Universidad de las Ciencias Informáticas

Facultad 3

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

Componente para la gestión de notificaciones para el marco de trabajo XEGFORT.

Autora: Esther García Naranjo.

Tutor: Ing. Reinier Silverio Figueroa.

Ing. Juan David Gómez Amador.

La Habana, julio de 2016

Declaración de autoría

Declaro ser el autor de la presente tesis y reconozco a la Universidad de Ciencias Informáticas de los derechos patrimoniales de la misma. Con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2016.

_____ _____

Ing. Reinier Silverio Figueroa Ing. Juan David Gómez Amador

Tutor Tutor

_____

Esther García Naranjo

Autora

Datos de contacto
Tutor: Ing. Reinier Silverio Figueroa.
Título: Ingeniero en Ciencias Informáticas.
Correo electrónico: rsilveriof@uci.cu

Tutor: Ing. Juan David Gómez Amador
Título: Ingeniero en Ciencias Informáticas.
Correo electrónico: juandg@uci.cu

"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein

Dedicatoria

Dedico este trabajo a mi abuelita por darme todo el cariño, confianza y fuerza para lograr los objetivos que me he trazado en estos años de universidad.

Agradecimientos

A Dios porque sin el nada hubiera funcionado en mi vida, porque es el mejor.

Agradecerle a mi familia y en especial a cuatro personas muy importantes en mi vida, a mi abuelita por saberme educar , apoyarme y darme todo el amor y cariño en todos estos años de vida; a mi tío por la educación que me brindó junto a mi abuela y por haber desempeñado el papel de padre; a mi mamá por estar presente en los momentos más difíciles de mi vida y por apoyarme siempre, a mi tía, que además de tía ha desempeñado el papel de madre y amiga en este último año de universidad y gracias por toda su comprensión, amor y cariño.

Un agradecimiento especial a mi tutor Reinier Silverio que fue un ángel, llego en el momento justo, cuando más lo necesitaba, gracias por la comprensión y por soportarme, gracias por confiar en mi aun cuando yo ni lo hacía, infinitamente le estaré agradecida.

A Juan David Gómez Amador por toda la ayuda que me brindó y la paciencia que tuvo conmigo.

A mi enano favorito Yordanis García por la ayuda que me brindó en todo momento, gracias por todo, gracias por estar ahí cuando más lo necesitaba.

A todos mis ciberamigos que a pesar de la distancia siempre me ayudaron y me apoyaron en todo momento. Gracias a todos por su gran compresión y por soportarme todos estos años y por si fuera poco este año me tuvieron que soportar más, pues tuve la dicha de conocerlos a todos, gracias por estar siempre a mi lado a pesar de las circunstancias, en especial a Raidel, Dennys, Marlon, Duniel, Ramoncito (Guille), Yancel, Oscar, Yoandi (el chino), Osvel, Yoan.

A todos mis amigos de Granma y a los que hice nuevos aquí en la UCI.

A mis amigas de siempre Daylenis Burgos y María Esther Orozco.

A mis compañeras de cuarto, gracias por su comprensión y cariño a pesar de todas las cosas.

A Alejandro Rodríguez Valerino por todas las cosas buenas que me dio en todos los años que estuvimos juntos, por enseñarme a vivir, por hacerme fuerte ante las adversidades y enseñarme a luchar cuando todo parezca perdido y por ayudarme siempre que pudo.

A todos los que de una forma u otra ayudaron a la realización de mi tesis y siempre confiaron en mí a pesar de todo.

Resumen

Las notificaciones son técnicas utilizadas para comunicarle a personas sobre eventos referentes a su trabajo o sobre noticias de su interés. Este trabajo presenta la propuesta de un componente de gestión de notificaciones para el marco de trabajo XEGFORT, posibilitando el envío de notificaciones a través del correo electrónico. También permite gestionar las cuentas de correo electrónico y los servidores de envío, además pueden enviar correos dándole a los usuarios la oportunidad de estar mejor informados sobre eventos de su interés.

Para dar cumplimiento al objetivo de la investigación se realizó una fundamentación teórica donde se analizaron los conceptos fundamentales relacionados con las notificaciones. Se hizo un estudio de las soluciones similares existentes a nivel internacional y nacional. También se efectuó un estudio de las tecnologías, herramientas y lenguajes utilizados en el desarrollo del componente, así como las pruebas realizadas para la validación de la solución propuesta.

Palabras claves: correo electrónico, notificación, XEGFORT

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

En la actualidad, el uso de las Tecnologías de la Información y las Comunicaciones (TIC) se ha desarrollado brindando innumerables beneficios a la sociedad. Las computadoras y sus programas están presente en la mayoría de las esferas donde se maneja información con el fin de tomar decisiones a tiempo e incrementar el control y la eficiencia. La introducción de las TIC en el sector público ha permitido la organización de actividades gubernamentales, llegando al momento en que la gestión y la participación ciudadana actúen con efectividad.

