Universidad de las Ciencias Informáticas
Facultad 3

Componente para la obtención de información desde el Sistema de Gestión para la Atención a la Población

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

Autor:

Karel Fernández Rojas

Tutor:

MSc. Yordanis García Leiva

Co-tutor

Ing. Robin Sencial Terrero

Junio, 2018

"Año 60 del Triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Karel Fernández Rojas

_____ _____

MSc. Yordanis García Leiva Ing. Robin Sencial Terrero

AGRADECIMIENTOS

Siempre supe que los agradecimientos de mi tesis sería lo más difícil para mí, inicialmente quiero agradecerle a mi tribunal y mi oponente por guiarme en el transcurso de los seminarios de tesis y la pre-defensa, a todos gracias por brindarme su experiencia como profesionales. Además agradecerle a todos mis profesores de la carrera, en especial a mi tutor Yordanis y mi co-tutor Robin, gracias por educarme y transmitirme los conocimientos que hicieron posible que hoy 29 de junio de 2018 se cumpla mi sueño de ser Ingeniero en Ciencias Informáticas.

En segundo lugar doy gracias a mis hermanitos Dalber, Yeider, Luis Angel, Víctor, Raulito y Andriel, y a mis amigos Fernando, Yarini y Leanna, a ustedes gracias por brindarme su linda amistad, gracias a su apoyo logré sobrepasar los obstáculos que fueron apareciendo durante 5 años, los considero mi familia de la universidad, siempre los tendré presente.

También quiero agradecerle a mis vecinos por siempre estar al pendiente de cómo me iban los estudios y darme su apoyo incondicional. A mis tíos, en especial a Thomás, Raúl, Canete, Magdalena y mi abuelita Igdeliza, a ustedes gracias por brindarme su apoyo incondicional principalmente en los momentos más difíciles.

Agradecer además a los padres de mi novia, a ustedes Nancy y José, gracias por confiar en mí y abrirme las puertas de su casa y su corazón.

Llegó el momento de darle las gracias a una personita muy especial para mí, mi futura esposa y madre mis hijos NEIVIS, a ti que más que novia haz sido mi amiga gracias por confiar en mí y darme tu amor, deseo pasar cada día de mi vida a tu lado y que se cumplan nuestros planes futuros, te amo mucho mi tata.

DEDICATORIA

Dedico mi tesis a mi papá Silvino y mi mamá Dulce.

A ti papá, gracias por darme la vida, quiero que sepas que a pesar de que un fatal accidente nos separó, no hay un día de este mundo que deje de pensar en ti, me hubiese encantado que estuvieras presente en este día tan especial, porque sé que tu mayor deseo era que me graduara en esta universidad, a ti papá donde quiera que estés, descansa en paz, ya cumplí contigo, ya tu hijito es Ingeniero en Ciencias Informáticas, te amo mucho.

A ti mamá, gracias por darme la vida, gracias por educarme, gracias por confiar en mí y apoyarme incondicionalmente siempre, gracias por darme fuerzas en todo momento, gracias por preocuparte y tener fe en mí, gracias por ser la mejor madre del mundo, te amo mucho.

RESUMEN

El Sistema de Gestión para la Atención a la Población constituye una aplicación web para la Asamblea Nacional del Poder Popular de la República de Cuba, su propósito es informatizar los procesos del área de atención a la población. La aplicación cuenta con un conjunto de funcionalidades que viabilizan el trabajo de los técnicos y especialistas del área, pero aún no brinda la posibilidad de obtener información oportuna sobre las estadísticas que generan los procesos informatizados a través de este software. La presente tesis de grado tiene como objetivo el desarrollo de un componente que se integre al software antes descrito y brinde la posibilidad a los usuarios de generar reportes estadísticos e informes de forma automática, así como la visualización de datos estadísticos a través de una pizarra de control, que orienten al usuario sobre el comportamiento del proceso de atención a la población por municipio, provincia u otros indicadores procesados a través del sistema.

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 definen indicadores que permiten verificar la relación causa efecto de la variable independiente sobre la dependiente de la investigación.

PALABRAS CLAVES: componente, informes, pizarra de control, reportes estadísticos, Sistema de Gestión para la Atención a la Población

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la República de Cuba la soberanía reside en el pueblo, del cual dimana todo el poder del Estado. Ese poder es ejercido directamente o por medio de las Asambleas del Poder Popular y demás órganos del Estado que de ellas se derivan, en la forma y según las normas fijadas por la Constitución y las leyes (Granma, 2014).

La Asamblea Nacional del Poder Popular (ANPP) es el órgano supremo del poder del Estado. Representa y expresa la voluntad soberana de todo el pueblo (Constitución, 2002). La ANPP en su estructura presenta el Área de Atención a la Población (AAP), la cual tiene como propósito atender a los ciudadanos cubanos o extranjeros que allí se dirijan, con la finalidad de presentar alguna queja, denuncia, solicitud o sugerencia hacia algún Organismo de la Administración Central del Estado (OACE), Asamblea Provincial del Poder Popular (APP) o Asamblea Municipal del Poder Popular de la Habana. El proceso de atención a la población inicia cuando un ciudadano se presenta ante alguno de los especialistas del AAP o se comunica con estos por medio de una carta, correo electrónico u otro medio de comunicación a través de internet.

En la actualidad un grupo de especialistas de la Universidad de las Ciencias Informáticas (UCI) se han dado a la tarea de informatizar el proceso de atención a la población de la ANPP, desarrollando el Sistema de Gestión para la Atención a la Población (SIGAP). Este sistema permite crear un expediente por cada ciudadano que se dirija a la asamblea para presentar un escrito de queja, denuncia, solicitud o sugerencia y archivar en este expediente todos los escritos que en el futuro la persona presente. El SIGAP posibilita también que los especialistas del AAP puedan tener un control sobre el estado de los expedientes, escritos presentados por los ciudadanos a la ANPP, así como de las respuestas recibidas desde los destinatarios a los cuales han sido dirigidos.

A pesar de las facilidades que brinda el SIGAP, los técnicos y especialistas del AAP de la ANPP aún no cuentan con la posibilidad de obtener información oportuna para realizar análisis estadístico por promoventes, provincia, municipio o relacionando otros datos registrados a través del sistema. Todo lo anterior dificulta también la confección del informe mensual que deben entregar al Presidente de la ANPP u otros informes que puedan ser solicitados al AAP en cualquier momento. Por otra parte, el SIGAP solo

procesa los datos recogidos en la creación y tramitación de un expediente, sin dar la posibilidad de visualizar las estadísticas que estos datos brindan una vez que el número de casos procesados aumente.

En la actualidad el análisis estadístico de información se dificulta en el AAP de la ANPP teniendo en cuenta que mensualmente son procesados unos 840 escritos. Esto trae como resultado que, a la hora de calcular las estadísticas generadas durante el proceso de atención a los ciudadanos, los profesionales del área tienen que procesar cada expediente de forma manual, buscando en ellos los escritos procesados en el periodo para el cual se necesita la información estadística, volviéndose esta tarea extensa y propensa a errores.

A partir de la problemática antes descrita se genera la necesidad de desarrollar una investigación que responda al siguiente **problema de investigación**:

¿Cómo obtener información oportuna del proceso de atención a la población desde el SIGAP, que permita a los técnicos y especialistas del AAP de la ANPP, agilizar los análisis estadísticos y la realización de informes?

Tomando en cuenta el problema antes propuesto se define como **objeto de estudio:** los sistemas de gestión de información.

Determinándose como **objetivo general:** desarrollar un componente para la obtención de información desde el SIGAP que permita a los técnicos y especialistas del AAP de la ANPP agilizar los análisis estadísticos y la realización de informes.

Para ello se identifica como **campo de acción:** sistemas de gestión de información para la atención a la población.

Definiéndose como **idea a defender:** el desarrollo de un componente para la obtención de información desde el SIGAP permite a los técnicos y especialistas del AAP de la ANPP agilizar los análisis estadísticos y la realización de informes.

Se desglosan del objetivo general los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación mediante un estudio de los referentes teóricos sobre el desarrollo de aplicaciones informáticas para la gestión del proceso de atención a la población.

- Desarrollar la identificación de los requisitos, análisis, diseño e implementación del componente para la obtención de información desde el SIGAP.
- Validar y verificar el correcto funcionamiento del componente propuesto aplicando métricas, criterios y pruebas de software.

La realización de dichas tareas se sustenta en los siguientes métodos de investigación:

Métodos Teóricos:

Histórico-Lógico: permitió realizar un estudio sobre las principales tendencias de la web respecto a los software existentes para el proceso de atención a la población y la forma en que han ido evolucionando estos.

Analítico-Sintético: posibilitó la realización del estudio teórico de la investigación logrando hacer una síntesis de cada uno de los contenidos estudiados.

Modelación: se empleó para la confección de los prototipos funcionales y la modelación de los artefactos generados por la metodología de desarrollo de software seleccionada.

Métodos Empíricos:

Observación: se aplicó con el propósito de conocer cómo los especialistas y técnicos de la ANPP desarrollan los procesos en el área atención a la población, también posibilitó identificar la necesidad que existe en esta área de informatizar el proceso de generación de reportes estadísticos.

Medición: permitió medir la calidad de la especificación de los requisitos y el grado de ambigüedad de estos, además de obtener una medida de la calidad del diseño para su validación.

Entrevista: fue utilizado para comprender las necesidades del cliente, capturar los requisitos correspondientes al componente y obtener información necesaria para la realización de la investigación.

El contenido de este trabajo consta de tres capítulos, definidos de la siguiente forma:

Capítulo 1: Fundamentación Teórica.

En este capítulo se describen una serie de conceptos relacionados con el objeto de estudio de la presente investigación, así como las características fundamentales de los

sistemas de atención a la población 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 al lenguaje de modelado utilizado. En el capítulo también se describen los patrones de diseño, lenguajes de programación y demás tecnologías utilizadas en la solución propuesta.

Capítulo 2: Análisis, diseño e implementación de la solución propuesta.

En este capítulo se realiza una descripción de las principales características del componente para la obtención de información desde el SIGAP. 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 en el capítulo se encuentran el proceso de captura y validación de los requisitos funcionales (RF) y no funcionales (RnF) con los que debe cumplir el componente propuesto. Además, se muestra la arquitectura de la solución y la aplicación de los patrones de diseño utilizados. Por otra parte, se exponen los estándares de codificación empleados en la disciplina de implementación.

Capítulo 3: Validación de la solución propuesta.

En este capítulo se muestran los resultados obtenidos luego de aplicar las técnicas de validación tanto al diseño del sistema como a la solución desarrollada. Documentándose los resultados alcanzados en cada caso. Para la validación del diseño se aplican métricas y para la validación de la solución se define una estrategia de prueba aplicando los 4 niveles de pruebas con sus respectivos métodos y técnicas. Por otra parte, se realiza una verificación del cumplimiento del objetivo general de la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el capítulo se describen una serie de definiciones relacionadas con el objeto de estudio de la presente investigación, así como las características fundamentales de los sistemas de atención a la población que existen en diversas regiones del mundo incluyendo Cuba. Además, se describe la metodología seleccionada para guiar el proceso de desarrollo de la investigación, junto al lenguaje de modelado utilizado. En el capítulo también se describen los patrones de diseño, lenguajes de programación y demás tecnologías utilizadas en la solución propuesta.

1.2 Conceptos fundamentales asociados al dominio del problema

Para una mejor comprensión del tema de investigación, es necesario dominar las siguientes definiciones:

Atención a la población:

Los espacios de atención son todos aquellos puntos de acceso o canales de comunicación, a través de los cuales las personas pueden participar en el quehacer de los servicios públicos. El marco normativo que rige a los espacios de atención, garantiza el derecho de acceso a la información y a la atención oportuna, sin discriminación de ninguna especie. Esta participación implica el ejercicio de los derechos ciudadanos, el cumplimiento de sus deberes, el acceso a productos y servicios que proveen las instituciones, la recepción de información acerca de programas sociales y la expresión de sus expectativas e intereses a través de reclamos, sugerencias, consultas y opiniones (Contreras, et. al, 2010).

Reporte estadístico:

Documento que analiza a través de cuadros y gráficos estadísticos la evolución de las principales variables de una institución, con el propósito de ayudar a la toma de decisiones empresariales (SNI, 2018).

Pizarra de control (DashBoard):

Es una representación gráfica de los principales indicadores que intervienen en la consecución de los objetivos de negocio. Está orientado a la toma de decisiones para

optimizar la estrategia de la empresa. Un *dashboard* debe transformar los datos en información y esta, en conocimiento para el negocio (Elósegui, 2014).

Promovente:

Acción y efecto de promover. Persona que promueve algo, haciendo las diligencias conducentes para su logro. Término utilizado especialmente en el ámbito jurídico (Lengua, 2018).

Sistema de Gestión para la Atención a la Población:

Es una aplicación web destinada a gestionar el proceso de atención a la población en la Asamblea Nacional del Poder Popular (ANPP) de la República de Cuba. Mediante su uso, los usuarios tienen acceso a funcionalidades que les permite crear un expediente por cada persona natural que se presente al área de la atención a la población de la ANPP a presentar un escrito de queja, denuncia, sugerencia o solicitud. Una vez que las personas se presentan se les crea un único expediente y dentro del mismo se registran todos los escritos que la persona presente a esta institución. La aplicación da cobertura a todo el proceso de tramitación de cada uno de los escritos hasta su cierre (Mejias Cruz, 2018).

1.3 Tendencias actuales

En la actualidad a nivel empresarial, una buena atención al cliente se vuelve un factor clave para viabilizar el éxito y el crecimiento de cualquier negocio. Con el ánimo de apoyar las estrategias corporativas de los servicios, existe la necesidad de crear sistemas que permitan tener diversos medios de contacto con el público (aplicaciones web, carta, correo electrónico u otros medios), para facilitar y agilizar el proceso de atención a la población. Entre los sistemas que existen actualmente en el extranjero para este fin, se encuentran:

Sistema de Peticiones, Quejas, Reclamos y Sugerencias (PQRS): es una herramienta que permite conocer las inquietudes que tienen los grupos de interés para tener la oportunidad de fortalecer el servicio. Su enfoque le permite mejorar la capacidad de registro de las solicitudes, respuestas e interacciones entre los clientes y funcionarios encargados de atender los requerimientos.

Entre otras funcionalidades este sistema permite controlar las peticiones, quejas, reclamos y solicitudes de una entidad por el rastreo de un software**,** además de los informes e

indicadores que genera el sistema. Asimismo, proporciona al ciudadano una herramienta de comunicación con su entidad mediante la conexión a internet (Aranda, 2016).

APROWEB: es una aplicación web fácil de usar para la recepción y seguimiento de quejas. Permite organizar y dar solución a las inquietudes de los clientes. Entre sus funcionalidades destacan (Soft, 2018):

- Permite recolectar quejas por medio de un formulario.
- Puede importar quejas al sistema por medio de un buzón de correo electrónico.
- Puede ingresar quejas capturadas directamente a través del sistema.
- Permite canalizar y direccionar quejas a diferentes miembros.
- Permite agregar notas a cada queja además de registrar fecha, hora y persona que atiende.
- Permite anexar documentos para difundir las soluciones y acciones a tomar.
- Recuerda vía correo electrónico al personal asignado que existen acciones pendientes a realizar.
- Permite establecer un flujo a seguir para la atención de la queja.
- Permite generar una base de conocimiento fácilmente consultable.

A pesar de que APROWEB brinda todas estas funcionalidades, posee licencia propietaria. Esta es difícil de adquirir en Cuba debido a su alto costo y no está en correspondencia con la soberanía tecnológica que tiene establecida el país.

Existen otros sistemas foráneos que también brindan el servicio de atención a los ciudadanos. Tal es el caso de los software utilizados en los ayuntamientos de Palma de Mallorca y Murcia en España y en la secretaría de educación de Colombia. Estos sistemas se centran en recepcionar las quejas, denuncias, solicitudes o sugerencias, mediante formularios web, vía correo o telefónica. Teniendo como desventaja fundamental que son software propietarios (ISOTools, 2017).

