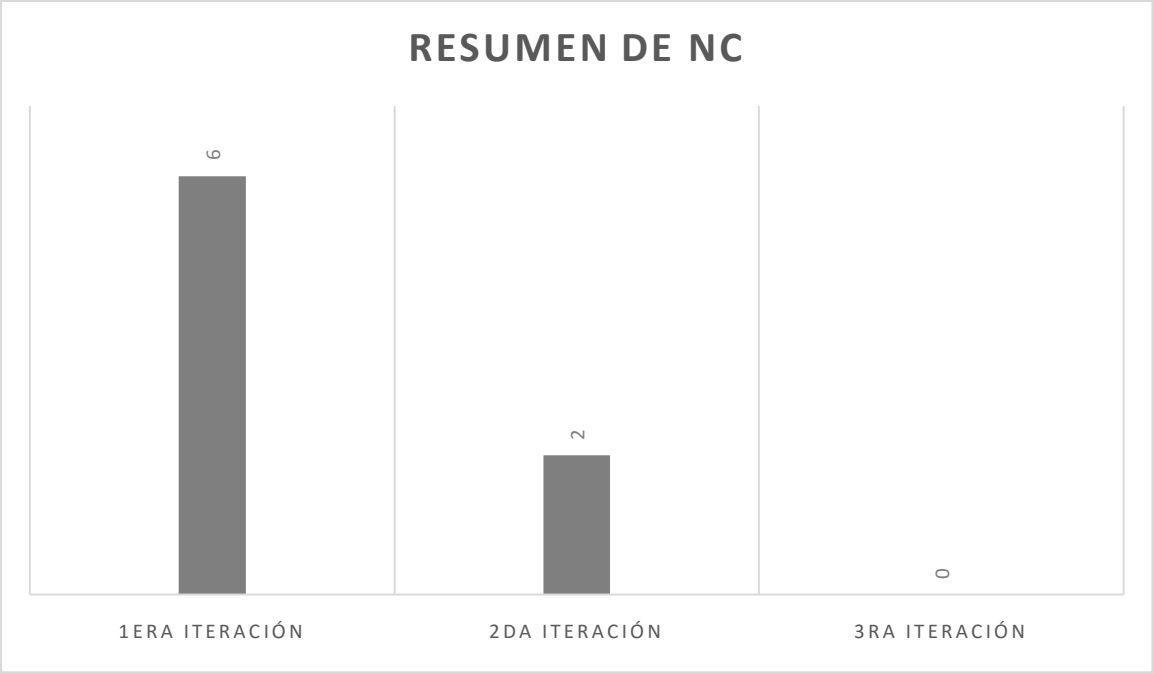


Figura 21NC detectadas al aplicar el método de Caja negra a nivel de integración y sistema.
Fuente: elaboración propia

En la disciplina de pruebas internas, las pruebas funcionales a nivel de integración y sistema se ejecutaron en un total de 3 iteraciones, como muestra la figura 21. Al terminar cada iteración todas las NC identificadas fueron solucionadas por el equipo de desarrollo antes de iniciar la próxima iteración. En la primera iteración se detectaron un total de 6 NC, 3 de estas fueron de funcionalidad, derivadas de un error en la implementación de los RF 1, 6 y 11, las otras 3 NC fueron de validación, derivadas de la no validación de algunos campos obligatorios en los formularios correspondientes a los RF 11,

17 y 25. En la segunda iteración se originaron 2 NC clasificadas de tipo consistencia y estándares, pues en los formularios correspondientes a los RF 26 y 36 existían campos para los cuales el tamaño de letra del nombre no se correspondía con el resto de la solución. En la tercera iteración los resultados fueron satisfactorios, obteniéndose cero NC. Este resultado demostró que los componentes y módulos de lasoluciónse integraron de forma satisfactoria y que el funcionamiento del software obtenido está en correspondencia con los RF que dieron lugar a su desarrollo.

3.4.2. Pruebas de liberación

Con el propósito de evaluar la integración de los componentes y módulos desarrollados y su correcto funcionamiento una vez integrados, se realizaron pruebas de libración de tipo funcionales a nivel de integración y de sistema, así como pruebas de usabilidad. Para el desarrollo de estas pruebas se utilizó el método de Caja negra con el empleo de las técnicas partición de equivalencia y análisis de valores límites, aplicando las mismas DCP usadas en las pruebas internas, así como listas de chequeo para el caso de las pruebas de usabilidad. Estas pruebas fueron realizadas por el equipo de calidad de CEGEL. A continuación, en la figura 22 se muestran los resultados obtenidos con el desarrollo de estas pruebas.