El desarrollo de estudios recientes sobre del impacto de estas tecnologías en las Ciencias Políticas y la Administración Pública demuestran cómo estas se han convertido en verdaderas estrategias para la modernización y optimización de procesos en entidades privadas y gubernamentales. Hoy en día la administración pública cubana es un firme reflejo de la evolución en la rama tecnológica. Cuba cuenta con muchas instituciones que sirven de base al aumento del conocimiento con respecto a las TIC, entre ellos la Universidad de Ciencias Informáticas (UCI).

La UCI surge como parte de los programas de la batalla de ideas, con la misión principal de formar profesionales comprometidos con su patria y altamente calificados en la rama de la informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación y servir de soporte a la industria cubana de la informática. Esta cuenta con varios centros distribuidos por las diferentes facultades, entre ellos el Centro de Gobierno Electrónico (CEGEL), especializado en realizar soluciones de gestión para la informática jurídica y el gobierno electrónico. Este centro presenta una elevada integración en los procesos de producción, investigación y formación de pregrado y postgrado. Su principal misión es la de satisfacer necesidades de clientes gubernamentales mediante el desarrollo de productos, servicios y soluciones integrales de alta confiabilidad, calidad y eficiencia.

CEGEL desarrolló XEGFORT, un marco de trabajo base implementado sobre la plataforma Java Enterprise Edition (J2EE), que permite desarrollar sistemas de gestión para la administración pública. Suministra a las aplicaciones que lo utilicen funcionalidades y componentes que aceleran y simplifican su desarrollo, como son la gestión de las sesiones de los usuarios y la gestión de excepciones (Matos Rodríguez, y otros, 2011).

En una encuesta realizada a especialistas de los proyectos del centro se determinó que cada día son mayores las solicitudes por parte de los clientes que requieren mecanismos de notificaciones a través de correos electrónicos (ver Anexo 1). Las notificaciones son un elemento normado en el campo del derecho administrativo, se han insertado allí como una alternativa inmediata para lograr que los procesos judiciales, que utilicen este medio, se desarrollen con una mayor celeridad, economía y seguridad procesal.

Desde XEGFORT se realizan algunas notificaciones para un solo destinatario, a partir de un evento determinado en un flujo de negocio. Sin embargo, para una situación similar donde existen múltiples destinatarios, se requiere salir del flujo y generar manualmente un reporte con el listado de los usuarios involucrados, teniendo en cuenta su dirección de correo electrónico, para luego enviar las notificaciones persona a persona. Lo anterior implica invertir tiempo adicional para redactar y enviar correos a cada destinatario. Además este marco de trabajo tampoco brinda la posibilidad del envío automático de correos en fechas planificadas. Lo antes descrito trae como resultado que el proceso sea repetitivo, implicando la utilización de mayor cantidad de tiempo y el riesgo de que algunos destinatarios no sean notificados.

Una vez analizada la situación anterior se formula el siguiente **problema a resolver:** ¿Cómo gestionar las notificaciones en XEGFORT de forma tal que contribuya a mejorar la simplicidad y efectividad del proceso?

Se define como **objeto de estudio** sistemas de gestión para la administración pública.

Teniendo como **campo de acción** sistemas de notificaciones por correo electrónico en CEGEL.

Para dar solución al problema antes planteado se traza el siguiente **objetivo general:** desarrollar un componente de notificaciones a través de correos electrónicos para XEGFORT que contribuya a mejorar la simplicidad y efectividad del proceso.

A partir del objetivo general se derivan los siguientes **objetivos específicos:**

- Realizar un estudio de estado del arte de sistemas existentes para notificaciones en la gestión de la administración pública.
- Desarrollar un componente para la gestión de notificaciones del marco de trabajo XEGFORT que contribuya a mejorar la simplicidad y efectividad del proceso.
- Evaluar la solución propuesta con la finalidad de comprobar el cumplimiento del objetivo de la investigación.

Determinando como **idea a defender:** el desarrollo de un componente para la gestión de notificaciones del marco de trabajo XEGFORT que contribuya a mejorar la simplicidad y efectividad del proceso.

Métodos científicos de la investigación

Métodos teóricos

Análisis-Síntesis: permitió analizar y sintetizar la información obtenida durante el estudio de los principales conceptos asociados a la creación del componente, proporcionando la extracción de los elementos importantes relacionados con este proceso.

Histórico-Lógico: posibilitó el estudio de trabajos realizados en el ámbito de notificar en un sistema informático y de soluciones previas existentes a problemas similares al actual. Se utilizó para

determinar a través de la evaluación conceptos de esta temática, que permiten conocer el estado actual de este fenómeno e identificar posibles mejoras y alternativas de solución.

Métodos empíricos

Observación: se emplea para constatar cómo marcha el desarrollo de la aplicación XEGFORT, para el envío de notificaciones por correo y verificar el estado de las mismas. Permitió el análisis de los resultados obtenidos en soluciones a problemas similares relacionados con el envío de notificaciones.