En el caso de Cuba, previo al desarrollo del SIGAP se tiene conocimiento de la existencia del software AvilaQuid, desarrollado en la Asamblea Nacional del Poder Popular con el fin de gestionar el proceso de atención a la población en este órgano supremo del poder del Estado. Este sistema tiene como principal desventaja que entre sus funcionalidades no se incluye la generación de reportes estadísticos que le viabilicen el trabajo a los especialistas del área de atención a la población de la ANPP.

Partiendo del análisis realizado de cada uno de los sistemas, y teniendo en cuenta las características principales definidas para dar solución al problema existente, se evidencia la

necesidad de desarrollar un componente que mejore el SIGAP. En la siguiente tabla comparativa se muestra el resultado de dicho análisis:

Tabla 1. Tabla comparativa de las soluciones estudiadas.

Características	AvilaQuid	APROWEB	PQRS
Información oportuna	No	Sí	No
Generación de reportes	No	Sí	No
Análisis estadístico	No	Sí	No
Software propietario	No	Sí	No

Fuente: elaboración propia.

A partir del estudio realizado sobre los diferentes tipos de software que existen en las entidades para informatizar el proceso de atención a la población. El autor de la presente tesis de grado concluye que ninguno de estos sistemas puede utilizarse para dar respuesta a la problemática que dio origen a la investigación, teniendo en cuenta que la mayoría no presentan funcionalidades que permitan obtener información oportuna para la generación de reportes estadísticos. El sistema APROWEB cumple con las características deseadas, pero tiene la desventaja de ser software propietario. Las características de estos sistemas demuestran la necesidad de desarrollar el componente propuesto en la presente tesis, teniendo en cuenta que no pueden ser integrados al SIGAP. Sin embargo, a partir del estudio del software APROWEB se obtuvieron características respecto a la forma en que este software realiza el análisis estadístico de indicadores, con el propósito de ser incorporadas a la solución propuesta, tales como la generación de reportes.

1.4 Metodología de Desarrollo de Software

Una metodología 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 involucrados, junto con prácticas y técnicas recomendadas, las guías de adaptación de la metodología al proyecto y las guías para uso de herramientas de apoyo (Figueroa, et. al, 2008).

Las metodologías se clasifican en tradicionales y ágiles. Estas últimas se basan en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa, permitiendo potenciar aún más el desarrollo de software a gran escala. Dentro de ellas destaca el

Proceso Unificado Ágil (AUP, por sus siglas en inglés), 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 (Rodríguez Sánchez, 2015).

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[1]-DEV v 1.3, conocida como AUP-UCI. La misma establece prácticas centradas en el desarrollo de productos y servicios de calidad (Rodríguez Sánchez, 2015).

1.4.1 Metodología AUP-UCI

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 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 (Rodríguez Sánchez, 2015). A continuación, se describen cada una de las fases de la variación AUP para la UCI.

Inicio: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación de este. En la 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 (Rodríguez Sánchez, 2015).

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, obtienen los requisitos, se elaboran 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 (Rodríguez Sánchez, 2015).

[1] *Capability Maturity Model Integration*

Cierre: en esta fase se analizan los resultados del proyecto relacionados con su ejecución y se realizan las actividades formales de cierre del proyecto (Rodríguez Sánchez, 2015).

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, pero a un nivel más atómico. 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) (Rodríguez 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 (Rodríguez Sánchez, 2015).

- **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 modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información (Rodríguez Sánchez, 2015).

- **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 (Rodríguez Sánchez, 2015).

- **Escenario No 4:** proyectos que no modelen negocio solo pueden modelar el sistema con 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 (Rodríguez 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. Estas disciplinas se desarrollan en la fase de ejecución y pueden estar organizadas en iteraciones, con el propósito de obtener resultados incrementales.

A partir del análisis realizado sobre la variación de la metodología AUP para la UCI, el autor de la presente tesis decide organizar el proceso de desarrollo del componente que da cumplimiento al objetivo general de la investigación a partir de las fases y disciplinas que propone esta variación de la metodología. La decisión se adopta teniendo en cuenta que el desarrollo del componente responde a un proyecto de la red de centros de la universidad. En el capítulo 2 se describen cómo son aplicadas las disciplinas y el escenario utilizado por el autor.

1.5 Lenguajes de programación

De los lenguajes de programación que existen para desarrollar aplicaciones orientadas a la web se encuentran dos grupos fundamentales de acuerdo con la arquitectura Cliente/Servidor, 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 (Lenguaje de Marcados para Hipertextos, por sus siglas en inglés *HyperText Markup Language).* 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. Para el desarrollo de la solución propuesta se hace necesario utilizar los siguientes lenguajes.

1.5.1 JavaScript ES5

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 (Venemedia, 2014).

TypeScript 2.9.1

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 común. Es llamado también Superset[2] de Javascript, lo que significa que si el navegador está basado en Javascript, este nunca llegará a saber que el código original fue realizado con *TypeScript* y ejecutará el Javascript como lenguaje original (Caceres, 2017).

1.5.2 HTML 5

HTML es el lenguaje que se utiliza para crear todas las páginas web de internet. Es reconocido universalmente y permite publicar información de forma global (Gauchat, 2012).

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

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 (Hogan, 2011).

1.5.3 CSS 3.0

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 y su 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 infinidad de dispositivos diferentes (Gauchat, 2012).

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 (Gauchat, 2012). 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 (Gauchat, 2012).

El autor de la presente investigación decide utilizar para el desarrollo del componente propuesto los lenguajes de programación antes descritos, teniendo en cuenta que esta solución debe responder a los mismas tecnologías y lenguajes en los que está desarrollado el software SIGAP.

1.6 Herramientas de modelado

Las herramientas de modelado de sistemas informáticos, son herramientas que se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. Permiten crear un "simulacro" del sistema, a bajo costo y riesgo mínimo. A bajo costo porque, es un conjunto de gráficos y textos que representan el sistema, pero no son el sistema físico real. Además minimizan los riesgos, porque los cambios que se deban realizar (por errores o cambios en los requerimientos), se pueden realizar más fácil y rápidamente sobre el modelo que sobre el sistema ya implementado. Además, permiten concentrarse en ciertas características importantes del sistema, prestando menos atención a otras. Los modelos

resultantes, son una forma de determinar si están representados todos los requerimientos del sistema, y también saber si el analista comprendió qué hará el sistema (Definición de Herramienta de modelado, 2018)**.**

1.6.1 Balsamiq Mockups 3.5.15

Es una herramienta para crear prototipos o bocetos. Esta tiene varias ventajas, ya que posee una interfaz fácil de usar y es instalable tanto en Windows como Linux. Además permite escoger entre objetos prediseñados como: barras de estado, menús y barras de progreso. También, permite exportar el diseño realizado a formato PNG, PDF e incluso al portapapeles. Por otra parte, posibilita incrustar diseños en páginas web o en informes de errores (ISDI, 2014).

1.6.2 Visual Parading 5.0

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite modelar todos los tipos de 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 (Rondón, et. al, 2011).

1.7 Marco 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, 2017). Para el desarrollo de la solución propuesta se hace necesario utilizar Angular 4 como marco de trabajo.

1.7.1 Angular 4

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 enriquecido. La palabra clave que permite dicho HTML enriquecido en Angular es "directiva", que no es

otra cosa que código JavaScript que mejora el HTML. También promueve y usa patrones de diseño de software básicamente en el patrón Modelo Vista Controlador (Programación de Interfaces de Usuario, 2018)**.**

El autor decide utilizar este marco de trabajo con el propósito de lograr la compatibilidad del componente con SIGAP.

1.7.2 Spring Boot 1.5.6

Es un sub-proyecto de Spring, el mismo facilita la creación de proyectos con framework Spring eliminando la necesidad de crear largos archivos de configuración XML (siglas en inglés de *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 main (Elkan, 2017).

1.8 Entorno de Desarrollo Integrado (IDE)

Un IDE, 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 (GUI). Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (fergarciac, 2013). Para el desarrollo de la solución propuesta se hace necesario utilizar Netbeans como IDE de desarrollo.

1.8.1 Netbeans 8.1

Es un entorno de desarrollo gratuito y de código abierto. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones web, o para dispositivos móviles. Da soporte a las siguientes tecnologías: Java, PHP, Groovy, C/C++ y HTML5. Además, puede instalarse en varios sistemas operativos: Windows, Linux o Mac OS (Genbetadev, 2017).

Para el desarrollo de la solución informática propuesta, se selecciona la herramienta Netbeans en su versión 8.1, teniendo en cuenta que es libre, y se integra perfectamente con el lenguaje de programación a utilizar. Además, el autor posee dominio sobre esta herramienta. Otra razón por la cual utilizar Netbeans, es que el componente propuesto responde a una solución que también está implementada sobre ese IDE.

1.9 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 (Power Data, 2016). Para el desarrollo de la solución propuesta se hace necesario utilizar el PostgreSQL como sistema gestor de bases de datos.

1.9.1 PostgreSQL 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 que se ha ganado una sólida reputación de fiabilidad e integridad de datos. Se ejecuta en los principales sistemas operativos que existen en la actualidad como Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows (Microbuffer, 2011).

El autor decide utilizar este SGBD teniendo en cuenta que el componente tiene que ser compatible con el software SIGAP, el cual también utiliza PostgreSQL.

1.10 Patrones de diseño

Un patrón de diseño es una solución a un problema recurrente en el diseño de software (Guerrero, et. al, 2013).
Cada patrón describe un problema que ocurre una y otra vez en un entorno, además describe el núcleo de la solución del problema de forma que se pueda utilizar esta solución, sin hacerlo nunca de la misma manera (Pressman, 2010).

Entre los patrones de diseño se encuentran los agrupados en el grupo de los GOF[3]:

Patrones GOF:

Describen soluciones simples y eficaces a problemas específicos en el diseño de software orientado a objetos (Gamma, et. al, 1994).

[3] GOF (Patrones Banda de los Cuatro, por sus siglas en inglés de *Gang of Four*)

Son combinaciones de componentes, casi siempre clases y objetos que por experiencia se sabe que resuelven ciertos problemas de diseño comunes (Braude, 2003).

Para el desarrollo del componente propuesto se utiliza de los patrones GOF, el Método de fabricación.

Método de fabricación (*Factory Method*): define una interfaz para crear un objeto, pero deja a las subclases decidir qué clase instanciar (Ochoa, 2018). Es utilizado cuando:

- Una clase no puede anticipar la clase de objetos que debe crear.
- Una clase quiere que sus subclases especifiquen las clases que ella debe crear.
- Utiliza una clase constructora abstracta con unos cuantos métodos definidos y otros abstractos.
- Las clases principales en este patrón son el creador y el producto.

Otros de los patrones utilizados en el desarrollo del componente son:

Observador (*Observer*): consiste en definir una dependencia entre objetos de forma que cuando un objeto cambie su estado, todos los objetos dependientes son notificados y cambian su estado automáticamente (Instinto Binario, 2017).
Inyección de dependencias (*Dependency Injection)*: consiste en colocar dentro de un objeto otros que puedan cambiar su comportamiento, sin que esto implique volver a crear el objeto (Pacheco, 2017).
Repositorio (*Repository)*: es un mediador entre el dominio de la aplicación y los datos que le dan persistencia. Con este planteamiento podemos pensar que el usuario de este repositorio no necesitaría conocer la tecnología utilizada para acceder a los datos, sino que le bastaría con saber las operaciones que nos facilita este “mediador”, el repositorio (Roche, 2013).
En el capítulo 2 el autor describe cómo los patrones definidos anteriormente son aplicados en el diseño del componente propuesto.

1.11 Pruebas

Uno de los elementos principales para certificar la calidad de una aplicación informática lo constituye el resultado de las pruebas que se practican. Un control y una gestión de calidad implementados de forma adecuada, especialmente durante el proceso de desarrollo del software, aumentan la calidad del sistema final, reducen los costos de avance y acortan el tiempo necesario para el desarrollo. Para determinar el nivel de calidad se deben efectuar

medidas o pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones principales del sistema (Pressman, 2010).

Pruebas de software: comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo (PMOinformatica, 2018).

Etapas, tipos y técnicas de pruebas de software

Las pruebas de software están compuestas por cuatro niveles o etapas, dos tipos y cuatro técnicas**,** ver figura 1:

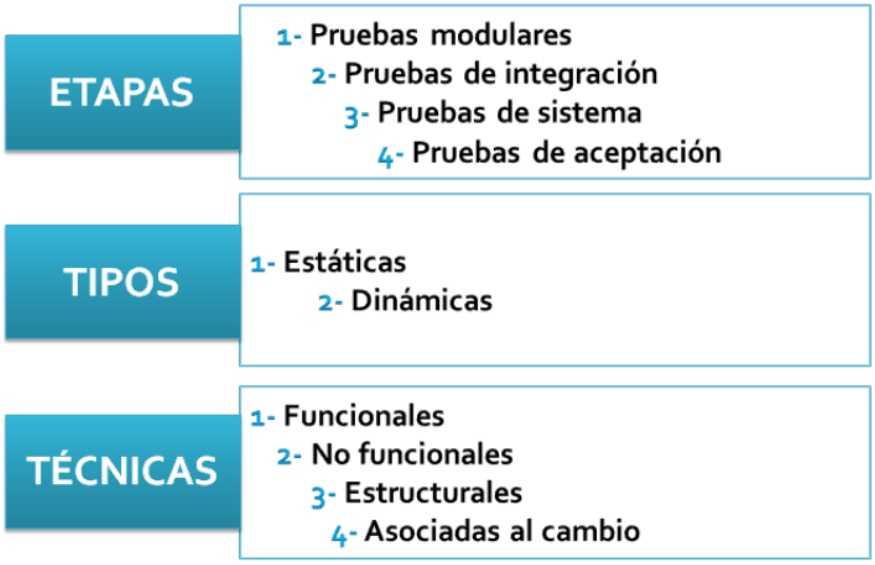


Figura 1. Niveles, tipos y técnicas de pruebas**.**

Fuente: (Campos Chiu, 2015).

A continuación se muestra una breve descripción de cada nivel (etapa) de prueba:

Pruebas Unitarias o de Componente: este tipo de pruebas son ejecutadas normalmente por el equipo de desarrollo, consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Adicionalmente, las pruebas sobre componentes unitarios, suelen denominarse pruebas de módulos o pruebas

de clases, siendo la convención definida por el lenguaje de programación la que influye en el término a utilizar (Abad Londoño, 2005).

Pruebas de Integración: este tipo de pruebas son ejecutas por el equipo de desarrollo y consisten en la comprobación de que elementos del software que interactúan entre sí, funcionan de manera correcta (Abad Londoño, 2005).

Pruebas de Sistema: este tipo de pruebas deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo, una buena práctica en este punto corresponde a la tercerización de esta responsabilidad. La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto. Los casos de prueba a diseñar en este nivel de pruebas, deben cubrir los aspectos funcionales y no funcionales del sistema. Para el diseño de los casos de prueba en este nivel, el equipo debe utilizar como bases de prueba entregables tales como: requerimientos iniciales, casos de uso, historias de usuario, diseños, manuales técnicos y de usuario final (Abad Londoño, 2005).

Pruebas de Aceptación: independientemente de que se haya tercerizado el proceso de pruebas y así la firma responsable de estas actividades haya emitido un certificado de calidad sobre el sistema objeto de prueba, es indispensable, que el cliente designe a personal que haga parte de los procesos de negocio para la ejecución de pruebas de aceptación, es incluso recomendable, que los usuarios finales que participen en este proceso, sean independientes al personal que apoyó el proceso de desarrollo (Abad Londoño, 2005).