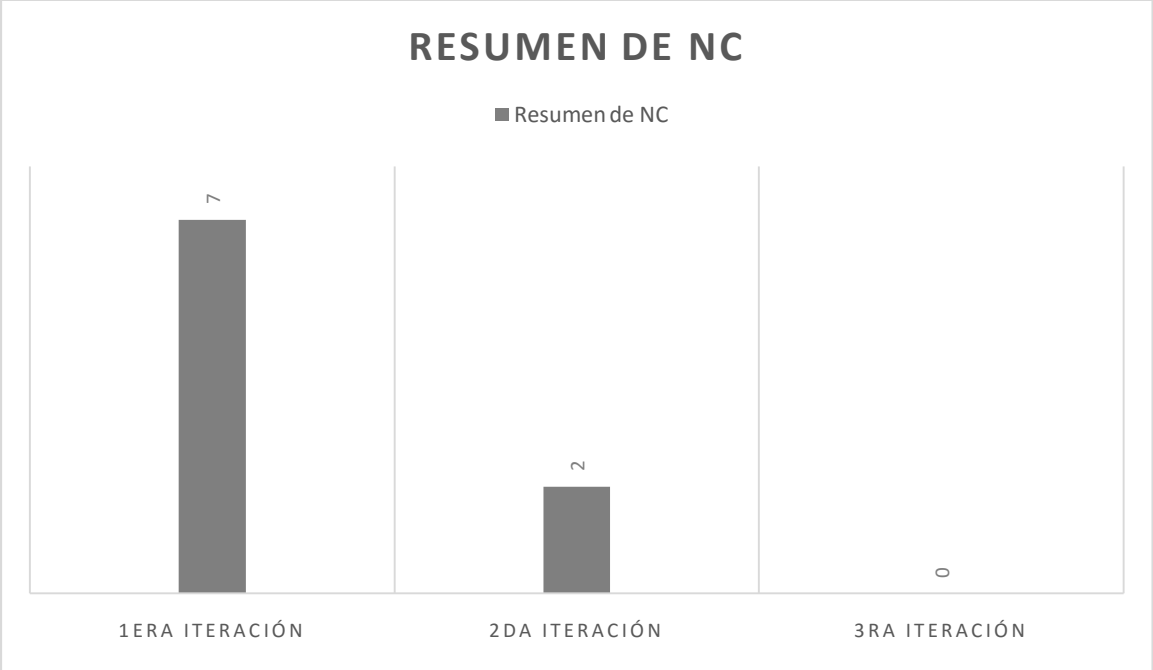


Figura 22NC detectadas en las pruebas de liberación.
Fuente: elaboración propia

En la disciplina de pruebas de liberación, las pruebas funcionales a nivel de integración y sistema se

ejecutaron en un total de 3 iteraciones, como muestra la figura 22. Al terminar cada iteración todas las NC identificadas fueron solucionadas por el equipo de desarrollo antes de iniciar la próxima iteración. En la primera iteración se detectaron un total de 7 NC, estas se clasifican en 1 de redacción identificada en el nombre del formulario correspondiente al RF 6, otras 2 NC fueron de funcionalidad, al existir error al pulsar el botón aceptar en las funcionalidades correspondientes a los RF 2 y 7, el resto de las NC fueron 2 de tipo estética y diseño, relacionadas con los RF 3 y 8, así 2 NC de tipo consistencia y estándares identificadas en los tamaños de letras de las pantallas correspondientes a las RF 9 y 14.En la segunda iteración se detectaron 2 NC de tipo consistencia y estándares, pues en los formularios correspondientes a los RF 25 y 37 existían campos para los cuales el tamaño de letra del nombre no se correspondía con el resto de la solución. En la tercera iteración los resultados fueron satisfactorios, obteniéndose cero NC. generándose así el Acta de Liberación por parte de la Asesora de Calidad de CEGEL (ver Anexo 3).

3.4.3. Pruebas de aceptación

En el caso de la presente investigación se realizaron pruebas a nivel de aceptación a partir de encuentros sostenidos con especialistas del MINCINcon el objetivo de validar que el sistema obtenido cumple con las necesidades del cliente, teniendo en cuenta los RF identificados al iniciar el proyecto. Los resultados obtenidos fueron satisfactorios, avalados por la compañera YalinaGarbey Rivera, Directora de Protección al consumidor en el MINCIN. Al finalizar las sesiones de trabajo se generó el Acta de Aceptación del Producto (ver Anexo 4).

3.5.Validación de los resultados de la investigación

Teniendo en cuenta que en la investigación realizada se define como idea a defender **“**si se desarrolla una personalización del SIGAP en el MINCIN, que permita gestionar las informaciones relacionadas con las reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas procesadas en la dirección de Protección al consumidor de este ministerio, se logra celeridad en el proceso y se reduce la ocurrencia de errores”; para analizar la relación causa efecto entre la variable independiente “si se desarrolla una personalización del SIGAP” y las variables dependientes “se logra celeridad en el proceso y se reduce la ocurrencia de errores”. Los autores de la presente investigación, definen un conjunto de criterios de medida que permiten verificar cómo a través delasolución desarrollada se logra la relación entre ambas variables. La obtención de estos criterios se realiza a partir de las principales deficiencias identificadas en la situación problemática que dan lugar al desarrollo de la solución propuesta.

Para el caso de la evaluación de la variable celeridad se definen los criterios:

- Información oportuna: para el dominio de la investigación los autores definen como información oportuna, la inmediatez con la cual los usuarios acceden a los datos que se generan enel procesogestión dereclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas.
- Proceso de generación de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas: describe el procedimiento que los trabajadoresde la dirección de Protección al consumidor del MINCIN deben seguir para generar los mismos.

Por otra parte, para la evaluación de la variable posibilidad de errores se define el criterio:

- Interacción entre la solución de atención a la población y protección al consumidor: describe el procedimiento que los trabajadores de la dirección de Protección al consumidor siguen para registrar una solución.

A continuación, en la tabla 12, se evalúan cada uno de los criterios de medida definidos anteriormente. La evaluación se realiza estableciendo un antes del desarrollo delasolución y un después. Esta evaluación se realiza con el propósito de verificar cómo mediante la solución propuesta se agilizan los procesos y se evitan errores. Los datos que se tienen en cuenta en la comparación fueron tomados a partir de un piloto realizado en el MINCIN.

Tabla 11 Evaluación de los criterios de medidas definidos.

Criterios de medida	Antes de la solución	Después de la solución
Información oportuna	Los trabajadores de la dirección de Protección al consumidor del MINCIN plantean que en la actualidad la información no está centralizada en un único servidor, por lo que se debe enviar la información requerida a través del correo electrónico, o mediante una llamada telefónica.	La solución posee un único servidor para todo el país por lo que, quien desea cualquier tipo de información podrá consultarla en todo momento desde cualquier sucursal

Proceso de generación de reclamaciones, quejas, denuncias, sugerencias, planteamientos y respuestas	Los trabajadores de la dirección de Protección al consumidor del MINCIN plantean que en la actualidad mucha de la información que se gestiona solo se registra en papeles, impidiendo tener un buen control y un rápido acceso a la información requerida	La solución implementada presenta un único servidor de base de datos, el cual posee toda la información gestionada por la dirección de Protección al consumidor, posibilitando un buen control y un rápido acceso a la información requerida
Interacción entre la solución de atención a la población y protección al consumidor	Los trabajadores de la dirección de Protección al consumidor del MINCIN plantean que en la actualidad necesitan consultar la solución implementada en *Acces* para obtener la información de los escritos, para luego darle soluciones a estos, en la solución implementada en la hoja de cálculo. Esto se ve expuesto a errores humanos debido al traspaso de información manual.	La solución permite buscar los escritos y a través de un botón directamente darle soluciones, disminuyendo la posibilidad de errores humanos.