Entrevista: se utilizó como técnica de recopilación de información con el propósito de conseguir los datos referentes a las herramientas utilizadas para el envío de notificaciones por correo electrónico, permitió conocer la experiencia de otros desarrolladores que han trabajado con el marco de trabajo XEGFORT y obtener los elementos necesarios para definir qué es lo que se necesita y los problemas existentes que se resolverían.

Estructura del documento

El presente trabajo de diploma se ha estructurado en tres capítulos para mostrar el desarrollo de la investigación y los resultados alcanzados, además de contar con Conclusiones, Bibliografía y Anexos.

Capítulo 1: Fundamentación teórica.

Este capítulo contiene los fundamentos teóricos que sostienen la investigación. Se definen las herramientas empleadas, la metodología a utilizar para el desarrollo de la solución y las estrategias de pruebas a seguir para verificar que la aplicación cumple con los requisitos definidos por el cliente.

Capítulo 2: Componente para la gestión de notificaciones para el marco de trabajo XEGFORT.

En este capítulo se exponen las características generales de la solución propuesta y se detallan los requisitos funcionales y no funcionales que se tienen en cuenta para su implementación. Se especifica la arquitectura, los patrones de diseño empleados y los artefactos derivados de la metodología de desarrollo seleccionada.

Capítulo 3: Validación de la solución propuesta.

En este capítulo se recoge la validación de la solución propuesta, así como los resultados realizados para asegurar su correcto funcionamiento.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se abordan los elementos teóricos necesarios para el desarrollo de la investigación, además de algunas aplicaciones existentes tanto en el ámbito nacional como internacional que efectúa el tema de las notificaciones. Se tratan aspectos relacionados con el uso de la metodología de desarrollo, herramientas y tecnologías que se utilizaran en el desarrollo del componente, así como las pruebas a realizar una vez terminado el componente.

1.2 Principales conceptos asociados a la investigación

En el siguiente epígrafe se describen los principales conceptos asociados a la investigación referente al envío de notificaciones y el tipo de notificación seleccionado.

1.2.1 Notificaciones

Según la Real Academia Española, plantea que una notificación es “Acción y efecto de notificar. Documento en que consta la notificación de una resolución“ (Española, 2015).

Otro concepto sobre notificación es la forma de hacer saber sobre una resolución de la autoridad con las formalidades preceptivas para el caso. Dar extrajudicialmente, con propósito cierto, noticia de una cosa (Océano, 2015).

La notificación es la forma que se emplea a la hora de comunicarle a una persona o un público determinado una noticia o una información. El efecto que produzca una notificación en la persona dependerá de la naturaleza de la misma. Este concepto se puede expresar a través de sus sinónimos como: comunicado, aviso, noticia, anuncio, proclama, declaración, certificación o circular (Pérez, y otros, 2005).

Entre los diferentes tipos de notificaciones analizados en la literatura se encuentran (Correa Selamé, 2006)**:**

- **Notificación personal**: consiste en entregar a la persona a quien se debe notificar, en forma personal, copia íntegra de la resolución y de la solicitud en que haya recaído, cuando esta fuese escrita.
- **Notificación personal subsidiaria:** es aquella que se aplica cuando al intentar la notificación personal, el notificado no se ha presentado. A diferencia de la notificación en persona, esta solo puede practicarla el receptor o eventualmente un notario si no hay receptor.
- **Notificación por avisos**: notificación sustituta de la personal, se utiliza respecto a personas cuya individualidad o residencia es difícil de determinar o que por su número dificulten considerablemente la práctica de la diligencia.

- **Notificación por carta certificada**: es aquella que se entrega en el domicilio o lugar de su actividad, esta contiene una copia de la resolución o actuación de que se trata y que el funcionario de correos debe entregar a cualquier persona adulta que se encuentre en la dirección de destino, debiendo esta firmar el recibo correspondiente.
- **Notificación por el estado diario:** es una clase de notificación que consiste en que la parte toma conocimiento directo del contenido de la resolución a través del estado. Un listado donde se exhiben todas las resoluciones dictadas por el tribunal durante el día. Se confecciona diariamente, se encuadernan por orden de fecha y se archivan mensualmente.
- **Notificación electrónica:** las notificaciones electrónicas son aquellas comunicaciones que son realizadas utilizando medios electrónicos y telemáticos, tales como el Internet y el correo electrónico.

  Para la realización del presente trabajo se toma que una notificación es la forma que se emplea a la hora de comunicarle a una persona o un público determinado una noticia o una información. Para ello se selecciona la notificación electrónica por ser la más factible para el desarrollo de la investigación.

1.2.2 Correo electrónico

Según la Real Academia Española, el correo electrónico es un “sistema de transmisión de mensajes por computadoras a través de redes informáticas” (Española, 2015).

El correo electrónico (conocido como e-mail) es un servicio que permite el intercambio de mensajes a través de sistemas de comunicación electrónicos de forma rápida y segura. Pero no solo se envían textos, también cualquier tipo de documentos digitales. Este tipo de tecnología se basa en el protocolo para transferencia simple de correo (SMTP). Además de ser más fácil, cuenta con múltiples posibilidades como de filtrar determinado tipo de mensaje o destinatario y como enviarle un mismo mensaje a varios destinatarios a la vez (Lamarca Lapuente, 2013).