Teniendo en cuenta la metodología seleccionada para guiar el desarrollo del componente propuesto por la presente investigación, a continuación, se describen las disciplinas de pruebas de software que esta plantea:

Pruebas Internas: son realizadas por sus propios desarrolladores, verificando el resultado de la implementación, probando cada construcción según sea necesario, así como las versiones finales a ser liberadas. Los artefactos necesarios para la realización de estas pruebas son los casos de pruebas (Rodríguez Sánchez, 2015).

Pruebas de Liberación: son diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (Rodríguez Sánchez, 2015).

Pruebas de Aceptación: son las únicas pruebas que son realizadas por los clientes. Consiste en comprobar si el producto está listo para ser implantado para el uso operativo en el entorno del usuario (Rodríguez Sánchez, 2015).

Para el desarrollo de la solución propuesta se adopta como estrategia de prueba aplicar los cuatro niveles y realizando pruebas funcionales con las técnicas que estas incluyen. En el capítulo 3 se realiza una descripción más amplia de la estrategia de prueba definida por el autor.

1.12 Conclusiones parciales

El desarrollo del marco teórico referencial relacionado con los términos atención a la población, reporte estadístico y pizarra de control, facilita una mejor comprensión del objeto de estudio de la presente investigación. Además, el estudio de soluciones nacionales y extranjeras dirigidas a informatizar procesos de atención a la población en diferentes partes del mundo, contribuyó a la definición de algunas de las funcionalidades del componente propuesto. Por otra parte, el análisis de las características de la variación de la metodología AUP para la UCI, las herramientas a utilizar, así como los patrones de diseño, niveles y métodos de pruebas de software, fundamenta la correspondencia de la selección de cada uno de estos elementos con la implementación y validación de la presente tesis.

CAPÍTULO 2: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En el capítulo se realiza una descripción de las principales características del componente para la obtención de información desde el SIGAP. 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) con los que debe cumplir el componente propuesto. Además, se muestra la arquitectura de la solución y la aplicación de los patrones de diseño utilizados. Por otra parte, se exponen los estándares de codificación empleados en la disciplina de implementación.

2.2 Disciplina de requisitos

Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema (Sommerville, 2016).

La gestión de requisitos es un conjunto de actividades que ayudan al equipo del proyecto a identificar, controlar y rastrear los requisitos y los cambios a estos en cualquier momento mientras se desarrolla el proyecto (Pressman, 2010).

2.2.1 Técnicas para la captura de requisitos

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de software obtiene las necesidades que debe cubrir el sistema. Esta actividad consiste en descubrir o averiguar lo que se debe construir (Admin. de requerimientos, 2014). Para identificar los requisitos que responden al componente propuesto, el autor decide utilizar la técnica de entrevista.

Entrevista: se aplicó a los técnicos y especialistas del área de atención a la población de la Asamblea Nacional del Poder Popular, teniendo en cuenta que estos constituyen los usuarios para los cuales se desarrolla la solución. En el proceso de selección de los entrevistados se tuvo en cuenta el nivel de integración de los mismos con el proyecto según el puesto que estos ocupan.

2.2.2 Definición de los requisitos funcionales

Los requisitos funcionales definen el funcionamiento del sistema y varían según el tipo de solución a implementar. Facilitan un entendimiento de los procesos a desarrollar, que se comprenda con profundidad el problema en cuestión y una mejor identificación de las funcionalidades que serán implementadas (Pressman, 2010).

La siguiente tabla muestra el listado de los requisitos funcionales identificados por el autor**:**

Tabla 2. Requisitos funcionales.

No.	Nombre	Descripción
RF1	Mostrar cantidad de casos por clasificación.	Permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por la clasificación de estos (quejas, denuncia, solicitud o sugerencia). Debe permitir mostrar los datos por un rango de fecha.
RF2	Cantidad de casos por vía utilizada.	Permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por la vía que estos llegan (carta, llamada telefónica, correo electrónico, entrevista u otro medio desde internet). Debe permitir mostrar los datos por un rango de fecha.
RF3	Cantidad de casos por provincia del promovente.	Permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por la provincia del promovente. Debe permitir mostrar los datos por un rango de fecha.
RF4	Cantidad de casos por municipio del promovente.	Permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por el municipio del promovente. Debe permitir mostrar los datos por un rango de fecha.

RF5	Cantidad de casos por promovente.	Permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por promovente. Debe permitir mostrar los datos por un rango de fecha.
RF6	Cantidad de casos pendientes a respuesta.	Permite visualizar la cantidad de casos atendidos en el área de atención a la población que se encuentren pendientes a una respuesta. Debe permitir mostrar los datos por un rango de fecha.
RF7	Visualizar estado permanente de los escritos.	Permite mostrar en la pantalla principal el estado de los escritos registrados durante un mes en el SIGAP.
RF8	Visualizar estado permanente de los asuntos.	Permite mostrar en la pantalla principal el estado de los asuntos registrados durante un mes en el SIGAP.
RF9	Generar informe.	Generar informe con el resultado estadístico de los reportes.

Nota: fuente de elaboración propia.

2.2.3 Definición de los requisitos no funcionales

Los requisitos no funcionales (RnF) son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo sobre el proceso de desarrollo y estándares. A menudo son aplicados en su totalidad al sistema y normalmente apenas se aplican a características o servicios individuales del sistema (Pressman, 2010).

En la siguiente tabla ilustrativa muestra los requisitos no funcionales:

Tabla 3. Requisitos no funcionales.

Requisitos no funcionales	
RnF	**Usabilidad**
1	La interfaz del componente a desarrollar debe ser sencilla.

2	El componente debe mostrar facilidad para interactuar y entender las actividades que realiza el usuario.
3	El componente debe funcionar sobre las plataformas Windows y Linux.
	Funcionabilidad
4	El componente debe contar con campos obligatorios para garantizar un manejo adecuado de la información introducida por el usuario.
5	El componente no permite la entrada de datos incorrectos.
	Mantenibilidad
6	El componente debe ser fácil a la hora de analizar el código fuente pues el mismo está comentado y la documentación está redactada en un lenguaje fácil de entender.
	Requerimientos de Software y Hardware
7	Máquinas clientes: Servidor web Apache 2.4. Procesador Core 2 duo a 2.0 GHz como mínimo 2 Gb de memoria RAM, además de una tarjeta de red.
8	Navegador Web, Chrome o Firefox v50 o superior
9	Máquinas para servidor aplicación: • JRE[4] 1.8 mínimo update 60 • Mongo DB 3.2.11 • PostgreSQL 9.6 • Zookeeper 3.4.9 • Apache Kafka 2.11 • Mínimo 4 Gb de RAM • Core i3 • Almacenamiento 1 Tb

Nota: fuente de elaboración propia.

[4] Java Runtime Environment

2.2.4 Validación de los requisitos

Con el objetivo de asegurar que el software está de acuerdo con su especificación, se comprueba que el sistema cumple con los requisitos funcionales y no funcionales que se han especificado utilizando las siguientes técnicas de validación de requisitos:

- **Revisiones formales de los requisitos:** se realizaron revisiones formales de cada requisito, por parte del cliente y el equipo de desarrollo, validando que la interpretación de cada una de las descripciones no sea ambigua, ni presenten omisiones o errores.
- **Construcción de prototipos:** se realizó un modelo ejecutable utilizando el componente para que los clientes puedan experimentar con él y determinar si cumple sus necesidades.

Para medir la calidad de la especificación de los requisitos de software se aplicó la métrica Calidad de la especificación (CE). El empleo de esta métrica permite obtener un alto nivel de entendimiento y precisión de los requisitos, para llegar a ello, se debe primeramente calcular el total de requisitos de software como se muestra a continuación:

Nr: total de requisitos de software.
Nf: cantidad de requisitos funcionales.
Nnf: cantidad de requisitos no funcionales.

$$Nr = Nf + Nnf$$

Sustituyendo los valores en la ecuación, se obtiene:

$$Nr = 9 + 10$$

$$Nr = 19$$

Finalmente, para calcular la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación:
Nui: número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

$$Q1 = Nui / Nr$$

Para sustituir los valores de las variables en la ecuación se tuvo en cuenta que, de los requisitos especificados para el desarrollo del componente, dos de ellos causaron contradicción (RF7 y RF8) en sus interpretaciones. Por tanto, la variable Q1 obtiene el siguiente valor:

$$Q1 = 17/ 19$$

Q1 = 0.89

Es importante aclarar que mientras más cerca de 1 está el valor de Q1, menor es la ambigüedad. Teniendo en cuenta el resultado anterior, igual a 0.89, se concluye que el 89% de los requisitos es entendible. Los dos requisitos identificados como ambiguos fueron modificados y validados para garantizar su correcta comprensión, llegando al resultado ideal de Q1=1.

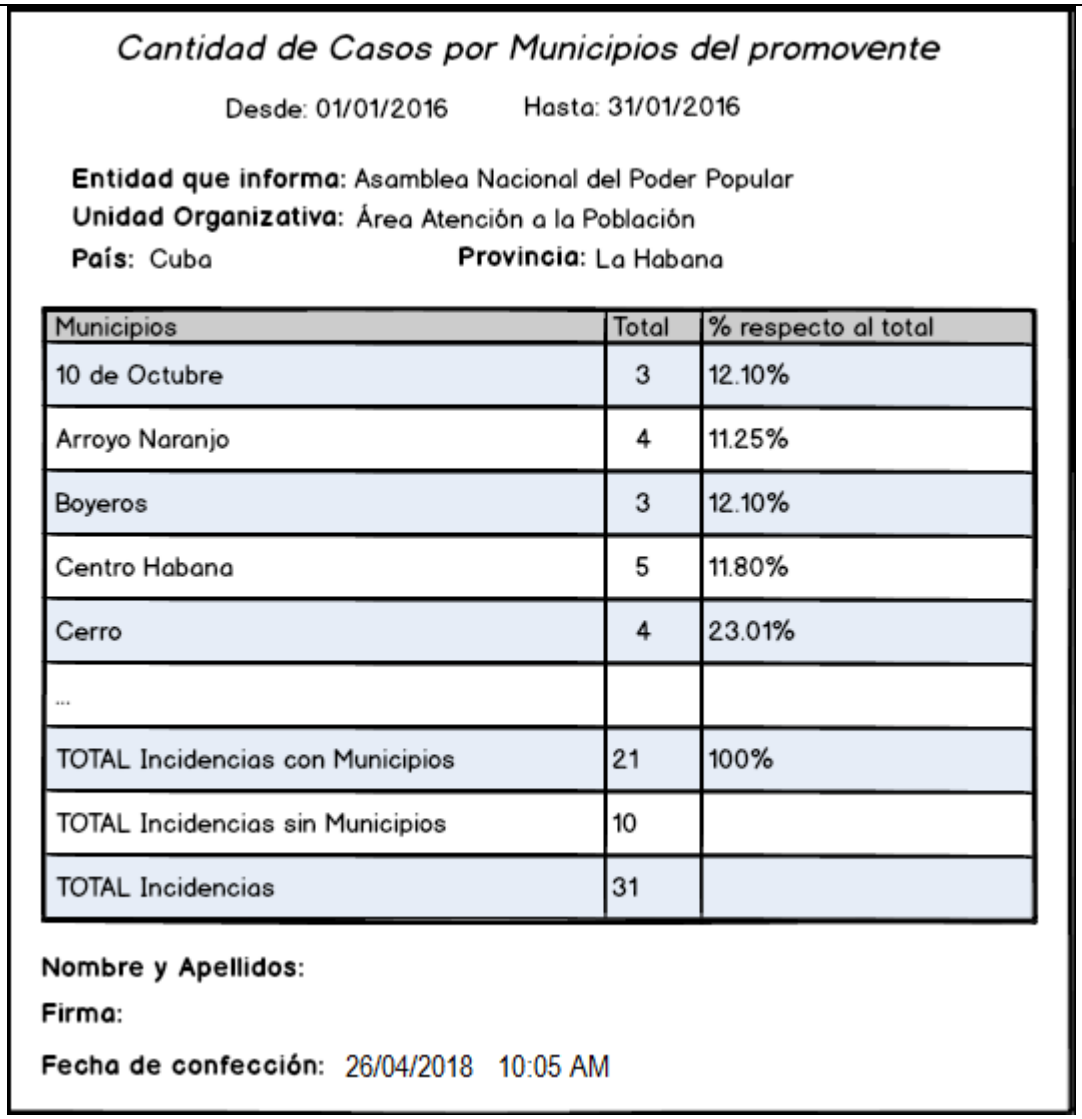
2.2.5 Historias de usuario

Las historias de usuario (HU) son una forma de administrar con rapidez 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 (Tabares García, et. al, 2015).

A continuación, se describen dos historias de usuario de prioridad alta presentes en el desarrollo del componente propuesto, para consultar el resto (ver Anexo 1):

Tabla 4. Historia de Usuario Cantidad de casos por municipio del promovente.

Historias de Usuario	
Número: 4	**Nombre de la HU:** cantidad de casos por municipio del promovente.
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: alto	**Tiempo real:** 10 días
Descripción: permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por el municipio del promovente. Debe permitir mostrar los datos por un rango de fecha.	
Observaciones:	
Prototipo de interfaz:	



Cantidad de Casos por Municipios del promovente

Desde: 01/01/2016 Hasta: 31/01/2016

Entidad que informa: Asamblea Nacional del Poder Popular
Unidad Organizativa: Área Atención a la Población
País: Cuba **Provincia:** La Habana

Municipios	Total	% respecto al total
10 de Octubre	3	12.10%
Arroyo Naranjo	4	11.25%
Boyeros	3	12.10%
Centro Habana	5	11.80%
Cerro	4	23.01%
...		
TOTAL Incidencias con Municipios	21	100%
TOTAL Incidencias sin Municipios	10	
TOTAL Incidencias	31	

Nombre y Apellidos:
Firma:
Fecha de confección: 26/04/2018 10:05 AM

Nota: fuente de elaboración propia.

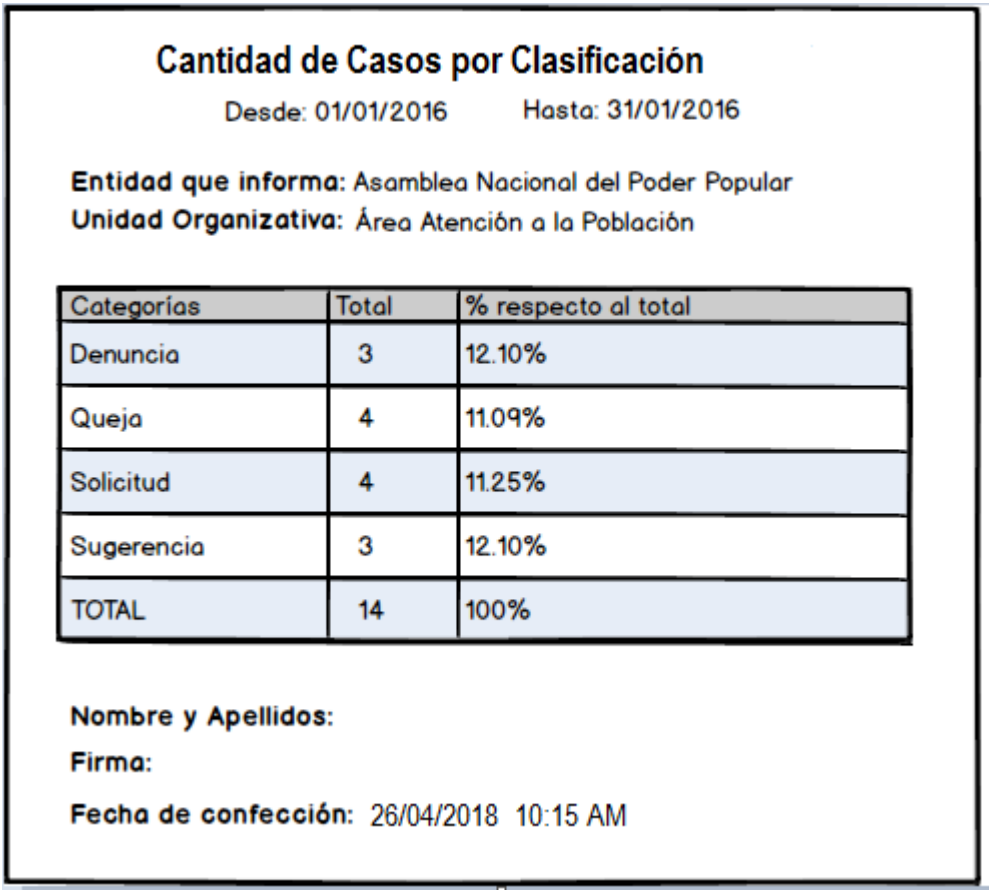
Tabla 5. Historia de Usuario Generar informe.