Fuente: elaboración propia

La comparación realizada en la tabla anterior a través de los criterios de medida antes definidos demuestra que, utilizando la solución desarrollada, se reduce el tiempo y la posibilidad de errores humanos en el procesamiento de la información que gestiona la dirección de Protección al consumidor del MINCIN.

3.6. Conclusiones parciales

- La aplicación de técnicas para la validación de los requisitos garantizó llegar a las disciplinas de análisis y diseño e implementación, con requisitos libres de ambigüedades, que describen cada una de las necesidades del cliente.
- El uso de métricas de validación del diseño certifica la obtención de una solución flexible al mantenimiento y a la introducción de cambios.
- La realización de pruebas internas y de liberación en los niveles de unidad, integración y sistema, con el empleo de sus respectivos métodos y técnicas, certifican que la solución obtenida cumple con cada uno de los RF y RNF que dieron lugar a su implementación, avalado por el acta de liberación emitida por el grupo de calidad de CEGEL.
- Con la verificación del resultado de la investigación, se demuestra el cumplimiento de la relación causa efecto de la variable independiente “si se desarrolla una personalización del SIGAP” sobre las variables dependientes “se logra celeridad en el proceso y se evita posibilidad de errores”.

CONCLUSIONES GENERALES

- El estudio y análisis de los referentes teóricos vinculados con el objeto de estudio de la presente investigación permitió comprender los principales conceptos asociados al dominio del problema, obtener características de sistemas homólogos con el fin de ser incorporarlas a la solución propuesta, así como definir las herramientas, tecnologías y buenas prácticas a utilizar en el desarrollo de la personalización.
- El empleo de técnicas de captura de requisitos, patrones de diseño y arquitectónicos, así como el uso de estándares de codificación en la implementación de la solución propuesta, permitió obtener unapersonalización del SIGAP para el MINCIN, flexible al mantenimiento y a la introducción de cambios.
- El desarrollo de pruebas de internas y de liberación en los niveles de unidad, integración y sistema, permitieron obtener un acta de liberación emitida por el grupo de calidad de CEGEL que corrobora el correcto funcionamiento dela personalización del SIGAP para el MINCIN, así como las pruebas de aceptación permitieron obtener el acta de aceptación del software por parte del cliente.
- Con la verificación de la relación causa efecto de la variable independiente sobre la variable dependiente de la investigación, se demuestra que con el desarrollo delasolución propuesta se reduce el tiempo y la posibilidad de errores en el procesamiento de la información gestionada por la dirección de Protección al consumidor del MINCIN.

RECOMENDACIONES

- Establecer permisos a través de niveles de acceso al sistema, de forma tal que el software sea desplegado a nivel nacional y cada instancia solo tenga acceso a sus casos o aquellos a los cuales se le asigna permisos.
- Implementar una aplicación móvil que se integre al SIGAP, a través de la cual los promoventes puedan añadir nuevos escritos de quejas y dar seguimiento al estado de estos.

BIBLIOGRAFÍA REFERENCIADA

Alegsa. 2019. [Online] 2019. http://www.alegsa.com.ar/Dic/framework.php.

Álvarez, Sara. 2007. Sistemas gestores de bases de datos. Introducción a este concepto y características especiales. [Online] 2007. http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

Association of Modern Technologies Professionals. 2019. It Knowledge portal. [Online] 2019. http://www.itinfo.am/eng/software-development-methodologies/.

AVOTZ. 2013. Pruebas de Usabilidad. *AVOTZ. Web Works.* [Online] 2013. www.avotz.com/es/component/content/article/32.html.

Ayala Catari, Iván, Romero Marca, Yecid Tomas and Serrano Urzagaste, Ronald Javier. 2010. Estudio de herramientas CASE de soporte UML y UML2. *Scribd.com.* [Online] 2010. http://es.scribd.com/doc/25374125/Estudio-de-Herramientas-CASE-de-Soporte-a-UML-y-UML2. DOI: 25374125.

Balsamiq Studios. 2016. Balsamiq. *Balsamiq.* [Online] Balsamiq Studios, 2016. [Cited: 08 27, 2020.] https://balsamiq.com/learn/articles/.

Banco Central de Cuba. 2006. Banco Central de Cuba. [Online] 2006. [Cited: 12 17, 2019.] http://www.bc.gob.cu/.

Benchimol , Daniel. 2011.*HACKING DESDE CERO: Manuales Users (Spanish Edition).* Argentina : Creative Andina Corp., 2011. 987177303X, 9789871773039.

C. Evans, Clark . The Official YAML Web Site. *Sitio web oficial de YAML.* [Online] http://www.yaml.org/.

Carrero , Angel. 2013. Conceptos básicos de ORM (Object Relational Mapping). *Programación en castellano.* [Online] 2013. http://www.programacion.com/articulo/conceptos_basicos_de_orm_object_relational_mapping_349.

Caswell, Brian, Beale, Jay and Baker, Andrew. 2007.*Snort Intrusion Detection and Prevention Toolkit.* Kurasa : Syngress, 2007. 978-1-59749-099-3.

CEGEL.*Manual de Usuario de SIGAP.*

Cepeda Asqui, Jessica Paulina and Tene Reino, Blanca Georgina . 2012. Investigación de la Herramienta Case para el Desarrollo del Sistema Academico Educativo en el Centro de Edcación Básica “Dr. Nicanor Larrea León, Basada en la Arquitectura .Net Framework. [Online] 2012. http://hdl.handle.net/123456789/67. DOI: 67.