Teniendo en cuenta los conceptos antes mencionados para la presente investigación se toma que un correo electrónico es un sistema de transmisión de mensajes por computadoras a través de redes informáticas en el cual se puede enviar cualquier tipo de documento.

1.2.3 Notificaciones por correo electrónicos

La notificación por correo electrónico es aquella comunicación dirigida a los domicilios o direcciones electrónicas de los usuarios. Estas direcciones o casillas electrónicas son las direcciones electrónicas procesales de las partes y constituye la residencia habitual, en la red de Internet, de la persona (Núñez Ponce, 2016).

Otro concepto define que es una comunicación electrónica que con plena validez, realizan las administraciones públicas por medios telemáticos de las resoluciones y actos administrativos que afectan al interesado. Es un servicio que permite al interesado ser notificado, por parte de las

administraciones públicas a través de una dirección electrónica accesible desde internet, de las resoluciones y actos administrativos que les afecten con plena validez jurídica (Díez de los Ríos, 2012).

Existen en el mundo numerosas implementaciones de estos sistemas entre los que se pueden mencionar: servicios de mensajería instantánea, aplicaciones que muestran el estado de un sistema, servicios de noticias e indicadores de cantidades almacenadas; además de sistemas más sofisticados como: los incorporados en los vehículos para proporcionar información de los mismos, los multi-monitor que indican excesos de rango y las cabinas de los aviones modernos.

Para el trabajo se toma que una notificación por correo electrónico es el envío de mensajes y avisos por programas o servicios para advertir de un evento a los usuarios, esta tiene como propiedad no causarle ninguna interrupción cuando se esté realizando alguna tarea en el sistema en el momento en que la misma llega.

1.3 Soluciones similares existentes

Se realizó una investigación de los sistemas existentes a nivel nacional e internacional que ejecutan el envío de notificaciones, facilitando un mayor entendimiento del problema planteado. A continuación se muestran algunos sistemas que ejecutan el envío de notificaciones.

1.3.1 Soluciones existentes a nivel internacional

Gmail Notifier

Es la herramienta de notificación de las cuentas de Gmail desarrollado oficialmente por Google. Entre las características de esta herramienta se tiene que la misma muestra una ventana de aviso en la esquina inferior derecha de la pantalla con datos como el remitente, asunto y las primeras líneas del mensaje y muestra un resumen de los mensajes nuevos. Además presenta un contador de mensajes y se actualiza automáticamente. Solo es para aquellos usuarios que poseen una cuenta de correo electrónico Gmail y está diseñado para Windows 2000 en adelante (Google, 2015).

Facebook

Facebook es un sitio web gratuito y una red social creado por Mark Zuckerberg. Las relaciones se efectúan a través de un perfil de usuario con otros usuarios denominados amigos, grupos o páginas a través de una línea de tiempo de noticias en la que se comparten los estados y donde se ven los estados de los amigos. Facebook permite configurar las notificaciones brindando dos opciones: cómo recibes las notificaciones y notificaciones que recibes. En la opción cómo recibes notificaciones permite configurar que lleguen las notificaciones de Facebook al correo electrónico y al teléfono móvil; y en la opción Notificaciones que recibes da la opción de configurar qué tipo de notificaciones y de quién las deseas recibir (Zuckerberg, 2016).

1.3.2 Soluciones existentes a nivel nacional

En la UCI se han creado algunas aplicaciones que entre sus módulos cuentan con la funcionalidad incorporan el envío de notificaciones a través de correos electrónicos. Entre ellas se encuentran:

Sistema de notificación de eventos (SNE)

Forma parte del proyecto de Video Vigilancia Inteligente desarrollado por la UCI para el MINTUR. Componente creado con el objetivo de enviar a una determinada persona una notificación sobre algún hecho que haya ocurrido en el área o recurso asignado para su protección y así poder prevenir algún hecho significativo. El envío de mensajes se realiza haciendo uso de diferentes medios de comunicación, entre las vías de comunicación se encuentra sistema de mensajes simples (SMS), protocolo de transferencia de hipertexto (HTTP), Beeper, Socket y el correo electrónico. Esto facilita una mejor comunicación entre el personal de cualquier empresa o institución que utilice los componentes en un determinado software. Además fue preparado para futuras necesidades en cuanto a medios de transmisión (Barroso Pérez, 2011).

Módulo de notificaciones del Sistema Integral de Análisis de Información (MNSIAI)

Es una aplicación web para la gestión de los eventos y una aplicación de escritorio para la notificación de los mismos en las máquinas clientes. Con dicha aplicación se obtiene una mejor centralización, organización, distribución y notificación de los eventos, de manera que los usuarios puedan estar informados de forma automática sin necesidad de realizar búsquedas por los diferentes medios. Actúa como intermediario en las comunicaciones entre aplicaciones, contribuye a mejorar el funcionamiento de soluciones con estas características y mejora la interoperabilidad entre aplicaciones (Cruz Figueredo, 2013).