Historias de Usuario	
Número: 1	**Nombre de la HU:** mostrar cantidad de casos por clasificación
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 15 días
Riesgo en desarrollo: alto	**Tiempo real:** 15 días

Descripción: permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por la clasificación de estos (quejas, denuncia, solicitud o sugerencia). Debe permitir mostrar los datos por un rango de fecha.

Observaciones:

Prototipo de interfaz:



Cantidad de Casos por Clasificación

Desde: 01/01/2016 Hasta: 31/01/2016

Entidad que informa: Asamblea Nacional del Poder Popular
Unidad Organizativa: Área Atención a la Población

Categorías	Total	% respecto al total
Denuncia	3	12.10%
Queja	4	11.09%
Solicitud	4	11.25%
Sugerencia	3	12.10%
TOTAL	14	100%

Nombre y Apellidos:
Firma:
Fecha de confección: 26/04/2018 10:15 AM

Nota: fuente de elaboración propia.

2.3 Disciplina de análisis y diseño

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

2.3.1 Diseño arquitectónico

El diseño arquitectónico garantiza cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global del mismo. En el modelo del 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, 2016).

La solución propuesta tiene como base la arquitectura del SIGAP. En la figura 2 se representan los elementos de esta arquitectura y las partes donde incide el componente propuesto. Es importante especificar que el SIGAP utiliza una arquitectura separada en *frontend* y *backend*. 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, que pueden ser de muchas y variadas formas, y los transforma ajustándolos a las especificaciones que demanda el *backend* para poder procesarlos, 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 ambas partes 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 Object Notation*).

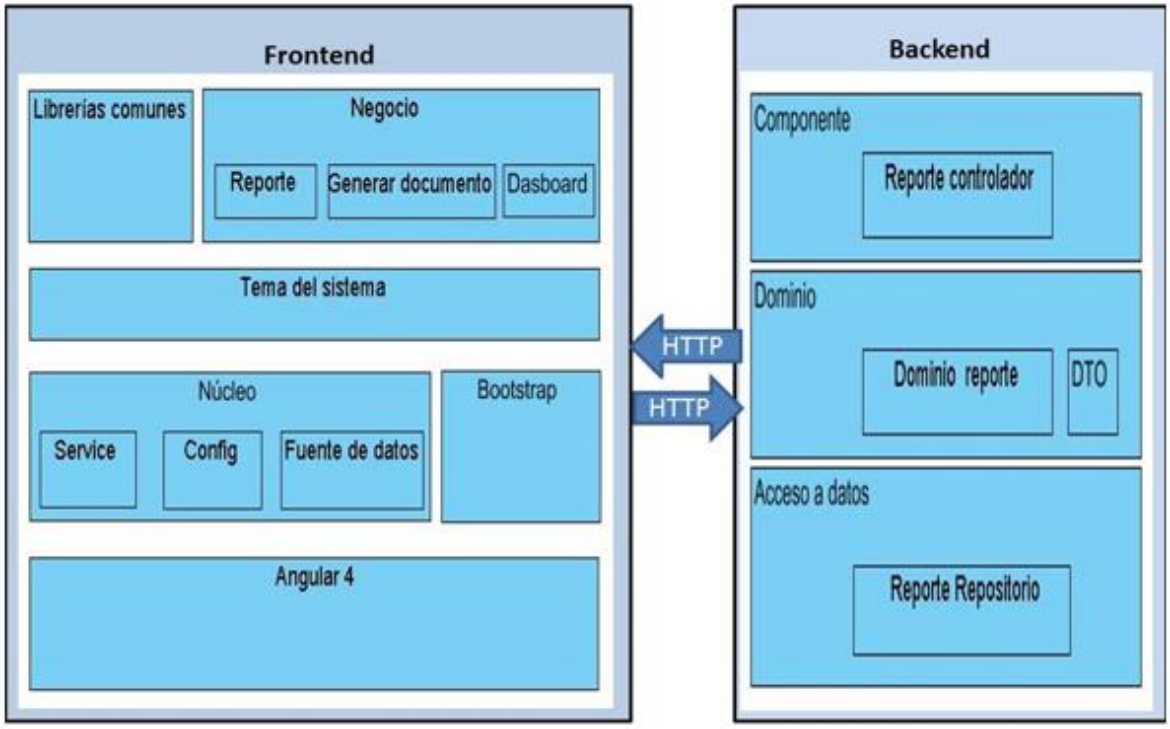


Figura 2. Arquitectura del SIGAP.

Fuente: elaboración propia

A partir del diagrama de la figura 2, en la parte del *frontend* el componente propuesto incide en los elementos del negocio. Este componente es implementado sobre la arquitectura *frontend* del SIGAP a través del uso del patrón arquitectónico Modelo Vista Controlador (MVC).

El patrón MVC separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

- El modelo contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La vista, o interfaz de usuario, compone la información que se envía al cliente y los mecanismos de interacción con éste.
- El controlador, actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno (Servicio de Informatica de ASP.NET MVC 3 Framework, 2017).

En la figura 3 se muestra una descripción más detallada de cómo se representa en el *frontend* del SIGAP la arquitectura del componente propuesto, a partir del uso del patrón arquitectónico MVC. En la figura el modelo está compuesto por el archivo *reporte.model.ts,* el cual contiene la información que se le solicita al usuario para confeccionar el reporte, ejemplo: fecha inicio, fecha fin, municipio, provincia o tipo de reporte. Por otra parte, la vista se compone por el archivo *reporte.component.html*, este representa la información visible al usuario e interactúa con funciones específicas del controlador. El controlador está compuesto por el archivo *reporte.component.ts,* encargado de interactuar con el modelo y mostrar la información a la vista.

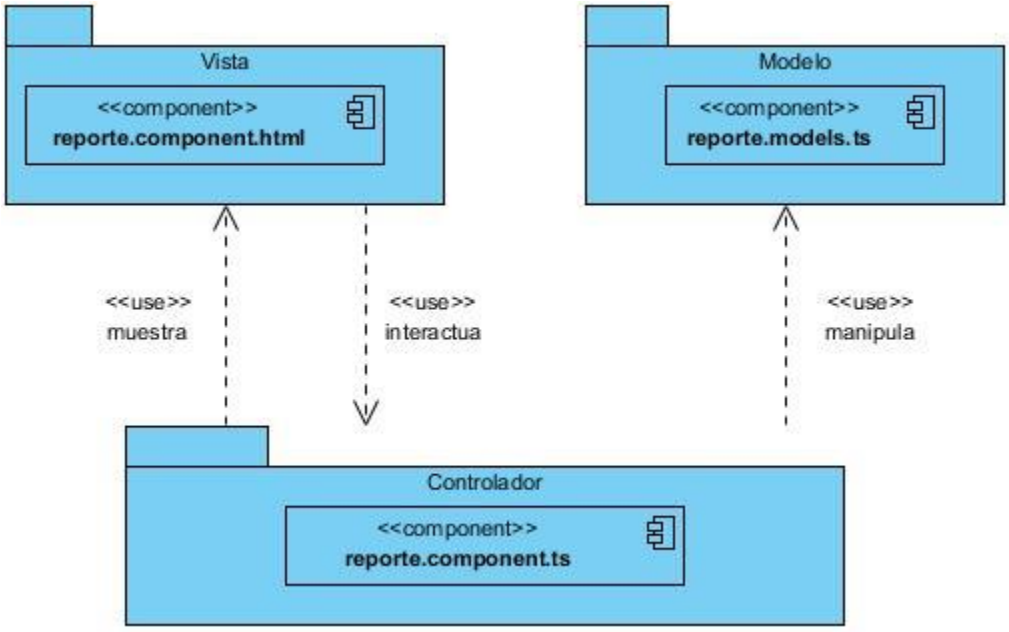


Figura 3. Arquitectura *frontend* del componente propuesto.

Fuente: elaboración propia

A partir del diagrama de la figura 2, en la parte del *backend* el componente propuesto incide en las capas componente, servicios y acceso a datos. Este componente es implementado sobre la arquitectura *backend* del SIGAP a través del uso del patrón arquitectónico N Capas.

El patrón arquitectónico N Capas es una extensión del patrón Capas tradicional (este ayuda a estructurar las aplicaciones que se pueden descomponer en grupos de subtareas en la que cada grupo de subtareas está en un nivel particular de abstracción). En el nivel más alto y abstracto, la vista de arquitectura lógica de un sistema puede considerarse como un conjunto de servicios relacionados agrupados en diversas capas (Escalante, 2014).

En la figura 4 se muestra una descripción más detallada de cómo se representa en el *backend* del SIGAP la arquitectura del componente propuesto, a partir del uso del patrón arquitectónico N Capas. En la figura el componente *ReporteControlador* de la capa Controlador usa al componente *ReporteDominio* de la capa Servicios, el cual usa al componente de la capa de Acceso a Datos *ReporteRepositorio* y los tres a la vez están orientados a la capa Dominio.

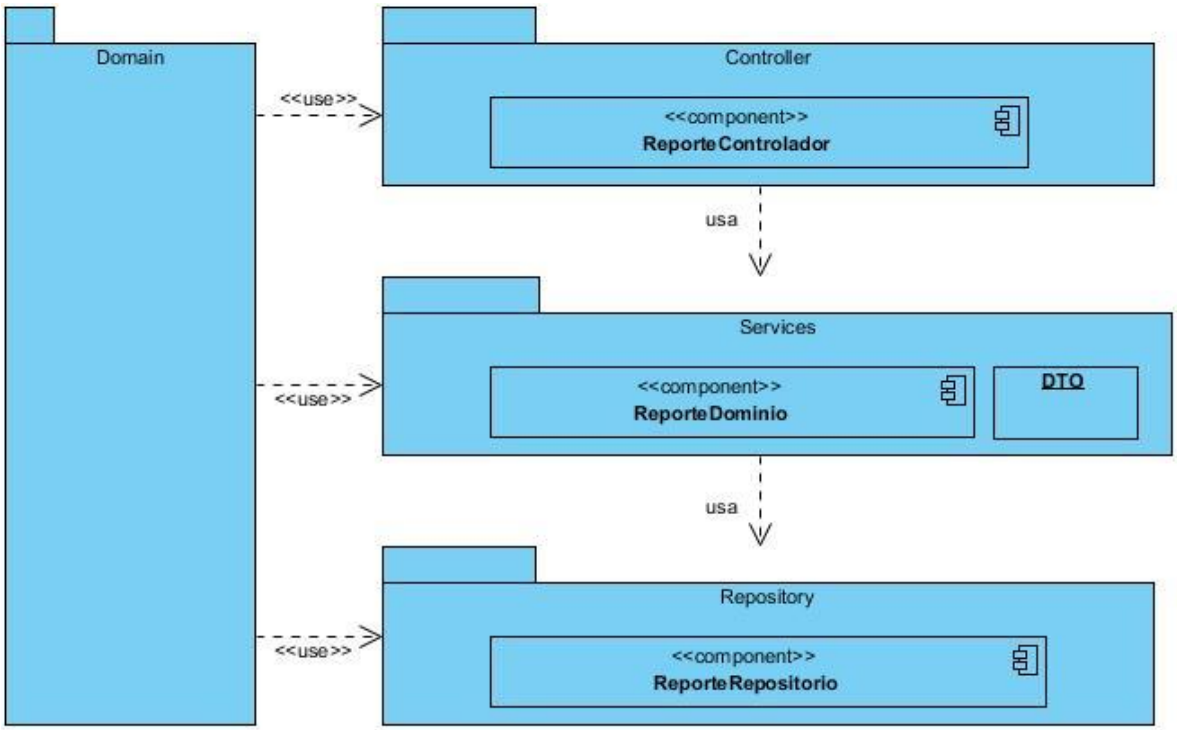


Figura 4. Arquitectura *backend* del componente propuesto.

Fuente: elaboración propia

2.3.2 Patrones de diseño

Para el diseño del componente propuesto se utilizaron un conjunto de patrones, los cuales constituyen la base para solucionar problemas comunes en el desarrollo de software. A continuación, se describen los patrones empleados.

- Observador

Este patrón es empleado en varias partes de la implementación del componente, ejemplo en la declaración de la variable observable, de la funcionalidad para la obtención del reporte Casos por Promovente.

```
promoventes$: Observable<Promovente[]>;
```

Figura 5. Declaración de la variable observable.

Fuente: elaboración propia

La figura 6 ilustra a través de un fragmento de código de la funcionalidad obtención del reporte Casos por Promovente, la aplicación del patrón Observador. Este patrón permite, a partir de la declaración de la variable mostrada en la figura 5, que cuando la lista de promoventes se modifique el componente este observando el cambio y se actualice visualmente.

```
this.promoventes$ = this.store.select(fromPromoventes.getAllPromoventes);
this.promoventes$.subscribe((promoventes: Promovente[]) => {
  const data = [];
  for (let i = 0; i < promoventes.length; i++) {
    const promovente: Promovente = promoventes[i];
    data.push({
      value: promovente.idpersonanatural,
      ''  + promovente.primerapellido + (promovente.segundoapellido ? ''+ promovente.segundoapellido : '') });  }
  this.promoventes = data;
});
label: promovente.primernombre + (promovente.segundonombre ? '' + promovente.segundonombre : '') +
''  + promovente.primerapellido + (promovente.segundoapellido ? ''+ promovente.segundoapellido : '') });  }
this.promoventes = data;
});
```

Figura 6. Aplicación del patrón Observador.

Fuente: elaboración propia

- Inyección de dependencia

La aplicación del patrón inyección de dependencia se evidencia en el código de la figura 7. Este código representa el constructor utilizado en la inyección de servicios útiles para el trabajo con los reportes. En este caso se tiene un enumerador con las listas de reportes y el servicio *UtilReporteService* que permite codificar el enumerador y listarlo en *"this.reportes"* y utilizarlo en el componente "*select"* de la aplicación.

```
constructor(private utils: UtilReporteService,
  private reporteService: ReporteService,
  private router: Router,
  private dsResolver: DsResolver,
  private municipioService: MunicipioService,
  private nomencladorService: NomencladorService,
  private route: ActivatedRoute,
  private store: Store<fromRoot.State>) {
  this.reportes = this.utils.stringEnumToKeyValue(ReportesEnums);
}
```

Figura 7. Aplicación del patrón inyección de independencia.