Chaffer, Jonathan. Drupal programming from an object-oriented perspective. *Drupal.* [Online] http://drupal.org/node/547518 .

Chile, Consejo Nacional de la cultura y las artes de. 2012. Quiénes Somos. *Cultura.* [Online] 2012. http://www.cultura.gob.cl/institucion/quienes-somos/.

Claudia. 2010. Drupal versus Joomla. *DrupalSoul.* [Online] 2010. http://www.drupalsoul.com/blog/drupal-versus-joomla.

Collis, Ta'eed and Harley , Alexander. 2010.*How to be a Rockstar. WordPress Designer.* s.l. : Rockable Press, 2010. 1707351881.

Consejo Nacional de la cultura y las artes de Chile. Quiénes Somos. *Cultura.* [Online] [Cited: noviembre 20, 2012.] http://www.cultura.gob.cl/institucion/quienes-somos/.

Cubana, Portal de la Televisión. 2019. Portal de la Televisión Cubana. *Portal de la Televisión Cubana.* [Online] 2019. [Cited: 11 16, 2019.] http://www.tvcubana.icrt.cu/destacados/4482-quejas-y-sugerencias-de-los-publicos-mejoras-en-la-pequena-pantalla.

Danysoft. 2013. Embarcadero ER/Studio. [Online] 2013. http://www.codegear-shop.com/Embarcadero-ER/Studio/es.

DeConceptos. 2014. Deconceptos. *Deconceptos.* [Online] 02 11, 2014. [Cited: 11 15, 2019.] https://deconceptos.com/.

Definicion ABC. 2007. Definición ABC. *Definición ABC.* [Online] 2007. [Cited: 12 17, 2019.] www.definicionabc.com.

del Castillo San Félix, Alvaro. 2000. El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha. [Online] 2000. http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html.

Del Pino Valdarrama, Santiago Luis. 2005. Programación extrema en pocos minutos: planificando la transición. Cuba : Tono. Revista Técnica de la Empresa de Telecomunicaciones de Cuba, S.A., 2005. 3, pp. 41-44. 18135056.

DESOFT. 2018. Soy Villa Clara. *Soy Villa Clara.* [Online] 2018. [Cited: 11 16, 2019.] http://www.soyvillaclara.gob.cu/es/asamblea-atencion-poblacion.

Drupal. 2013. Hooks. *Drupal API.* [Online] 2013. http://api.drupal.org/api/drupal/includes!module.inc/group/hooks/6.

—. 2013. Smile Open Source Solutions Iberia. *Drupal.* [Online] 2013. http://www.smile-iberia.com/Productos/Drupal.

Ecured. 2020. Ecured. *Ecured.* [Online] 2020. [Cited: 02 24, 2020.] https://www.ecured.cu/Atenci%C3%B3n_a_la_Poblaci%C3%B3n_(Cuba).

Editorbfb. 2011. Qué es un entorno de desarrollo integrado, IDE. *Programación Desarrollo.* [Online] 2011. http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/.

Ercoli , Jorge . 2007. Qué es un ORM (object-relational mapping) . *Arquitectura de Sistemas.* [Online] 2007. http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html.

Fakhroutdinov, Kirill. 2015. UML-Diagrams. [Online] 2015. [Cited: 08 27, 2020.] https://www.uml-diagrams.org/component-diagrams.html.

Figueroa, Pablo. 2013. Conceptos en un Diagrama de Implementación. [Online] 2013. http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/implementacion01.html.

Flanagan, David. 2011.*JavaScript : the definitive guide. 6th Edition.* Beijing : Sebastopol, CA : O'Reilly, 2011. 9780596805524; 0596805527.

Garbey Rivera, Yalina. 2019.*La dirección de protección al consumidor.* [interv.] Ramsés Gónzalez Toledo. La habana, 04 19, 2019.

García Leyva, Yordanis. 2019. ¿ Qué es CEGEL y SIGAP? [interv.] Ramsés González Toledo and Leonel Dominguez Ávila. *SIGAP.* La Lisa, 11 14, 2019.

Gauchat, Juan Diego. 2012.*El gran libro de HTML5, CSS3 y JavaScript.* Barcelona : MARCOMBO, S.A., 2012. 978-84-267-1770-2.

Givertex. 2019. Softwareseleccion. *Softwareseleccion.* [Online] 2019. [Cited: 11 05, 2019.] https://www.softwareseleccion.com/givertex-p-3066.

González Boticario, Jesús and Gaudioso Vázquez, Elena. 2001.*"Capítulo 6. JavaScript". Aprender y formar en Internet.* Madrid, España : International Thomson Editors Spain Paraninfo, S. A., 2001. 84-283-2743-2.

Google. 2019. Angular. [Online] 2019. https://angular.io/docs.

—. 2019. Angular. [Online] 2019. https://angular.io/guide/observables-in-angular.

—. 2019. Angular. [Online] 2019. angular.io/guide/dependency-injection.

Guardado, Iván. 2010. Utilizando Doctrine como ORM en PHP. *Web.ontuts.* [Online] 2010. http://web.ontuts.com/tutoriales/utilizando-doctrine-como-orm-en-php/.

Hommel, Scott. 2019.*Estandares_de_codificacion_para_Java.* 2019.

ICOM. 2013.*Código de Deontología del ICOM para los Museos.* s.l. : © ICOM, 2013. 978-92-9012-407-8.

Ing. Oré , Alexander. 2009 . UNIT TESTING - PRUEBAS UNITARIAS. *CalidadySoftware.com .* [Online] 2009 . http://www.calidadysoftware.com/testing/pruebas_unitarias2.php .

Ing. Valdarrama del Pino, Santiago Luis . 2005. Programación extrema en pocos minutos: planificando la transición. *Revista Técnica de la Empresa de Telecomunicaciones de Cuba S.A.* Cuba : s.n., 2005. 3. 18135056. DOI: 28012845.