Resultados del estudio de los sistemas similares

Una vez analizados los sistemas a nivel nacional e internacional se identificaron varios criterios comparativos: el componente debe presentar mecanismos de integración compatibles con la implementación existente del marco de trabajo XEGFORT, debe ser agnóstico de dominio, tener la capacidad de enviar notificaciones a destinatarios no registrados en el sistema y no forzar al destinatario utilizar componentes extras. En la Tabla 1 se muestran los resultados de este estudio.

Tabla 1. Comparación de sistemas similares existentes (Fuente: elaboración propia).

Sistemas analizados	Criterios analizados			
	Mecanismos de integración	Orientados a dominios	Envío a usuarios no registrados	Necesidad de componentes extras
Gmail Notifier	NO	NO	NO	SI
Facebook	NO	SI	NO	NO
SNE	NO	SI	SI	NO

MNSIAI	NO	SI	NO	SI

Debido a que los sistemas estudiados no alcanzan a cumplir la totalidad de las necesidades descritas, se descartan como solución al problema. Por tanto, se concluye que se debe desarrollar un componente que realice el envío de notificaciones a través de correo electrónico.

1.4 Metodología de desarrollo

Una metodología es un conjunto integrado de técnicas y métodos que permiten abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge hasta que se cumple el objetivo por el cual fue creado (Samuel, 2013).

Para la implementación del componente es necesario utilizar una metodología de desarrollo que guíe el proceso, para la presente investigación fue seleccionada la metodología XP, la cual fue inicialmente utilizada en el desarrollo del marco de trabajo XEGFORT.

1.4.1 Programación extrema (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito, promoviendo el trabajo en equipo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo. Adecuada para proyectos pequeños con requisitos imprecisos y cambiantes. Todos los requisitos son expresados como escenarios (llamados Historias de Usuarios (HU)). Los programadores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código (Somerville, 2015).

Fases de XP (Navarro Cadavid, y otros, 2013)

Exploración: en esta fase se plantean a grandes rasgos las HU que son de interés para la primera entrega del producto. Se prueba la tecnología y se analizan las posibilidades de la arquitectura del sistema.

Planificación de la entrega: en esta fase el cliente establece la prioridad de cada HU y los desarrolladores estiman el esfuerzo necesario de cada una.

Iteraciones: esta fase incluye varias iteraciones sobre el sistema antes de ser entregado.

Producción: en esta fase se realizan pruebas adicionales y revisiones de rendimiento antes de que el sistema sea puesto en el entorno del cliente. Se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios que se produzcan durante esta fase.

Mantenimiento: en esta fase se mantiene el sistema funcionando mientras se van desarrollando otras iteraciones.

1.5 Tecnologías usadas para el desarrollo

Se realiza un análisis de las tecnologías utilizadas en el desarrollo del componente de notificaciones. Estas tecnologías fueron previamente seleccionadas por el grupo de arquitectos del proyecto XEGFORT y su uso se considera apropiado para la presente investigación.

1.5.1 Herramientas de la Ingeniería de Software Asistida por Computadora (CASE)

Son varias aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software. Estas ayudan en muchos aspectos a la hora del desarrollo del software en tareas como: el proceso de realizar el diseño del proyecto, cálculos de costes, implementación de partes del código, detección de errores (Ramírez Giraldo, 2013).

Roger Pressman plantea que las herramientas CASE ayudan a los gestores y practicantes de la Ingeniería de Software (IS) en todas las actividades asociadas a los proyectos de software, automatizan las actividades de gestión de proyectos, gestionan todos los productos de los trabajos elaborados a través del proceso y ayudan a los ingenieros en el trabajo de análisis, diseño y codificación (Pressman, 2014).

Características básicas de una herramienta CASE:

- Permiten a los usuarios dibujar diagramas para la planificación, análisis o diseño en la pantalla de un ordenador.
- Solicitan información acerca de cada uno de los objetos de un diagrama y de las interrelaciones entre dichos objetos.
- Comprueban la exactitud, integridad y completitud de cada diagrama. Los diagramas que se ofrezcan deben ser elegido con el objetivo de facilitar esta labor.

Visual Paradigm

Visual Paradigm es una herramienta de Lenguaje Unificado de Modelado (UML) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar (Paradigm, 2015).

Características principales:

- Soporte de UML versión 2.1.
- Licencia gratuita y comercial.
- Compatibilidad entre ediciones.
- Disponibilidad en múltiples plataformas.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.

- Generación de código - Modelo a código, diagrama a código.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Editor de figuras.

UML

UML es ante todo un lenguaje que admite modelar, construir y documentar los elementos que forman un software orientado a objetos; es la continuación de una serie de procesos de análisis y diseño. En este caso, este lenguaje se centra en la representación gráfica de un sistema. Se ha convertido en el estándar de la industria, a pesar de no tener un respaldo formal de una autoridad institucional o un organismo de normalización, es ampliamente usado. UML describe lo que supuestamente realizará un sistema, pero no dice cómo implementarlo. Sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas de software como para la arquitectura hardware donde se ejecuten (Miranda, 2013).