Fuente: elaboración propia

- **Repositorio**

La aplicación del patrón repositorio se evidencia en el en el código de la figura 8. Este código representa cómo el usuario a partir del trabajo con el objeto *escritoRepository*, no necesita conocer la tecnología utilizada para acceder a los datos, sino que utiliza una capa de abstracción para acceder a estos, mediante las operaciones que provee el repositorio.

```
@Override
public Page<Escrito> listarEscritosAReasignar(EscritoForm escritoForm, Pageable pageable) {
    Nomencladorbase estadoCerradoEscrito = nomencladorService.nomencladorDadodescripcionAndTipo(NomencladoresValues.ESTADO_CERRADO.getDescipcion(),
            TipoNomenclador.NESTADO_ESCRITO);
    return escritoRepository.findAll(EscritoSpecs.listarEscritosAReasignar(escritoForm, estadoCerradoEscrito), pageable);
}
```

Figura 8. Aplicación del patrón Repositorio

Fuente: elaboración propia

- **Método de fabricación:**

La aplicación del patrón Método de fabricación se evidencia en el código de la figura 9, donde se define una interfaz para crear un objeto, dejando a las subclases decidir qué clase instanciar.

```
@Override
public Agregable cantidadDeCasosPorProvincia(ReporteDTO reporteDTO) {
    CriteriaBuilder builder = entityManager.getCriteriaBuilder();
    CriteriaQuery<Object[]> query = builder.createQuery(Object[].class);
    Root<Escrito> root = query.from(Escrito.class);
    Join<Object, Object> expediente = root.join( s: "expediente");
    Join<Object, Object> personanatural = expediente.join( s: "personanatural", JoinType.LEFT);
    Root<Nomencladorbase> nbRoot = query.from(Nomencladorbase.class);

    query
            .multiselect(nbRoot.get("descripcion"),
                    builder.count(root.get("id")))
            .where(
                    builder.between(root.get("fechaescrito"), reporteDTO.obtenerFechaInicial(), reporteDTO.obtenerFechaFinal()),
                    builder.equal(personanatural.get("direccionPostal").get("idprovincia"), nbRoot.get("id")))
            .groupBy(personanatural.get("direccionPostal").get("idprovincia"), nbRoot.get("descripcion"));
    TypedQuery<Object[]> q = entityManager.createQuery(query);
    List<Object[]> resultList = q.getResultList();

    return new ContenidoAgregable(resultList.stream().map(contenidoConverter::with).collect(Collectors.toList()));
}
```

Figura 9. Aplicación del patrón Método de fabricación.

Fuente: elaboración propia

2.4 Disciplina de implementación

En esta fase a partir de los resultados del Análisis y el Diseño se construye la solución que da lugar a la presente investigación.

2.4.1 Estándares de codificación

Los estándares de codificación constituyen buenas prácticas o conjunto de reglas no formales que han ido surgiendo en las comunidades de desarrolladores de software con el paso del tiempo, con el propósito de facilitar el entendimiento del código en las tareas de mantenibilidad de un software.

Los estándares de codificación utilizados en el desarrollo del componente fueron definidos por el equipo de desarrollo del SIGAP. A continuación, se describen los mismos.

- Los nombres de las clases serán con mayúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán de igual forma, en la Figura 10 se evidencia el uso de este estándar en la clase *ReportesService*.

```
@Service
public class ReportesServiceImpl implements ReportesService {
```

Figura 10. Representación del estándar de codificación 1.

Fuente: elaboración propia

- Los nombres de los métodos serán con minúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán con mayúscula, ver Figura 11.

```
@Override
public Agregable cantidadDeCasosPorClasificacion(ReporteDTO reporteDTO) {
```

Figura 11. Representación del estándar de codificación 2.

Fuente: elaboración propia

- Los identificadores para las variables y los parámetros serán con letras en minúsculas y en caso de ser un nombre compuesto las siguientes palabras se escribirán de igual forma, en la Figura 12 se evidencia el uso de este estándar de codificación.

```
private String fechainicial;
private String fechafinal;
private Long idprovincia;
```

Figura 12. Representación del estándar de codificación 3.

Fuente: elaboración propia

- La clase controladora del componente comienza con el prefijo Reportes y luego el nombre de la clase (*ReportesServiceImpl.java*), ver Figura 13.

```java
@Service
public class ReportesServiceImpl implements ReportesService {

    @PersistenceContext
    EntityManager entityManager;
    private final ContenidoConverter contenidoConverter;
    private final NomencladorService nomencladorService;

    public ReportesServiceImpl(ContenidoConverter contenidoConverter, NomencladorService nomencladorService) {
        this.contenidoConverter = contenidoConverter;
        this.nomencladorService = nomencladorService;
    }
```

Figura 13. Representación del estándar de codificación 4.

Fuente: elaboración propia

- Los nombres de variables o funciones deben ser lo suficientemente descriptivos, sin exceder de 30 caracteres, ver figuras de la 10 a la 13.
- Todas las funciones deben tener comentarios, que describan su propósito (Javadoc[5]). En la Figura 14, se evidencia el uso de este estándar en el método *cantidadDeCasosPorClasificación ()*.

```java
/**
 * Mostrar cantidad de casos por clasificación.
 *
 * @param reporteDTO
 * @return ResponseEntity<List < Contenido>>
 */
@RequestMapping(path = ENTITY_URI + "/clasificacion", method = RequestMethod.POST, consumes = "application/json")
public ResponseEntity<List<Contenido>> cantidadDeCasosPorClasificacion(@RequestBody ReporteDTO reporteDTO) {
    try {
        return ResponseEntity.ok()
                .body(reportesService.cantidadDeCasosPorClasificacion(reporteDTO).getContenido());
    } catch (Exception ex) {
        return ResponseEntity.badRequest()
                .headers(HeaderUtil.badRequestAlert( mensaje: "Error obteniendo el reporte")).build();
    }
}
```

Figura 14. Representación del estándar de codificación 6.

Fuente: elaboración propia

[5] Es una utilidad de Oracle para la generación de documentación de interfaz gráficas de aplicaciones en formato HTML a partir de código fuente Java.

2.5 Descripción del sistema

Una vez concluida la disciplina de implementación se logra un componente para la obtención de información desde el SIGAP caracterizado por brindar las posibilidades de:

- Obtener los siguientes reportes estadísticos:
  - Cantidad de casos por clasificación: este reporte muestra a partir de un rango de fecha seleccionado, la cantidad de casos procesados por el Área de Atención a la Población de la Asamblea Nacional del Poder Popular, según la clasificación del escrito (queja, denuncia, solicitud o sugerencia).
  - Cantidad de casos por vía utilizada: este reporte muestra a partir de un rango de fecha seleccionado, la cantidad de casos procesados por el Área de Atención a la Población de la Asamblea Nacional del Poder Popular, según la vía utilizada para hacer la entrega del escrito, la cual puede ser por medio de correo postal, correo electrónico, teléfono, entrevista u otra vía a través de internet.
  - Cantidad de casos por provincia del promovente: este reporte muestra a partir de un rango de fecha seleccionado, la cantidad de casos procesados por el Área de Atención a la Población de la Asamblea Nacional del Poder Popular organizados por las provincias los promoventes de los escritos.
  - Cantidad de casos por municipio del promovente: este reporte muestra a partir de un rango de fecha y una provincia seleccionada, la cantidad de casos procesados por el Área de Atención a la Población de la Asamblea Nacional del Poder Popular organizados por los municipios de los promovente de los escritos.
  - Cantidad de casos por promovente: este reporte muestra a partir de un rango de fecha seleccionado, la cantidad de casos procesados por el Área de Atención a la Población de la Asamblea Nacional del Poder Popular organizados por cada promovente.
  - Cantidad de casos pendientes a respuesta: este reporte muestra a partir de un rango de fecha seleccionado, la cantidad de escritos tramitados por el Área de Atención a la Población de la Asamblea Nacional del Poder Popular y que se encuentran pendiente a respuesta por parte de la entidad a la cual fue tramitado hacia la ANPP.

La Figura 15 muestran la funcionalidad generar reporte estadístico, desde la cual se obtienen los reportes antes descritos. Por otra parte, la Figura 16 muestra una vista del reporte Cantidad de casos por clasificación, generado a partir del componente propuesto.

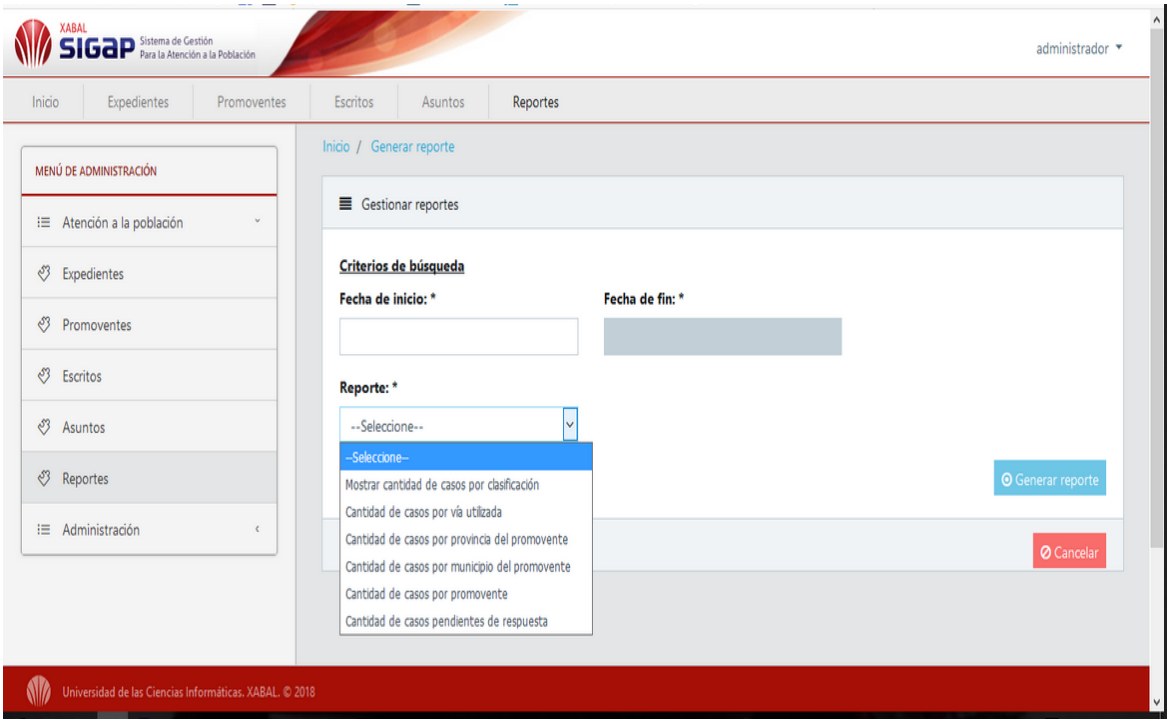
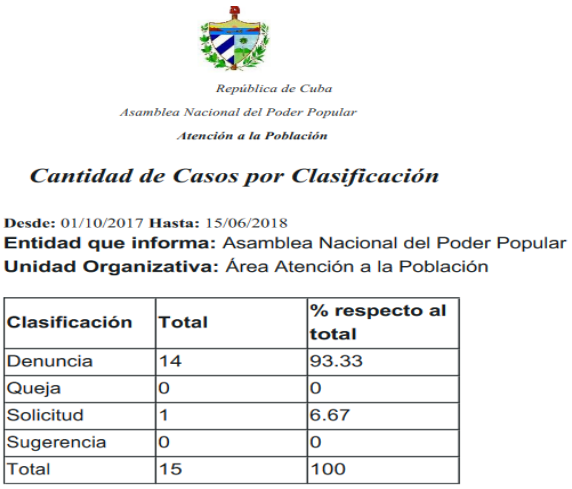


Figura 15. Componente para generar reportes.

Fuente: elaboración propia



República de Cuba

Asamblea Nacional del Poder Popular

Atención a la Población

Cantidad de Casos por Clasificación

Desde: 01/10/2017 **Hasta:** 15/06/2018

Entidad que informa: Asamblea Nacional del Poder Popular

Unidad Organizativa: Área Atención a la Población

Clasificación	Total	% respecto al total
Denuncia	14	93.33
Queja	0	0
Solicitud	1	6.67
Sugerencia	0	0
Total	15	100

Figura 16. Cantidad de casos por clasificación.

Fuente: elaboración propia

- Visualizar estado permanente de los escritos y asuntos procesados a través del SIGAP. Esta funcionalidad, permite mediante una pizarra de control mostrar en tiempo real la cantidad de escritos y asuntos procesados por el sistema, que se encuentran en los estados al límite, en término o fuera de término para las respuestas. La Figura 17 muestra una vista de cómo queda este elemento del componente incorporado al SIGAP.

Figura 17. Pizarra de control.

Fuente: elaboración propia

- Generar informe. Esta funcionalidad permite generar un informe predeterminado sobre el cual se cargan automáticamente los datos estadísticos y permite editar nuevas informaciones que deseen incorporar al informe.

2.6 Conclusiones parciales

El empleo de la metodología AUP en su variación para la UCI en función de describir el proceso de desarrollo del componente propuesto, permitió organizar el desarrollo de la solución y generar los artefactos necesarios. Por otra parte, la aplicación de los patrones arquitectónicos MVC para la arquitectura del *frontend* y N-Capas para la arquitectura del *backend*, el uso de patrones de diseño y estándares de codificación garantizaron la obtención de una solución con poca dependencia entre clases, flexible al mantenimiento y

a la introducción de cambios. Por tanto, el componente obtenido extiende las funcionalidades del SIGAP sin afectar el funcionamiento de este sistema.

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 tanto al diseño del sistema como a la solución desarrollada. Documentándose los resultados alcanzados en cada caso. Para la validación del diseño se aplican métricas y para la validación de la solución se define una estrategia de prueba aplicando los 4 niveles de pruebas con sus respectivos métodos y técnicas. Por otra parte, se realiza una verificación del cumplimiento del objetivo general de la investigación.

3.2. Validación del diseño

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

3.2.1 Relaciones entre clases (RC)

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase (Pressman, 2002).

A continuación, se exponen los pasos que se llevaron a cabo para aplicar la métrica a todas las clases del componente propuesto en la presente investigación:

- Determinar la Cantidad de Relaciones de Uso (CRU) que poseen las clases a medir.
- Calcular el promedio de las CRU.
- 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 6.

Tabla 6. 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

Nota: Fuente (Lorenz, et. al, 1994)

En la Figura 18 se muestra el resultado de aplicar la métrica RC:

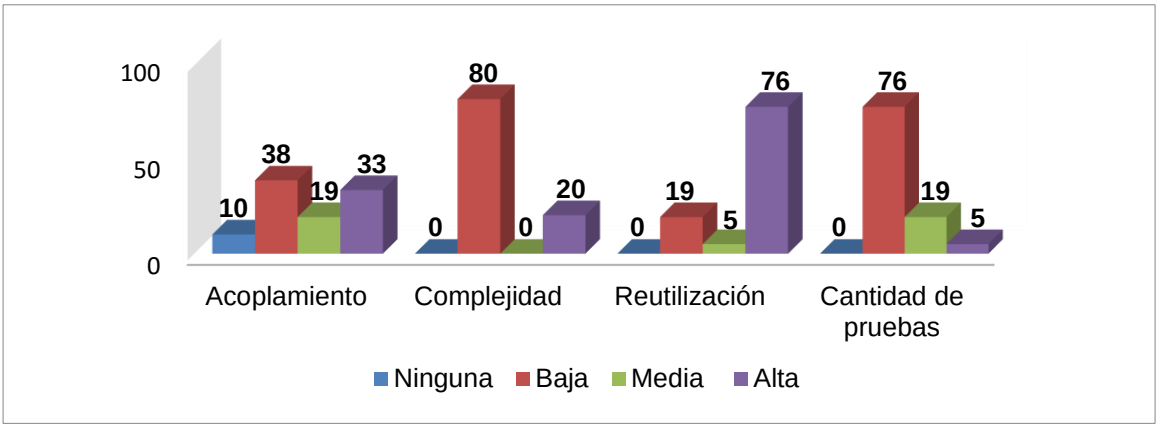


Figura 18. Representación en (%) de los resultados de la aplicación de la técnica RC.

Fuente: elaboración propia

Acoplamiento: según los resultados que se muestran, el 10% de las clases no posee relaciones de uso y el 38 % posee un acoplamiento bajo, el 19% un acoplamiento medio y el 33 % un acoplamiento alto, por lo que la mayoría de las clases poseen valores de acoplamiento nulo, bajos y medio, validando una realización correcta del diseño.

Complejidad de mantenimiento: según los resultados que se muestran en la figura anterior, el 80% de las clases presentan una complejidad de mantenimiento baja, demostrando que son de fácil soporte.