INTECO. 2009. Ingeniería del software: Metodologías y ciclos de vida. *Scientific Electronic Librery Online (Scielo).* [Online] 2009. http://www.inteco.es/file/N85W1ZWFHifRgUc_oY8_Xg.

—. 2009. Revista Scielo. *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA.* [Online] Marzo 2009. http://www.ieee.org.sv/concapan/descargas/memoria_secciones/Jueves_10/izalc/P72.pdf.

IsoTools. 2019. IsoTools. *IsoTools.* [Online] 2019. [Cited: 11 16, 2019.] http://info.isotools.org/.

Jacobson, Ivar , Booch, Grady and Rumbaugh, James. 2000.*El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación, Inc., 2000. 84-7829-036-2.

Janssen, Thorben. 2019. Thoughts on JAVA. *Implementing the Repository pattern with JPA and Hibernate.* [Online] 2019. https://thoughts-on-java.org/implementing-the-repository-pattern-with-jpa-and-hibernate/.

JetBrains. 2019. JetBrains. [Online] 2019. https://www.jetbrains.com/idea/.

—. 2019. JetBrains. [Online] 2019. https://www.jetbrains.com/webstorm/.

Kendall, Kenneth E. and Kendall, Julie E. 2005.*Análisis y Diseño de Sistemas. Sexta Edición.* México : Pearson Educación de México, S.A. de C.V., 2005. 970-26-0577-6.

Kniberg, Henrik. 2007.*Scrum and XP from the Trenches.* Estados Unidos de América : C4Media Inc., 2007. 978-1-4303-2264-1.

Larman, Craig. 1999.*UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-17-0261-1.

—. 2016.*UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 3ra . s.l. : Prentice-Hall, 2016.

Leffingwell, Dean and Widrig, Don. 2003.*"Using Software Engineering Techniques for Business Modeling. The Unified Modeling Language". Managing software requirements : a use case approach.* United States of America : Pearson Education, Inc., 2003. 0-321-12247-X.

Leyva Samada, Lisandra Isabel . 2009. Flujo de Investigación para la Metodología Ágil SXP. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. [Online] 2009. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_2435_09. TD_2435_09.

Lorenz, M., & Kidd, J. 1994.*Object-oriented software metrics: a practical guide.* New Jersey, Prentice-Hall : s.n., 1994.

Martín Fernández, Francisco J. and Hassan Monter, Yusef. 2003. Revista No Solo Usabilidad. Qué es la Arquitectura de la Información. 2003. Vol. nº 2. 1886-8592.

Martínez Illa, Santi and Mendoza, Roser . 2007. TIC y gestión de la cultura: ¿Políticas e-culturales? *Centro de Estudios y Recursos Culturales. Universidad de Alicante.* [Online] 2007. http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB4QFjAA&url=http%3A%2F%2Fcolombiadigital.net%2Fnewcd%2Fcomponent%2Fdocman%2Fdoc_download%2F302-tic-y-gestion-de-la-cultura-ipoliticas-e-culturales-&ei=7U-oUKfdH_SC0QHsvICYDA&usg=AFQjC.

Megías, Braulio. 2011. ¿Claves primarias naturales o subrogadas? *reThink.net.* [Online] 2011. http://bmegias.wordpress.com/2011/01/31/%C2%BFclaves-primarias-naturales-o-subrogadas/.

Miles, Russ and Hamilton, Kim . 2006.*Learning UML 2.0.* s.l. : O'Reilly, 2006. 978-0-59-600982-3.

MINCIN. 2020. Portal del MINCIN. *Portal del MINCIN.* [Online] 2020. [Cited: 01 05, 2020.] https://www.mincin.gob.cu.

MinCult. 2012. Consejo Nacional de Patrimonio Cultural (CNPC). *Sitio oficial del Ministerio de Cultura de la República de Cuba.* [Online] 2012. [Cited: Noviembre 9, 2012.] http://www.min.cult.cu/loader.php?sec=instituciones&cont=cnpc.

—. 1983.*Decreto No. 118. Reglamento para la Eejecución de la Ley de Protección al Patrimonio.* La Habana : Gaceta Oficial de la República de Cuba, 1983.

—. 2009.*Ley No. 106. Ley del Sistema Nacional de Museos de la República de Cuba.* La Habana : Gaceta Oficial de la República de Cuba, 2009.

Mozilla. 2019. MDN web docs mozilla. [Online] 2019. developer.mozilla.org/en-US/docs/Web/JavaScript.

—. 2019. MDN web docs mozilla. [Online] 2019. developer.mozilla.org/en-US/docs/Web/HTML.

—. 2019. MDN web docs mozilla. [Online] 2019. developer.mozilla.org/en-US/docs/Web/CSS/CSS3.

MSc. García Perdigón, Jorge R., et al. 2009. Manual sobre el trabajo técnico de los museos adscritos al Consejo Nacional de Patrimonio Cultural. *Sitio oficial del Consejo Nacional de Patrimonio Cultural.* [Online] 2009. http://www.cnpc.cult.cu/Portada/Manual_de_museos.pdf.

Netbeans.org. 2012. Bienvenido a NetBeans y www.netbeans.org. *NetBeans.* [Online] 2012. http://netbeans.org/index_es.html.

OMG®. 2012. Introduction to OMG's Unified Modeling Language™ (UML®). *OMG®.* [Online] 2012. [Cited: Enero 8, 2012.] http://www.omg.org/gettingstarted/what_is_uml.htm.

ORACLE. 2019. ORACLE. [Online] 2019. https://docs.oracle.com/en/java/.

Peck, Steven. 2013. Understanding Drupal. *Drupal.* [Online] 2013. http://drupal.org/documentation/understand.

Pentaho Data Integration. 2013. Pentaho Data Integration Kettle ETL tool. *ETL-Tools.Info.* [Online] 2013. [Cited: Marzo 26, 2013.] http://etl-tools.info/en/pentaho/kettle-etl.htm.