Ventajas:

- Mayor rigor en la especificación.
- Permite realizar una verificación y validación del modelo realizado.
- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto.

1.5.2 Lenguajes de programación

Los lenguajes de programación definen un proceso que es ejecutado en la computadora. Existen desarrollados un significativo número de lenguajes que siguen el concepto de lograr la mayor abstracción posible y facilitar el trabajo del desarrollador. Permite construir nuevas abstracciones que se adapten al dominio del programa (López, 2011).

Java

Para la implementación se selecciona java como lenguaje de programación. Este es un lenguaje de programación de propósito general y como tal es válido para realizar todo tipo de aplicaciones profesionales, concurrente, orientado a objetos y basado en clases, diseñado específicamente para tener pocas dependencias de implementación. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (Groussard, 2012). Incluye una combinación de características que lo hacen único y está siendo adoptado por multitud de fabricantes como herramienta básica para el desarrollo de aplicaciones comerciales de gran

repercusión. Una de las características más importantes es que los programas “ejecutables”, creados por el compilador de Java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos. Las aplicaciones creadas en este lenguaje son susceptibles de contener pocos errores, principalmente porque la gestión de memoria y punteros es realizada por el propio lenguaje y no por el programador (Groussard, 2012).

Los tipos de programas más comunes que se pueden hacer con Java son los *applets* (se ejecutan en el navegador de la máquina cliente) y las aplicaciones (programas que se ejecutan directamente en la Máquina Virtual de Java (JVM)). Otro tipo especial de programa se denomina *servlet* que es similar a los *applets* pero se ejecutan en los servidores Java (Dayne, y otros, 2012).

Características principales:

1. Es simple.
2. Es un lenguaje independiente de la plataforma, lo que significa que los programas desarrollados en Java se ejecutan en cualquier sistema sin cambios.
3. Orientado a objetos, usa clases para organizar el código en módulos.
4. Es multiproceso, porque los programas de Java pueden contener múltiples procesos en ejecución, lo que permite a los programas manejar varias tareas simultáneamente.
5. Tiene un recolector de basura, ya que los programas de Java no se encargan de liberar de memoria los objetos, esto es una tarea del administrador de memoria y el recolector de basura.
6. Es robusto porque el intérprete de Java revisa todos los accesos al sistema dentro de un programa, por esto, los programas desarrollados en Java no pueden tirar el sistema.
7. Es seguro porque el compilador no sólo verifica todos los accesos a memoria, sino que también se asegura que no entren virus en un *applet* en ejecución.

1.5.3 Plataforma

Es todo soporte hardware y/o software que utilizan aplicaciones que sirven para su ejecución.

Java Enterprise Edition (J2EE)

Es una plataforma que habilita el desarrollo, uso efectivo y manejo de aplicaciones multicapas centralizadas en el servidor. J2EE utiliza la plataforma Java Standard Edition (JSE), para tener una completa, estable, segura y rápida plataforma Java en el ámbito de la empresa.

Esta tecnología soporta una gran variedad de tipos de aplicaciones, desde aplicaciones web de gran escala a pequeñas aplicaciones cliente-servidor. Su objetivo principal es crear un simple modelo de desarrollo para aplicaciones empresariales utilizando componentes. En este modelo los componentes utilizan servicios proporcionados por un contenedor, que de otro modo tienen que estar incorporados en el código de la aplicación. La plataforma de J2EE provee un conjunto de

interfaces de programación de aplicaciones (API) de java y servicios necesarios para el soporte de aplicaciones para empresas (Oracle, 2014).

Las aplicaciones J2EE están compuestas de diferentes componentes. Entre se pueden mencionar:

- Las Aplicaciones Clientes y los *Applets* son componentes que se ejecutan del lado del cliente.
- Los componentes *Java Servlet* y la tecnología *JavaServerPages* son componentes web que se ejecutan del lado del servidor.
- Los Enterprise JavaBeans (*beans enterprise*) son componentes de negocio que se ejecutan en el servidor de aplicaciones (Oracle, 2014).

Además de estos componentes principales, J2EE incluye servicios y tecnologías de soporte como:

- Java Naming and Directory Interface (JNDI) proporciona acceso a nombres y directorios.
- Java Transaction API (JTA) proporciona soporte para transacciones a los componentes J2EE.
- Java DataBase Connectivity (JDBC) tecnología que proporciona acceso a sistemas de bases de datos relacionales.
- Remote Method Invocation (RMI) es un protocolo para la invocación de métodos remotos.
- Java Persistence API (JPA) proporciona una API para poder trabajar con el mapeo de clases a partir de esquemas relacionales (Oracle, 2014).