Reutilización: según los resultados que se muestran en la figura, el 76% de las clases tiene un grado alto de reutilización.

Cantidad de pruebas: luego de aplicar la métrica se obtuvo que el 76% de las clases poseen un grado bajo de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software.

Teniendo en cuenta lo anterior se concluye que el diseño del componente propuesto en la presente investigación alcanzó resultados positivos durante la evaluación de la métrica. Se obtuvo un 38% de bajo acoplamiento, 80% de baja complejidad de mantenimiento, el 76% presenta un bajo grado de esfuerzo destinado a pruebas y una reutilización alta. Estos datos muestran que las clases poseen bajo acoplamiento, complejidad de mantenimiento, esfuerzo para realizar pruebas y en efecto presentan un alto grado de reutilización.

3.2.2 Tamaño Operacional de Clases (TOC)

La métrica TOC fue aplicada 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, 2002).

A continuación, se explican 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.
- 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 7.

Tabla 7. 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

Nota: Fuente (Lorenz, et. al, 1994)

En la Figura 19 se muestra el resultado de aplicar la métrica TOC:

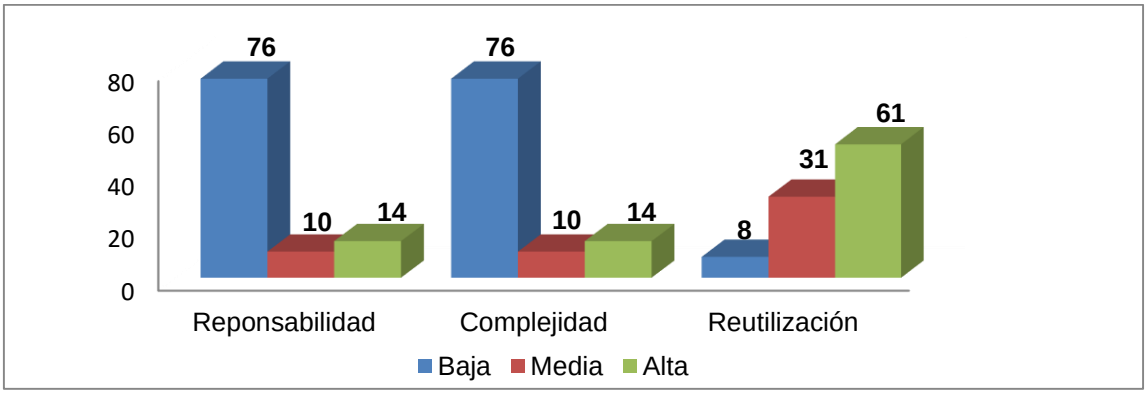


Figura 19. Representación en (%) de los resultados de la aplicación de la técnica TOC

Fuente: elaboración propia

Responsabilidad: después de aplicar la métrica, se obtuvieron resultados satisfactorios que reflejan una responsabilidad baja con un valor del 76%.

Complejidad de implementación: luego de haber realizado la medición de la métrica, se obtuvieron resultados positivos ya que la complejidad de las clases es baja en un 76%.

Reutilización: se obtuvieron valores que según muestra la gráfica de la figura anterior la reutilización se comporta en un nivel alto con un 61%.

Teniendo en cuenta lo anterior se concluye que el diseño del componente propuesto en la presente investigación alcanzó resultados positivos durante la evaluación de la métrica. Se obtuvo un 76% de baja responsabilidad de las clases y complejidad de implementación respectivamente, mientras que la reutilización es alta para un 61%. Estos datos muestran que las clases tienen poca responsabilidad, lo cual amplía su reutilización y facilita la implementación.

3.3. Pruebas de Software

Las pruebas de software comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo.
Los niveles de pruebas de software desarrollados como parte del proceso de validación del componente propuesto en la presente investigación son: unidad, integración, sistema y aceptación.

Las pruebas a nivel de unidad se realizaron con el propósito de medir la calidad del código del componente. Las pruebas a nivel de integración y de sistema se realizaron aplicando el método de caja negra y las técnicas partición de equivalencia y valores límites. Las pruebas de aceptación se realizaron con el cliente, con el propósito que este validara la utilidad del componente. A continuación, se describen las pruebas realizadas en cada uno de estos niveles y los resultados obtenidos en las mismas.

3.3.1 Pruebas a nivel de unidad

La prueba de unidad se concentra 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, 2005).

Método de caja blanca

El método de caja blanca, en ocasiones llamada pruebas de cristal, es un método que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de esta prueba se derivan casos de pruebas que:

- Garantizan que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
- Ejercitan los lados verdaderos y falsos de todas las decisiones lógicas.
- Ejecutan todos los bucles en sus límites y dentro de sus límites operacionales.
- Ejercitan estructuras de datos internos para asegurar su validez
  (Pressman, 2010).

Para aplicar este método se empleó la técnica de la ruta básica. A continuación, se muestra la aplicación de la técnica sobre la funcionalidad *cantidadDeCasosPorClasificacion*, perteneciente a la clase *ReportesService*. La selección de esta porción de código se realizó teniendo en cuenta que el mismo responde a una de las principales funcionalidades del componente. La complejidad ciclomática obtenida con la aplicación de la técnica indica la cantidad de caminos independientes a recorrer para probar el código analizado. Es decir, se obtiene el número de casos de pruebas a desarrollar para la validación del método de la Figura 20. En esta se muestra la numeración de los nodos definidos en cada porción del código.

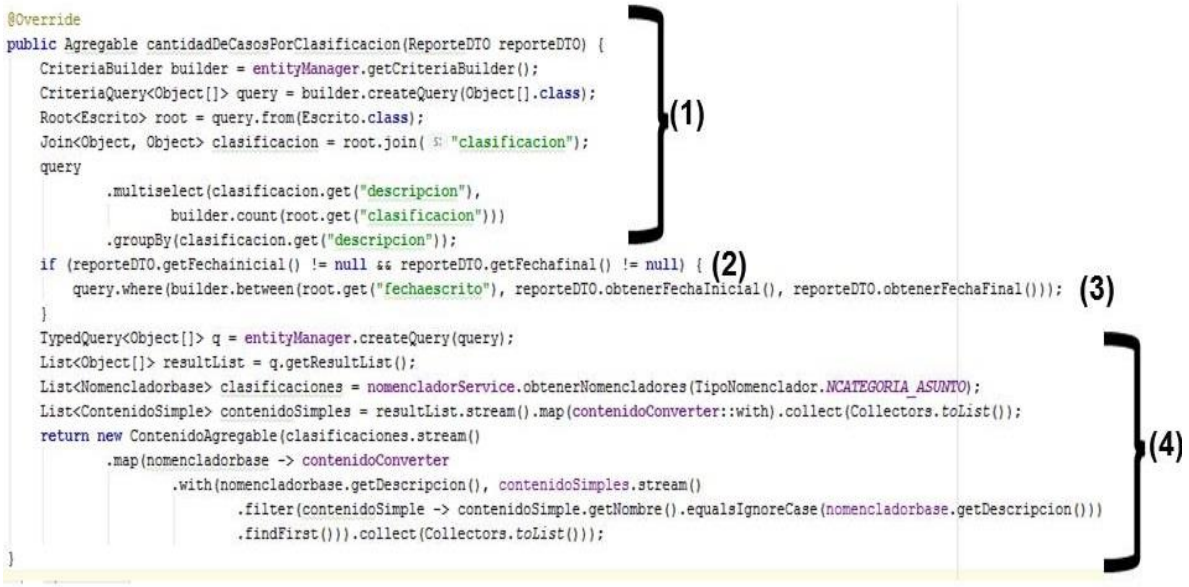


Figura 20. Método c*antidadDeCasosPorClasificacion* de la clase *ReporteSercicelmpl.*

Fuente: elaboración propia.

A continuación, se detallan los pasos que se utilizaron al aplicar la técnica ruta básica:

1. Confeccionar el grafo de flujo: usando el código de la Figura 17 se realizó la representación del grafo de flujo, el cual describe un flujo de control lógico y está compuesto por los siguientes elementos:
   - Nodos de gráfica de flujo: son círculos que representan una o más instrucciones procedimentales.
   - Aristas o enlaces: 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 Figura 21 se presenta el grafo de flujo obtenido:

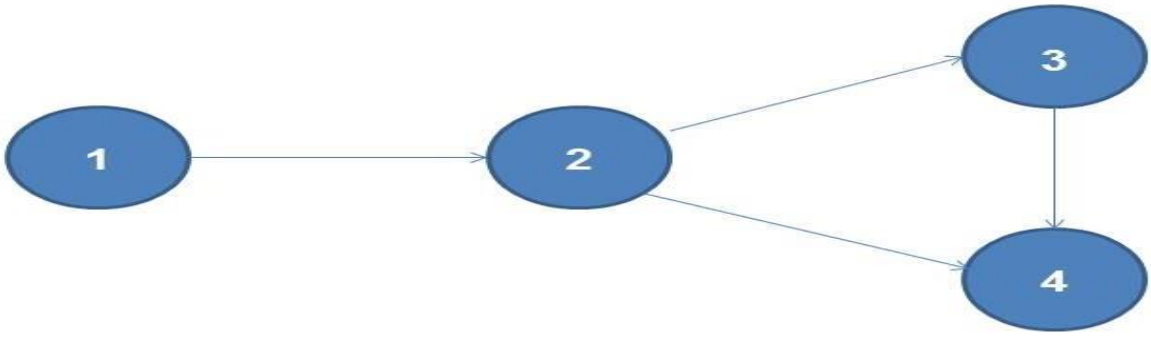


Figura 21. Grafo de la ruta básica del método *cantidadDeCasosPorClasificacion*.

Fuente: elaboración propia.

2. Calcular la complejidad ciclomática: proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado al menos una vez. La complejidad ciclomática se calcula mediante las tres formas siguientes:

- El número de regiones del gráfico de flujo (Pressman, 2010). El gráfico de flujo tiene dos regiones.
- **V (G) = E – N + 2**, donde E es el número de aristas del grafo de flujo y N es el número de nodos del mismo (Pressman, 2010).
  V (G) = E – N + 2 = 4 aristas – 4 nodos + 2 = 2
- **V (G) = P + 1**, donde P es el número de nodos predicados[6] contenidos en el gráfico de flujo (Pressman, 2010).
  V (G) = 1 nodo predicado + 1 = 2

1. Determinar un conjunto básico de rutas linealmente independientes: el valor de V (G) indica el número de rutas linealmente independientes de la estructura de control del programa, por lo que se definen los 2 caminos obtenidos:
   Ruta Básica 1: 1-2-3-4
   Ruta Básica 2: 1-2-4

[6] Cada nodo que contiene una condición y está caracterizado porque dos o más aristas emergen de él.

2. Obtención de casos de pruebas: cada ruta independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. En este caso se obtuvieron 2 caminos básicos, que dan lugar a la confección de igual número de casos de pruebas, para aplicar las pruebas a este método. A continuación, se muestran los casos de pruebas:

Tabla 8. Caso de prueba de caja blanca para el camino básico #1.

Caso de prueba Camino Básico #1	
Descripción: este camino permite que el método *cantidadDeCasosPorClasificacion* se ejecute cuando los campos fecha inicial y fecha final no presentan valores nulos.	
Entrada	Un valor de Fecha inicial y un valor de Fecha final.
Resultado	Muestra la cantidad de casos por clasificación que se encuentran en el rango de fechas especificados.
Condiciones	Deben estar especificadas ambas fechas y la fecha de inicio debe ser menor igual que la fecha final.

Nota: fuente de elaboración propia.

Tabla 9. Caso de prueba de caja blanca para el camino básico #2.

Caso de prueba Camino Básico #1	
Descripción: este camino permite que el método *cantidadDeCasosPorClasificacion* se ejecute cuando los campos fecha inicial y fecha final sean nulos.	
Entrada	Ninguna
Resultado	Muestra todos los casos por clasificación que se encuentran en el sistema.
Condiciones	Los campos fecha inicial y fecha final deben estar vacíos.

Nota: fuente de elaboración propia.

Descripción de la ejecución de los casos de prueba

Para ejecutar cada caso de prueba se realizaron pruebas manuales al código probando cada camino descrito a través de los casos. En el caso de prueba # 1 se introdujo una fecha inicial y una fecha final, mostrándose en el reporte los datos para el rango de fecha especificado, esto evidencia que el caso de prueba se ejecutó satisfactoriamente. En el caso de prueba # 2 no se introducen datos en los campos de fecha y se obtiene un reporte con todo el historial de datos existentes en el sistema, esto evidencia que el caso de prueba # 2 también se ejecutó satisfactoriamente.

Una vez ejecutados los dos casos de pruebas obtenidos a través de la aplicación de la técnica ruta básica, se concluye que los mismos fueron probados satisfactoriamente, demostrando que todas las rutas de este código se ejecutaron al menos una vez.

3.3.2 Pruebas a nivel de integración y sistema

Las pruebas 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, 2005).

Por otra parte, las pruebas 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, 2005).

Método de caja negra

Las pruebas de caja negra, también denominadas, pruebas de comportamiento, se concentran en los requisitos funcionales del software. Es decir, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. La prueba de caja negra no es una opción frente a las técnicas de caja blanca. Es, en cambio, un enfoque complementario que tiene probabilidades de describir una clase diferente de errores de los que se descubrirán con los métodos de caja blanca (Pressman, 2005).

Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

- 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 (Pressman, 2005).

Para desarrollar el método de caja negra existen varias técnicas, entre ellas están (Pressman, 2010):

Técnica de la partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Técnica del análisis de valores límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Con el propósito de evaluar la integración del componente con el SIGAP y el comportamiento de este una vez integrado al sistema. Se realizaron las pruebas a nivel de integración y sistema, aplicando el método de caja negra con el uso de las técnicas partición de equivalencia y análisis de valores límites. A continuación, en la figura 19 se muestran los resultados de la aplicación del método, ejecutado por especialistas del equipo de calidad de CEGEL en un total de tres iteraciones.

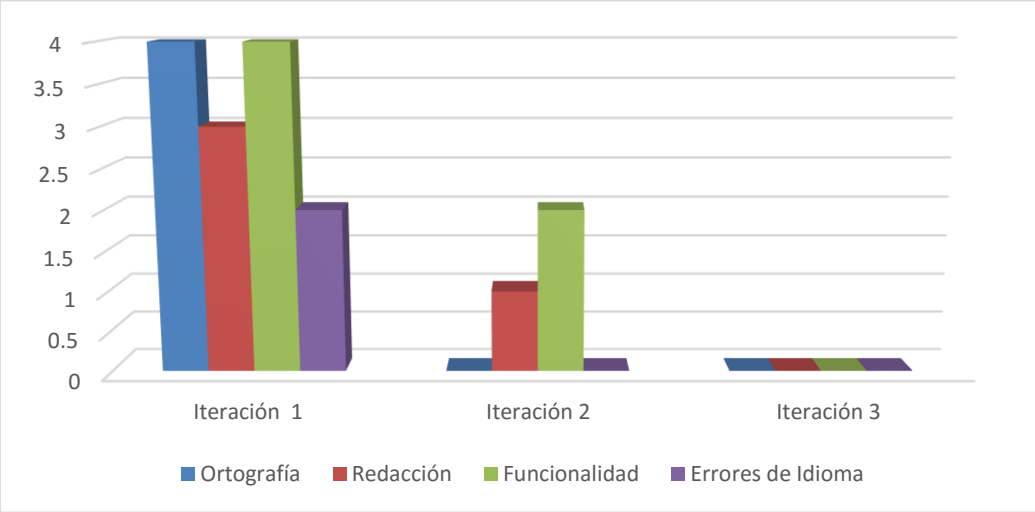


Figura 22. No conformidades detectadas al aplicar el método de caja negra.

Fuente: elaboración propia.