Pentaho Open Source Business Intelligence. 2013. La plataforma Pentaho Open Source Business Intelligence. *Portada sobre la plataforma Pentaho Open Source Business Intelligence.* [Online] 2013. http://pentaho.almacen-datos.com/.

—. 2013. Proceso ETL. *ETL-Tools.Info.* [Online] 2013. [Cited: Marzo 26, 2013.] http://etl-tools.info/es/bi/proceso_etl.htm.

Pérez Porto, Julián and Merino, María . 2010. Definicion. de. *Definicion. de.* [Online] 2010. [Cited: 11 15, 2019.] https://definicion.de.

Pivotal. 2019. Spring. [Online] 2019. spring.io/projects/spring-boot.

Pressman, Roger S. 2010.*Ingeniería de Software. Un enfoque práctico.* Septima . México DF : McGraw-Hill INTERAMERICA EDITORES, 2010.

—. 2003.*Ingeniería del Software. Un enfoque práctico. Sexta Edición.* s.l. : Mc Graw Hill, 2003. 970-10-5473-3.

—. 2001. Ingeniería del Software: una tecnología estratificada. *Ingeniería del Software. Un enfoque práctico. Quinta Edición.* España : McGraw Hill, 2001.

Prieto Díaz, Vicente, et al. 2010. Impacto de las tecnologías de la información y las comunicaciones en la educación y nuevos paradigmas del enfoque educativo. *Biblioteca Virtual en Salud.* [Online] 2010. http://bvs.sld.cu/revistas/ems/vol25_1_11/ems09111.htm. 09111.

Puertas Ortega, Juan and Orellana Zubieta, Francisco Javier . 2011. Un-paseo-por-PHP. *Scribd.com.* [Online] 2011. http://es.scribd.com/doc/51830143/Un-paseo-por-PHP. DOI: 51830143.

Raggett, Dave, Le Hors, Arnaud and Jacobs, Ian (Eds.). 2001. Especificación HTML 4.01. *ucistore.* [Online] 2001. ftp://ucistore.uci.cu/documentacion/Programacion/HTML/HTML%201/html401-es.pdf.

Rodríguez Sala, Jesús Javier . 2003.*Introducción a la programación: teoría y práctica.* s.l. : Editorial Club Universitario, 2003. 9788484542742.

Rodríguez, Fran Gil. 2012.*Experto en Drupal 7. Curso de creación y gestión de portales web con Drupal 7. Nivel Avanzado. Aprende Drupal con Forcontu.* s.l. : Forcontu S.L., 2012. 978-84-939410-5-5.

—. 2012.*Experto en Drupal 7. Curso de creación y gestión de portales web con Drupal 7. Nivel Inicial. Aprende Drupal con Forcontu.* s.l. : Forcontu S.L., 2012. 978-84-939410-3-1.

—. 2012.*Experto en Drupal 7. Curso de creación y gestión de portales web con Drupal 7. Nivel Intermedio. Aprende Drupal con Forcontu.* s.l. : Forcontu S.L., 2012. 978-84-939410-4-8.

Rojas, M. J. 2010.*Patrones de Diseño.* 2010.

Rojas, Nohemi. 2010. CMS y Framework dos conceptos distintos. [Online] 2010. http://nohemirojas.wordpress.com/2010/03/25/cms-y-framaework-dos-conceptos-distintos/.

Romero, Gladys Marsi Peñalver. 2008. MA-GMPR-UR2 Metodología ágil para proyectos de software libre. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. [Online] 2008. http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_1309_08. TD_1309_08.

Rouse, Margaret. 2019. TeachTarget. *Database management system (DBMS).* [Online] 2019. https://searchsqlserver.techtarget.com/definition/database-management-system.

—. 2019. TechTarget. *Integrated Development Environment (IDE).* [Online] 2019. searchsoftwarequality.techtarget.com/definition/integrated-development-environment.

—. 2019. TechTarget. *pattern (design pattern).* [Online] 2019. https://searchsoftwarequality.techtarget.com/definition/pattern.

—. 2019. TechTarget. *User Story.* [Online] 2019. https://searchsoftwarequality.techtarget.com/definition/user-story.

Rumbaugh, James , Jacobson, Ivar and Booch, Grady . 2007.*El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison-Wesley Iberoameri, 2007. 9788478290871.

Rumbaugh, James , Jacobson, Ivar and Booch, Grady . 2004. Unified Modeling Language Reference Manual, The (2nd Edition). [Online] 2004. http://my.safaribooksonline.com/book/software-engineering-and-development/uml/0321245628/background/part01. 0321245628.

Sæther Bakken, Stig, Aulbach, Alexander and Schmid, Egon . 2001. Manual de PHP. *ucistore.uci.cu.* [Online] 2001. ftp://ucistore.uci.cu/documentacion/Programacion/PHP/Manual%20de%20PHP.pdf.

Safari. 2019. Safari Books Online. *Oreilly.* [Online] 2019. www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html.

Sánchez Maza, Miguel Ángel . 2012.*JavaScript.* s.l. : IC Editorial, 2012. 978-8495733184.

Sánchez, Tamara Rodríguez. 2015.*Metodología de desarrollo para la Actividad productiva de la UCI. .* La Habana. Cuba : s.n., 2015.

Schardt Chonoles, Michael Jesse and Schardt, James A. 2003.*UML 2 for Dummies.* U.S.A : Wiley Publishing, Inc., 2003. 0764526146.

Sierra, Manuel. 2012. Qué es y para qué sirve el lenguaje CSS (Cascading Style Sheets - Hojas de Estilo). *Aprenderaprogramar.com.* [Online] 2012. http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=546:que-es-y-para-que-sirve-el-lenguaje-css-cascading-style-sheets-hojas-de-estilo&catid=46:lenguajes-y-entornos&Itemid=163).

Software-talk.org. 2012. Netbeans vs Eclipse: An IDE comparison. *Software Talk.* [Online] 2012. http://software-talk.org/blog/2012/01/netbeans-vs-eclipse-an-ide-comparison/.