Existen 3 tipos de enterprise java beans:

- *Bean* de sesión (con o sin estado): representa una conversación temporal con un cliente. Pues cuando el cliente termina su ejecución, el bean de sesión y sus datos desaparecen.
- Bean de entidad (manejados por el bean o por el contenedor): representa datos persistentes almacenados en una fila de una tabla/relación de una base de datos. Si el cliente termina o si se apaga el servidor, los servicios subyacentes se aseguran de grabar el bean.
- Bean dirigidos por mensajes: combina las características de un bean de sesión y de un oyente de Java Message Service (JMS), permitiendo que un componente de negocio reciba asíncronamente mensajes JMS (Hinojosa Tinoco, 2013).

J2EE incluye además un API denominada JavaMail que permite leer, componer y enviar correos electrónicos.

JavaMail

Es una API opcional a la versión estándar de Java (J2SDK) que proporciona funcionalidades de correo electrónico, a través del paquete javax.mail. Permite realizar desde tareas básicas como enviar, leer y componer mensajes hasta tareas más sofisticadas como manejar gran variedad de formatos de mensajes y datos, y definir protocolos de acceso y transporte.

JavaMail soporta por defecto los protocolos de envío y recepción SMTP, protocolo de acceso a mensajes de internet (IMAP), protocolo de oficina de correo (POP3) y las versiones seguras de

estos protocolos SMTPS, IMAPS, POP3S. El envío y recepción son independientes del protocolo, aunque se puede usar uno u otro protocolo, según sean las necesidades. Este paquete va unido al uso del paquete JavaBeans Activation Framework (JAF), además viene incorporado a la versión empresarial J2EE (Gálvez Rojas, y otros, 2006).

Posibilidades de JavaMail

JavaMail está diseñada para facilitar las funcionalidades más comunes, también posee características más avanzadas, que permiten sacar el máximo partido a los estándares de correo electrónico. Las funciones más importantes se detallan a continuación (Gálvez Rojas, y otros, 2006):

- Componer y enviar un mensaje: el primer paso que ha de realizar cualquier aplicación que pretenda enviar un correo electrónico es componer el mensaje. Es posible crear un mensaje de texto plano, es decir, un mensaje sin adjuntos que contenga exclusivamente texto formado por caracteres ASCII; pero es igualmente sencillo componer mensajes más completos que contengan adjuntos. También se pueden componer mensajes que contengan código de lenguaje de marcas de hipertexto (HTML) e incluso imágenes embebidas en el texto.
- Descargar Mensajes: se pueden descargar los mensajes desde la carpeta que se indique ubicada en el servidor que en ese momento se esté usando. Estos mensajes pueden ser de texto plano, contener adjuntos, HTML.
- Usar banderines: para indicar, por ejemplo, que un mensaje ha sido leído o borrado, en función de cuáles de estos banderines estén soportados por el servidor.
- Establecer una conexión segura: actualmente algunos servidores de correo requieren de una conexión segura, ya sea para descargar el correo almacenado en ellos o para enviar mensajes a través de ellos. JavaMail ofrece la posibilidad de establecer este tipo de conexiones, indicando que se trata de una conexión segura, además de poder indicar otros parámetros, en algunos casos necesarios, como el puerto donde establecer la conexión. Esta capacidad está disponible desde la versión de JDK 1.3.2.
- Autenticarse en un servidor: ciertos servidores requieren autenticación ya no sólo para leer sino también para enviar. JavaMail ofrece también la posibilidad de autenticación a la hora de enviar un correo.
- Definir protocolos: JavaMail soporta por defecto los protocolos de envío y recepción SMTP, IMAP, POP3 (y sus correspondientes seguros a partir de la versión de JDK 1.3.2).
- Manejar adjuntos: JavaMail ofrece la posibilidad de añadir adjuntos a los mensajes de salida, así como obtener y manipular los adjuntos contenidos en un mensaje de entrada.
- Acuse de recibo y prioridad: se puede incluir un acuse de recibo en los mensajes de salida, para que el remitente sepa si un mensaje ha sido leído o no por el destinatario, si

bien no todos los servidores soportan esta funcionalidad. Además se puede establecer la prioridad de un mensaje de forma muy sencilla (Gálvez Rojas, y otros, 2006).

1.5.4 Herramientas de desarrollo

Para el desarrollo del componente se seleccionó un conjunto de herramientas las cuales se exponen a continuación:

Glassfish

Es un servidor de aplicaciones desarrollado por *Sun Microsystems* que implementa tecnologías definidas en la plataforma J2EE y permite ejecutar aplicaciones que siguen esta especificación. Soporta las últimas versiones de tecnologías como: Java ServerPages (por sus siglas en inglés JSP), Servlets y Enterprise JavaBeans (conocida por sus siglas en inglés EJB).

Proporciona generalmente gran cantidad de funcionalidades *built-in* de forma transparente al usuario de manera que no sea necesario escribir código fuente. Estas funcionalidades son posibles ya que los componentes se ejecutan dentro del contenedor en un espacio de ejecución virtual llamado dominio de ejecución. Su función principal es la de interponerse entre las llamadas que se hacen a los métodos de los *beans* y las implementaciones de los mismos, de modo que pueda hacer las comprobaciones para verificar si el usuario que llama el método tiene los permisos adecuados, antes de llamarlo (Oracle Corporation, 2011).