Como muestra la figura 22, en la primera iteración se detectaron un total de 13 No Conformidades (NC), clasificadas en 4 de ortografía, 3 de redacción, 4 de funcionalidad y 2 de errores de idioma. Luego en una segunda iteración persistieron 1 de redacción y 2 de funcionalidad. En la tercera iteración los resultados fueron satisfactorios, obteniéndose cero NC, generándose así el Acta de Liberación Interna de Productos de Software (ver Anexo 2).

3.3.3 Pruebas a nivel de aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Rodríguez, 2015).

Para la aplicación de esta prueba, se confeccionó un caso de prueba de aceptación por cada HU. A continuación, se muestra el caso de prueba de aceptación de la HU: Mostrar cantidad de casos por clasificación, teniendo en cuenta que esta HU responde a una de las principales funcionalidades del componente.

Tabla 10. Caso de prueba de Aceptación de la HU "Mostrar cantidad de casos por clasificación".

<table>
<tr><th colspan="3">Caso de prueba de aceptación</th></tr>
<tr><td colspan="2">Código de caso de prueba: 01</td><td>Nombre historia de usuario: Mostrar cantidad de casos por clasificación</td></tr>
<tr><td colspan="3">Nombre de la persona que realiza el caso de prueba: Karel Fernández Rojas</td></tr>
<tr><td colspan="3">Descripción de la prueba: revisar a través del componente integrado al SIGAP el correcto funcionamiento del RF 1: Mostrar cantidad de casos por clasificación.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: se deben haber registrado escritos en el SIGAP, para que así el reporte que responde a este RF muestre resultados. Para obtener el reporte se debe seleccionar una fecha de inicio y otra fecha de fin, o dejar ambos campos de fechas vacíos.</td></tr>
<tr><td colspan="2">Entrada/Pasos de ejecución</td><td rowspan="2">Resultados esperados:</td></tr>
<tr><td>Acción:</td><td>Entrada:</td></tr>
<tr><td>Se decide obtener el reporte cantidad de casos por clasificación.</td><td>Fecha inicio y fecha fin, o ninguna de las dos fechas.</td><td>Se genera el reporte cantidad de casos por clasificación.</td></tr>
<tr><td colspan="3">Evaluación de prueba: Satisfactoria</td></tr>
</table>

Nota: fuente de elaboración propia.

Se realizó un encuentro con la compañera Madalina Marrero Delgado, jefa del área de atención a la población de la ANPP, con el objetivo de realizar las pruebas de aceptación al componente propuesto, considerando los casos de pruebas definidos para esta etapa. Obteniendo resultados satisfactorios, avalados por la funcionaria antes mencionada. Al finalizar el encuentro se generó el Acta de Aceptación del Producto (ver Anexo 3).

3.4. Verificación de los resultados de la investigación

Teniendo en cuenta que en la investigación realizada se define como idea a defender que: “el desarrollo de un componente para la obtención de información desde el SIGAP permite a los técnicos y especialistas del AAP de la ANPP agilizar los análisis estadísticos y la realización de informes”.

Para analizar la relación causa efecto entre la variable independiente “el desarrollo de un componente” y la variable dependiente “agilizar los análisis estadísticos y la realización de informes”, el autor de la presente investigación, define un conjunto de criterios de medida que permiten validar cómo a través del componente 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.

Criterios de medida definidos:

- Información oportuna: para el dominio de la investigación el autor define como información oportuna, la inmediatez con la cual los especialistas del área de atención a la población de la ANPP acceden a las estadísticas que generan la veintena de casos que diariamente procesan.
- Proceso de generación de informes: describe el procedimiento que los especialistas del área de atención a la población de la ANPP deben seguir para generar informes con las estadísticas de los procesos del área.
- Generación de reportes estadísticos: este criterio debe medir la forma en que los especialistas del área de atención a la población de la ANPP generan reportes con datos estadísticos de los procesos que realiza el área.

A continuación, en la Tabla 11, se evalúan cada uno de los criterios de medida definidos anteriormente. La evaluación se realiza estableciendo un antes del desarrollo del componente y un después de obtenido el componente, con el propósito de verificar cómo

mediante la solución propuesta se agilizan los análisis estadísticos y la realización de informes en el área de atención a la población de la ANPP. Los datos tenidos en cuenta en la comparación fueron tomados a partir de un piloto realizado en la ANPP.

Tabla 11. Evaluación de los criterios de medidas definidos.

Criterios de medida	Antes del componente	Después del componente
Información oportuna	Los especialistas y técnicos que trabajan en el área de atención a la población tardaban entre 1 y 2 horas en el conteo de casos por provincia, municipio o cualquier otro criterio de búsqueda, teniendo en cuenta que cada expediente se encuentra almacenado en formato duro en archivos foliados a los cuales había que acceder, buscar los casos en cada expediente e irlos contando manualmente. Esto trae como consecuencia que los análisis estadísticos de la cantidad de casos por clasificación, promovente, provincia u otros parámetros generados del proceso, no se puedan obtener	El proceso de obtención de cualquier información estadística relacionada con los casos que tramita diariamente el área de atención a la población de la ANPP, con el uso del componente desarrollado se reduce al tiempo de respuesta del sistema en mostrar los resultados de la búsqueda, el cual oscila entre unos 30 o 50 segundos, en relación a las 1 o 2 horas que había que esperar anteriormente cuando el proceso se realizaba de forma manual. .

	con inmediatez.	
Proceso de generación de informes	Los especialistas del área de atención a la población realizaban los informes de forma manual, teniendo primeramente que cuantificar los datos estadísticos a incluir en cada informe, siguiendo el procedimiento descrito en el indicador anterior, para luego elaborar cada informe.	Con la aplicación del componente desarrollado el proceso de elaboración de informes se reduce a solo tener que acceder a la funcionalidad generar informe y obtener el mismo de forma automática con los datos estadísticos ya incluidos.
Generación de reportes estadísticos	Los especialistas del área de atención a la población realizaban los reportes de forma manual, teniendo que cuantificar los datos estadísticos manualmente, siguiendo el procedimiento descrito en el primer indicador, proceso que se hacía lento y con riesgo de cometerse errores en el cálculo de algún indicador.	Con la aplicación del componente desarrollado el proceso de obtención de reportes estadísticos se reduce a solo tener que acceder a la funcionalidad generar reporte, definir el rango de fecha para el cual se desea obtener el reporte y generar el mismo de forma automática, sin riesgo de equivocación en los cálculos.

Nota: fuente de elaboración propia.

La comparación realizada en la tabla anterior, a través de los criterios de medida antes definidos, demuestra cómo utilizando el componente propuesto en la presente investigación, se agilizan los análisis estadísticos y la realización de informes.

3.5. Conclusiones parciales

La aplicación de métricas de validación del diseño y los niveles de pruebas de unidad, integración, sistema y aceptación del componente propuesto, certifican la obtención de una solución informática funcional, avalada en cada caso por las actas de liberación y aceptación emitidas por el grupo de calidad de CEGEL y los especialistas de la ANPP. Por otra parte, la validación del resultado de la investigación, demuestra el cumplimiento de la relación causa efecto de la variable independiente “el desarrollo de un componente” sobre la variable dependiente “agilizar los análisis estadísticos y la realización de informes”.

CONCLUSIONES GENERALES

El estudio de los referentes teóricos vinculados con soluciones informáticas para la gestión del proceso de atención a la población, permitieron obtener características como la forma de generar reportes estadísticos, con el propósito de ser incorporadas al desarrollo del componente propuesto.

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, permiten obtener un componente que agiliza los análisis estadísticos y la realización de informes a través del SIGAP.

El desarrollo de pruebas de software en los niveles de unidad, integración, sistema y aceptación, permitieron obtener resultados que corroboran el correcto funcionamiento del componente propuesto en la presente investigación.

Con la validación de la relación causa efecto de la variable independiente sobre la variable dependiente de la investigación, se demuestra que con el desarrollo del componente propuesto se agilizan los análisis estadísticos y la realización de informes.

RECOMENDACIONES

Incorporar al componente nuevos reportes que enriquezcan las funcionalidades del SIGAP.

Desarrollar mejoras en la arquitectura del componente que permitan el desarrollo de reportes dinámicos.

BIBLIOGRAFÍA REFERENCIADA

Administración de Requeriminetos. (2014). Obtenido de https://administracionderequerimientos.wordpress.com/2014/08/26/tecnicas-para-la-captura-de-requisitos/

Alegsa.com.ar. (2017). Obtenido de http://www.alegsa.com.ar/Dic/framework.php

Caceres, M. (2017). *Devcode*. Obtenido de https://devcode.la/blog/que-es-typescript/

Calleja, M. A. (2010). *Estándares de codificación.*

Contreras Gálvez, M., Fuenzalida López, F., López Rojas, C., & Mosso Gutiérrez, M. (2010). *Sistema Integral de Información y Atención Ciudadana, SIAC.* Chile.

Definición de Herramienta de modelado. (2018). Obtenido de http://www.alegsa.com.ar/Dic/herramienta_de_modelado.php

Elkan, M. (2017). *Tutor de Programación*. Obtenido de http://acodigo.blogspot.com/2014/11/spring-boot-introduccion.html

Elósegui, T. (2014). *Analitica Web*. Obtenido de https://tristanelosegui.com/2014/10/27/que-es-y-para-que-sirve-un-dashboard/

Escalante, L. C. (2014). *El patrón de arquitectura n-capas con orientación al dominio.*

fergarciac. (2013). Obtenido de https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/

Francisco José García Peñalvo, C. P. (2012). *Diagramas de Clase en UML .* Burgos.

Genbeta:dev. (2017). Obtenido de https://www.genbetadev.com/herramientas/netbeans-1

Industrias, S. N. (2018). *SNI: Sociedad Nacional de Industrias*. Obtenido de http://www.sni.org.pe/?page_id=872

Instinto Binario. (2017). Obtenido de https://instintobinario.com/patrones-de-diseno-patron-observador/

ISDI. (2014). Obtenido de ISDI: https://www.isdi.education/es/isdigital-now/herramienta-te-permite-realizar-prototipos-de-tus-proyectos-balsamiq

ISOTools. (2017). *Sistema De Atención De Peticiones, Quejas, Reclamos Y Sugerencias (PQRS)*. Obtenido de https://www.isotools.org/2017/10/24/sistema-atencion-peticiones-quejas-reclamos-sugerencias-pqrs/

Katerine Villamizar Suaza, J. J. (2015). *Mejora de historias de usuario y casos de prueba.*

Lengua, A. M. (2018). *Academia Méxicana de la Lengua*. Obtenido de http://www.academia.org.mx/

Londoño, J. H. (2005). *Ingeniería de Software*. Obtenido de http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html

Lorenz, M., & Kidd, J. (1994). *Object-oriented software metrics: a practical guide.* Nueva Jersey: Prentice-Hall.

Mejias Cruz, J. (2018). *Manual de usuario del Sistema de gestión para la atención a la población.* La Habana.

Microbuffer. (2011). Obtenido de https://microbuffer.wordpress.com/2011/05/04/que-es-postgresql/

Mora, R. C. (2016). *AdictosAlTrabajo.com*. Obtenido de https://www.adictosaltrabajo.com/tutoriales/grasp/

Ochoa, J. (2018).

Pacheco, J. C. (2017). *Inyección de Dependencias*. Obtenido de https://msdn.microsoft.com/es-es/communitydocs/net-dev/csharp/inyeccion-de-dependencias

Power Data. (2016). Obtenido de https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-es-un-gestor-de-datos-y-para-que-sirver

Pressman, R. S. (2002). *Ingeniería del Software. Un enfoque práctico. Quinta Edición.* Mc Graw Hill.

Programación de Interfaces de Usuario. (2018). Obtenido de http://algo3.uqbar-project.org/material/herramientas/angular/angular---conceptos-principales

Roche, E. (2013). */DEV/Blog*. Obtenido de https://barradevblog.wordpress.com/2013/04/23/el-patron-repositorio-repository-pattern-implementacion-practica-con-entity-framework/

Rodríguez, T. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana.

Sánchez, T. R. (2015). *METODOLOGIA UCI 2015.pdf.*

Servicio de Informatica de ASP.NET MVC 3 Framework. (2017). Obtenido de https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

Soft, S. (2018). Obtenido de http://www.serpol.net.mx/quejas

SOFTWARE, A. (2016). Obtenido de https://arandasoft.com/por-que-tener-un-sistema-para-atencion-de-peticiones-quejas-y-reclamos/

Tedeschi, N. (2017). *Microsof*. Obtenido de https://msdn.microsoft.com/es-es/library/bb972240.aspx

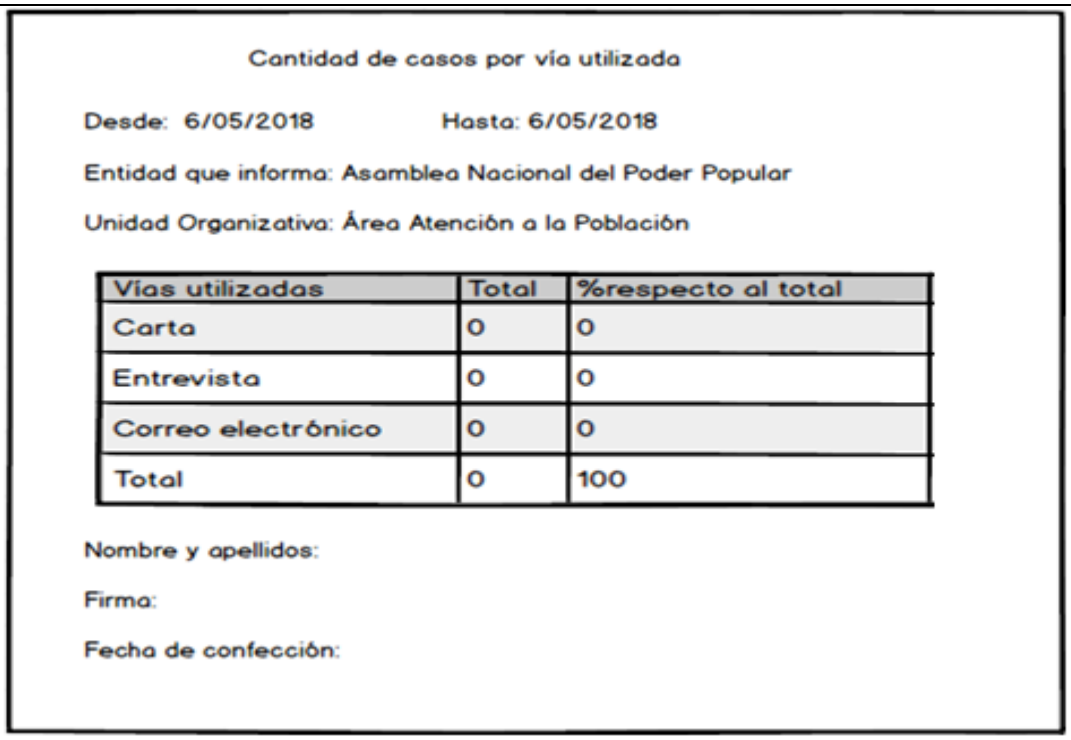
Venemedia. (2014). *conceptodefinicion.de*. Obtenido de http://conceptodefinicion.de/javascript/

ANEXOS

Anexo 1: Historias de Usuario

Tabla 12. Historia de Usuario Cantidad de casos por vía utilizada.