Sommerville. 2011.*Ingeniería de Software, 9na Edición.* 2011.

Sommerville, Ian. 2005.*Ingeniería del software. Séptima Edición.* Madrid. España : Pearson Educación. S. A., 2005. 84-7829-074-5.

The PostgreSQL Global Development Group. 2019. PostgreSQL. *The World's Most Advanced Open Source Relational Database.* [Online] 2019. https://www.postgresql.org/.

Tinoco Gómez, Oscar, Rosales López, Pedro Pablo and Salas Bacalla, Julio . 2010.*Criterios de selección de metodologías de desarrollo de software .* Perú : Industrial Data. Revista de la Facultad de Ingeniería Industrial, 2010. 1810-9993.

Typescript. 2019. Typescript. [Online] 2019. www.typescriptlang.org.

UA. 2019. Universidad de Alicante. [Online] 2019. https://si.ua.es/es/documentacion/asp-net-mvc-3/.

UCI. 2019. UCI. *UCI.* [Online] 2019. [Cited: 03 06, 2020.] https://www.uci.cu/investigación-y-desarrollo/centrosde-desarrollo/centro-de-gobierno-electronico-cegel.

Vera, H. Solano. 2005. Definición y diseño de la aplicación web Interfaz para el monitoreo de redes de comunicaciones mediante una aplicación web. Tesis Licenciatura Ingeniería en Sistemas Computacionales. *Departamento de Ingeniería en Sistemas*

Computacionales. [Online] 2005. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/solano_v_h/capitulo_4.html.

Vértice, Equipo. 2009.*Diseño Básico de Páginas Web en HTML.* España : Publicaciones Vértice S. L, 2009. 978-84-9931-034-3.

Visual-Paradigm. 2019. Visual-Paradigm. [Online] 2019. www.visual-paradigm.com/solution/freeumltool/.

W3C. 2013. Guía Breve de Servicios Web. *W3C.es.* [Online] 2013. http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

—. 2005. Introducción a la Accesibilidad Web. *W3C.es.* [Online] 2005. http://www.w3c.es/Traducciones/es/WAI/intro/accessibility.

WebSite4Cuba. 2016. Banco Metropolitano. *Banco Metropolitano.* [Online] 2016. [Cited: 12 17, 2019.] http://www.banco-metropolitano.com.cu/.

Wolanin, Peter. 2012. Is Drupal secure? *Drupal.* [Online] 2012. http://drupal.org/documentation/is-drupal-secure .

Young, Ralph R. 2004.*Chapter 1. The Importance of Requirements: What Are Requirements and Why Are They Important? The Requirements Engineering Handbook.* Boston : Artech House Inc., 2004. 1-58053-266-7.

ANEXOS

Anexo 1: Entrevista

Guía de preguntas utilizadas en el desarrollo de la entrevista con los profesionales delMINCIN.

1. ¿Qué mecanismos emplean para la gestión de sus datos?
2. ¿Cuáles son los datos que manejan actualmente?
3. ¿Cómo analizan ustedes los datos que generansus procesos?
4. ¿Cuáles son los errores que generalmente se derivan a partir de la forma en que ustedes procesan la informaciónhoy en día?
5. ¿Quién les solicita a ustedes esta información?
6. ¿Cuánto tiempo tardan en tener lista la información solicitada?
7. Teniendo en cuenta que en la actualidad ustedes procesan toda la información con software de la suit de *Microsoft Office*, que el cual es privativo y que el SIGAP no cumple en su totalidad con los requisitos exigidos. ¿Cuáles son las funcionalidades que ustedes consideran que debe tener una personalización de SIGAP?
8. ¿A parte de estas funcionalidades con que requerimientos creen que deba funcionar?

Anexo 2: Acuerdos tomados en las tormentas de ideas con el cliente

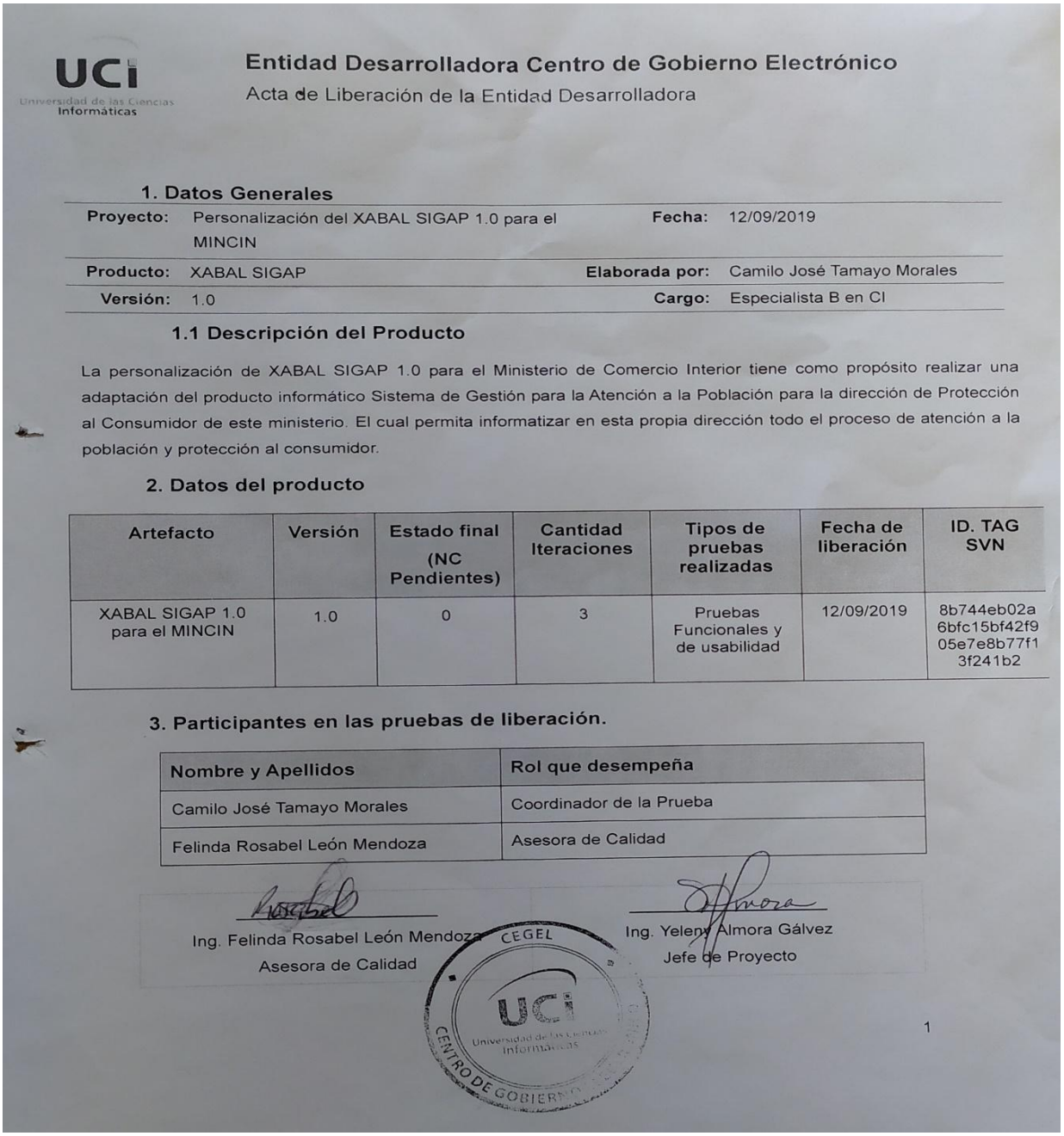
Tabla 12 Acuerdos tomados en las tormentas de ideas con el cliente.