Historias de Usuario	
Número: 2	**Nombre de la HU:** cantidad de casos por vía utilizada.
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: alto	**Tiempo real:** 10 días
Descripción: permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por la vía que estos llegan (carta, llamada telefónica, correo electrónico, entrevista u otro medio desde internet. Debe permitir mostrar los datos por un rango de fecha.	
Observaciones:	
Prototipo de interfaz:	



Cantidad de casos por vía utilizada

Desde: 6/05/2018 Hasta: 6/05/2018

Entidad que informa: Asamblea Nacional del Poder Popular

Unidad Organizativa: Área Atención a la Población

Vías utilizadas	Total	%respecto al total
Carta	0	0
Entrevista	0	0
Correo electrónico	0	0
Total	0	100

Nombre y apellidos:

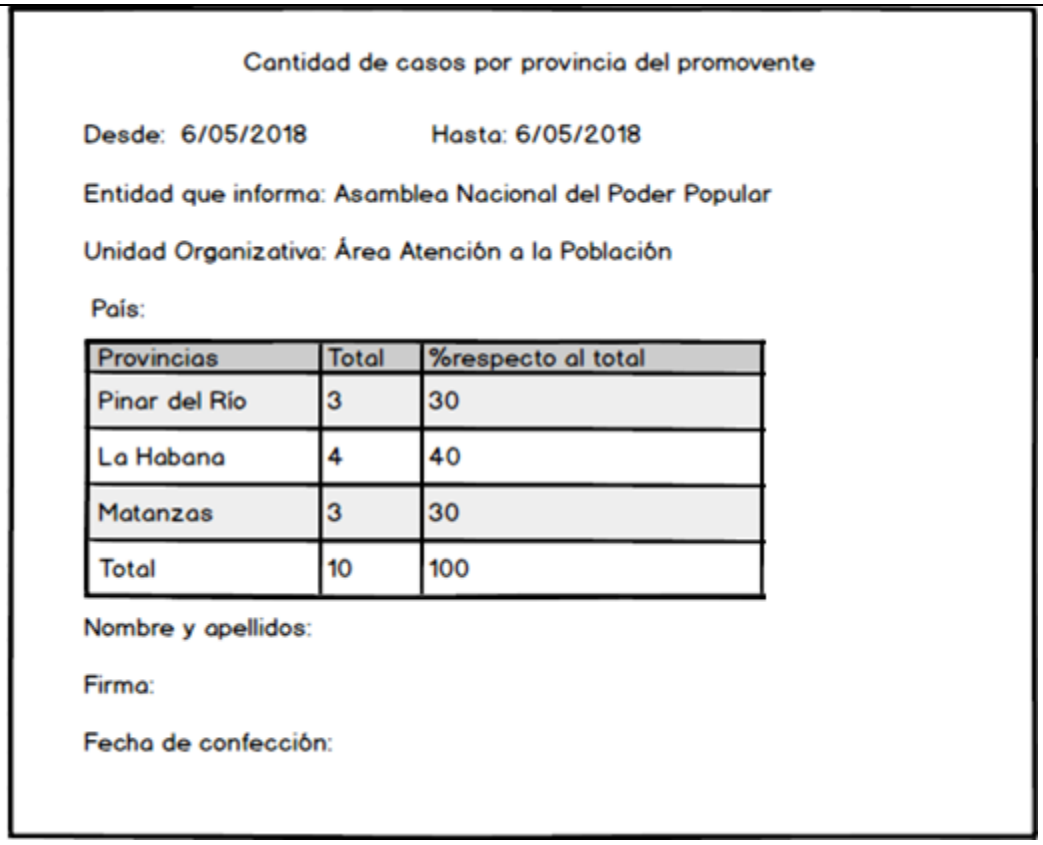
Firma:

Fecha de confección:

Fuente: elaboración propia.

Tabla 13. Historia de Usuario Cantidad de casos por provincia del promovente.

Historias de Usuario	
Número: 3	**Nombre de la HU:** cantidad de casos por provincia del promovente.
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: alto	**Tiempo real:** 10 días
Descripción: permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por la provincia del promovente. Debe permitir mostrar los datos por un rango de fecha.	
Observaciones:	
Prototipo de interfaz:	



Cantidad de casos por provincia del promovente

Desde: 6/05/2018 Hasta: 6/05/2018

Entidad que informa: Asamblea Nacional del Poder Popular

Unidad Organizativa: Área Atención a la Población

País:

Provincias	Total	%respecto al total
Pinar del Río	3	30
La Habana	4	40
Matanzas	3	30
Total	10	100

Nombre y apellidos:

Firma:

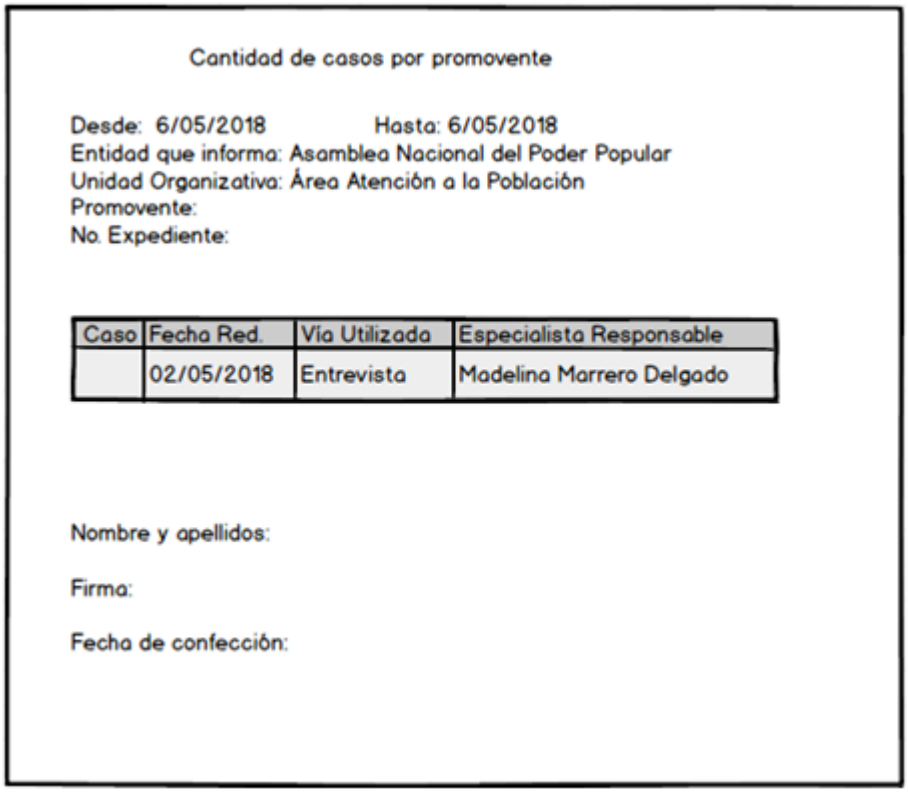
Fecha de confección:

Fuente: elaboración propia.

Tabla 14. Historia de Usuario Cantidad de casos por promovente.

Historias de Usuario	
Número: 5	**Nombre de la HU:** cantidad de casos por promovente.
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: alto	**Tiempo real:** 10 días
Descripción: permite visualizar la cantidad de casos atendidos en el área de atención a la población, organizados por promovente. Debe permitir mostrar los datos por un rango de fecha.	
Observaciones:	

Prototipo de interfaz:

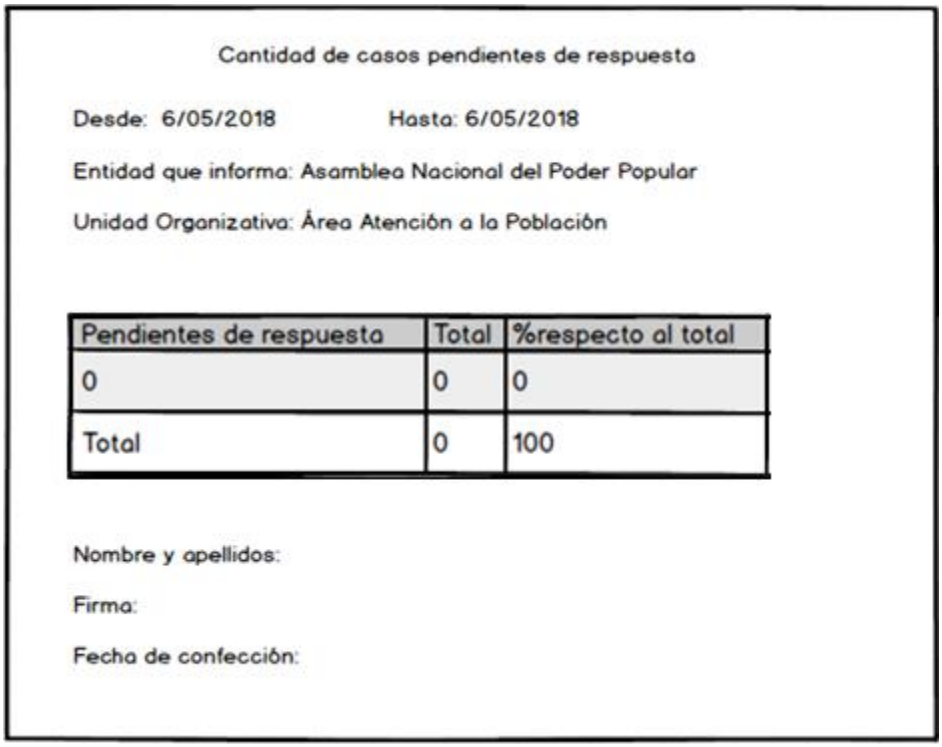


Fuente: elaboración propia.

Tabla 15. Historia de Usuario Cantidad de casos pendientes a respuesta.

Historias de Usuario	
Número: 6	**Nombre de la HU:** cantidad de casos pendientes a respuesta
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: alto	**Tiempo real:** 10 días
Descripción: permite visualizar la cantidad de casos atendidos en el área de atención a la población que se encuentren pendientes a una respuesta. Debe permitir mostrar los datos por un rango de fecha.	

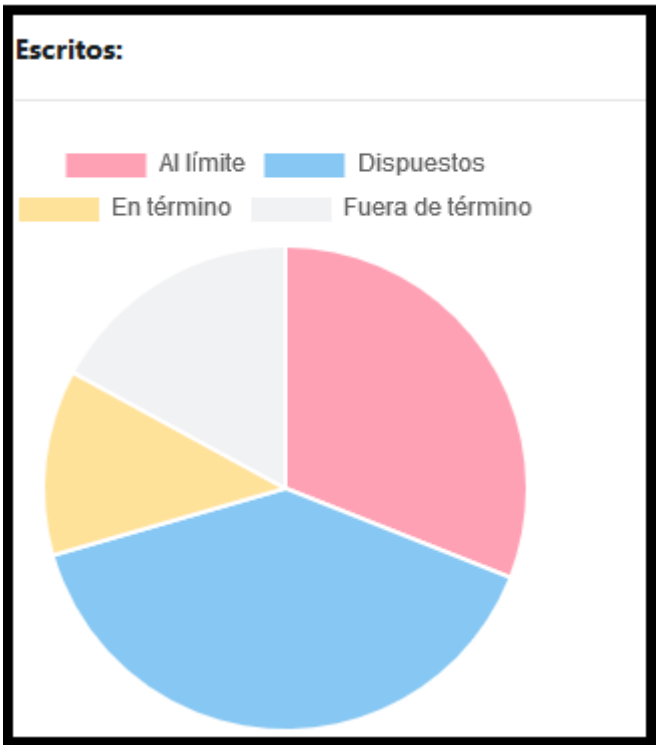
Observaciones:
Prototipo de interfaz:



Fuente: elaboración propia.

Tabla 16. Historia de Usuario Visualizar estado permanente de los escritos.

Historias de Usuario	
Número: 7	**Nombre de la HU:** Visualizar estado permanente de los escritos.
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: alto	**Tiempo real:** 10 días
Descripción: permite mostrar en la pantalla principal el estado de los escritos registrados durante un mes en el SIGAP.	

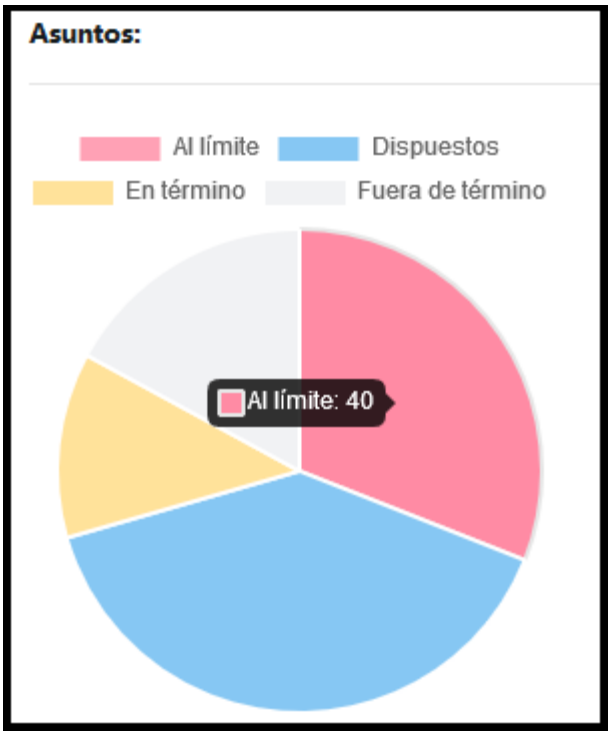
Observaciones:
Prototipo de interfaz: 

Fuente: elaboración propia.

Tabla 17. Historia de Usuario Visualizar estado permanente de los asuntos.

Historias de Usuario	
Número: 8	**Nombre de la HU:** Visualizar estado permanente de los asuntos.
Programador: Karel Fernández Rojas	**Iteración asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: alto	**Tiempo real:** 10 días
Descripción: permite mostrar en la pantalla principal el estado de los asuntos registrados durante un mes en el SIGAP.	

Observaciones:
Prototipo de interfaz:



Fuente: elaboración propia.

Anexo 2: Acta de Liberación de Productos de Software

Universidad de las Ciencias Informáticas

Acta de Liberación del Laboratorio de Pruebas CEGEL

1. Datos Generales

Centro:	Centro de Gobierno Electrónico	**Fecha:**	04-06-2018
Proyecto:	Sistema para la Gestión de Quejas y Opiniones	**Elaborada por:**	Ing. Isis Bertami Barrios
Producto:	SIGAP	**Cargo:**	Asesora de Calidad
Versión:	1.0		

1.1 Descripción del Producto

SIGAP es una herramienta que permite realizar los procesos asociados a la atención a la población. Este sistema sería la cara que recibiría el ciudadano cubano cuando decida presentar un escrito ante los órganos del estado competentes. Una vez recibido el escrito, el sistema facilitará la realización de los diferentes procesos asociados al tratamiento del mismo, según las competencias de cada entidad gubernamental.

2. Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
Producto SIGAP	1.0	0	4	Funcionalidad	04-06-2018

3. Participantes en las pruebas de liberación.

Nombre y Apellidos	Rol que desempeña
Isis Bertami Barrios	Asesora de Calidad
Camilo José Tamayo	Probador
Lisandra Santamaría Pérez	Probador
Felinda Rosabel León Mendoza	Probador

CEGEL
Isis Bertami Barrios
Asesora de Calidad CEGEL
Universidad de las Ciencias Informáticas
CENTRO DE GOBIERNO ELECTRÓNICO

Ing. Francisco Gabriel Rubio
Líder del Proyecto

Figura 23. Acta de liberación de calidad.

Fuente: elaboración propia

Anexo 3: Acta de Aceptación del Producto

UCI Universidad de las Ciencias Informáticas

Acta de aceptación

En cumplimiento del desarrollo de la Tesis: **Componente para la obtención de información desde el Sistema de Gestión para la Atención a la Población.** Se hace entrega del producto:

- Sistema de gestión para la atención a la población (SIGAP), al cual se integra el Componente para la obtención de información desde el SIGAP.

La Parte Cliente, luego de haber revisado el software, considera que la solución cumple con cada uno de los requisitos que originaron su desarrollo. Conforme a lo anterior se expresa la aceptación de la misma.

Entrega	Recibe
Nombre y apellidos: Karel Fernández Rojas	**Nombre y apellidos:** Madalina Marrero Delgado
Cargo: Estudiante	**Cargo:** Jefa del Área de Atención a la Población de la Asamblea Nacional del Poder Popular

Fecha: 14/06/2018

Figura 24. Acta de aceptación del componente.

Fuente: elaboración propia