Nº	Acuerdo	Responsable	Fecha de cumplimiento
1	El sistema debe cumplir con todas las funcionalidades actuales que presenta la solución ya existente, implementada en la suit de *Microsoft office*	Leonel Domínguez Ávila	05/11/2019
2	El sistema debe permitir buscar cualquier tipo de dato que se gestione en la entidad, aplicando una cierta cantidad de filtros de búsqueda que permitan llegar al resultado con gran rapidez	Leonel Domínguez Ávila	10/112019
3	El sistema además debe permitir la visualización constante de los escritos.	Leonel Domínguez Ávila	15/11/2019
4	Desde cualquier pantalla, se debe poder acceder a la que se desee.	Leonel Domínguez Ávila	20/11/2019
5	En la interfaz donde se muestren cada uno de los datos que se gestionan, deben de existir criterios de búsqueda a	Leonel Domínguez Ávila	25/11/2019

	través de filtros que permitan organizar la información que se muestra en la interfaz.		

Fuente: elaboración propia

Anexo 3: Acta de Liberación de Productos de Software por el grupo de calidad Centro CEGEL



UCI
Universidad de las Ciencias Informáticas

Entidad Desarrolladora Centro de Gobierno Electrónico

Acta de Liberación de la Entidad Desarrolladora

1. Datos Generales

Proyecto:	Personalización del XABAL SIGAP 1.0 para el MINCIN	**Fecha:**	12/09/2019
Producto:	XABAL SIGAP	**Elaborada por:**	Camilo José Tamayo Morales
Versión:	1.0	**Cargo:**	Especialista B en CI

1.1 Descripción del Producto

La personalización de XABAL SIGAP 1.0 para el Ministerio de Comercio Interior tiene como propósito realizar una adaptación del producto informático Sistema de Gestión para la Atención a la Población para la dirección de Protección al Consumidor de este ministerio. El cual permita informatizar en esta propia dirección todo el proceso de atención a la población y protección al consumidor.

2. Datos del producto

Artefacto	Versión	Estado final (NC Pendientes)	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación	ID. TAG SVN
XABAL SIGAP 1.0 para el MINCIN	1.0	0	3	Pruebas Funcionales y de usabilidad	12/09/2019	8b744eb02a6bfc15bf42f905e7e8b77f13f241b2

3. Participantes en las pruebas de liberación.

Nombre y Apellidos	Rol que desempeña
Camilo José Tamayo Morales	Coordinador de la Prueba
Felinda Rosabel León Mendoza	Asesora de Calidad

Ing. Felinda Rosabel León Mendoza
Asesora de Calidad

Ing. Yelenys Almora Gálvez
Jefe de Proyecto

CEGEL

1

Figura 23 Acta de Liberación de Productos de Software por el grupo de calidad Centro CEGEL.
Fuente: elaboración propia

Anexo 4: Acta de Aceptación del Producto por el MINCIN

UCI Universidad de las Ciencias Informáticas

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento del **Convenio de colaboración** el **Centro de Gobierno Electrónico (CEGEL)** y en función de la ejecución del proyecto Personalización del XABAL SIGAP 1.0 para el MINCIN, se hace entrega de los productos que se relaciona a continuación:

- Personalización del XABAL SIGAP 1.0 para el MINCIN
- Manual de Usuario

La Parte Cliente, luego de haber revisado los productos de trabajo que se entregan, determina aceptar los mismos. De igual forma se acuerda con el cliente una vez firmada esta acta continuar con el proceso de capacitación y entrenamiento del cliente con el software y extender el uso del sistema a las provincias.

Y para que así conste se suscribe la presente Acta en La Habana a los 09 días del mes de diciembre de 2019.

Entrega: Centro de Gobierno Electrónico (CEGEL)

Recibe: Ministerio de Comercio Interior (MINCIN)

Nombre y apellidos: Aldis J. Abreu Medina

Nombre y apellidos: Yalina Garbey Rivera

Cargo: Director

Cargo: Directora de Protección al Consumidor en el MINCIN.

Nombre y apellidos: Yelepy Almora Galvez

Cargo: Jefa de Proyecto

11

Figura 24 Acta de Aceptación del Producto por el MINCIN.
Fuente: elaboración propia