Dispone de una arquitectura modular basado en el modelo de componentes dinámico y completo para Java Iniciativa de Pasarela de Servicio Abierto (por sus siglas en inglés OSGi). Las aplicaciones y/o componentes pueden remotamente instalarse, iniciarse, actualizarse sin necesidad de reiniciar el servidor. Es posible usar Glassfish como una biblioteca más en la JVM, seleccionado solo lo que se necesita y probando pequeñas aplicaciones web sin necesidad de correr todo el *AppServer*, teniendo en cuenta las limitaciones de no estar instalado (Oracle Corporation, 2011).

Swing

Swing es un conjunto de clases desarrolladas por primera vez para java 1.2 para mejorar el anterior paquete que implementaba clases para fabricar interfaces de usuario. Viene incluida con el entorno de desarrollo de Java. Actualmente es una extensión de la herramienta abstracta de ventana (por sus singlas en inglés AWT) añadiendo las siguientes características (Montenegro, 2012):

- Componentes Swing.
- Soporte de Look & Feel.
- API de accesibilidad.
- Java 2D API.
- Soporte de Drag & Drop.

Swing sigue trabajando con los conceptos de la AWT:

1. Contenedores.

2. Componentes.
3. Administrador de plantillas.
4. Eventos.

Con Swing, se pueden diseñar interfaces con componentes de árboles, tablas, cuadros de diálogo con fichas, información sobre herramientas y un conjunto cada vez mayor de otras características que los usuarios están acostumbrados.

Swing no se basa en componentes nativos de la plataforma en tiempo de ejecución. Está escrito completamente en Java y crea sus propios componentes. Este enfoque ha resuelto la mayoría de los problemas de portabilidad ya que los componentes no heredan comportamientos extraños desde el entorno de ejecución. La oscilación está en completo control de los componentes, que se encuentra en el control de la forma en que los componentes se ven en la pantalla y le da más control sobre la apariencia de sus aplicaciones. Swing hace una distinción muy clara entre los datos de un componente pantallas (el "modelo") y la visualización real (la "vista") (Montenegro, 2012).

NetBeans

El IDE NetBeans es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans es una aplicación que permite a los desarrolladores crear con rapidez aplicaciones web, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, procesador de hipertexto (PHP), Java Script y Ajax, Ruby y Ruby on Rails, Groovy, Grails y C/C++. El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos, así como una amplia selección de complementos de terceros. Incluye módulos tales como un editor, herramientas para trabajar con el código fuente (Java, Hoja de estilo en cascada (CCS), HTML) y control de versiones. NetBeans incluye herramientas para la gestión de dependencias y construcción de proyectos como es el caso de Maven (Netbeans, 2015).

Maven

Es una herramienta de código abierto creada en 2001 con el objetivo de simplificar los procesos de compilar y generar ejecutables a partir del código fuente. Es una excelente herramienta para desarrolladores Java y también es posible usarla para gestionar el ciclo de vida de sus proyectos. Como herramienta de gestión de ciclos de vida funciona en torno a fases y no con el desarrollo estilo Ant por "tareas". Maneja todas las fases del ciclo de vida de los proyectos, incluyendo validación, generación de código, compilación, pruebas, empaquetamiento, pruebas de integración, verificación, instalación, despliegue y creación e implementación de sitios.

La gestión de dependencias entre módulos y distintas versiones de bibliotecas se hace muy sencilla. En este caso, solo se debe indicar los módulos que componen el proyecto o qué bibliotecas utiliza el software que se está desarrollando en un fichero de configuración de Maven del proyecto llamado POM.

Además, en el caso de las bibliotecas, no hay necesidad de descargarlas a mano. Posee un repositorio remoto (Maven central) donde se encuentran la mayoría de bibliotecas que se utilizan en los desarrollos de software y que la propia herramienta se descarga cuando sea necesario (Zara, 2012).

PostgreSQL

Es un sistema de administración de bases de datos relacionales orientadas a objetos (por sus siglas en inglés, ORDBMS) basado en POSTGRES, versión 4.2, desarrollado en el Departamento de Ciencias Computacionales de la Universidad de California, Berkeley. POSTGRES fue pionero en muchos conceptos que solo llegaron a aparecer en algunos sistemas de bases de datos comerciales mucho tiempo después. PostgreSQL es un descendiente de código abierto del código original de Berkeley (Group, 2015).

Características principales:

- Bases de datos de nivel empresarial.
- Multiplataforma: se ejecuta en los sistemas operativos más utilizados, incluyendo GNU/Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows.
- Incluye la mayoría de tipos de datos de SQL: 2008, como INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP.
- Almacenamiento de objetos binarios grandes, incluyendo imágenes, sonido y video.
- Documentación excepcional.
- Transacciones anidadas.
- Optimizador/Planificador de consultas sofisticado.
- Conjuntos de caracteres internacionales.
- Soporta Unicode.
- Y por su licencia libre, cualquier persona puede usar, modificar y distribuir PostgreSQL de manera libre y gratuita para cualquier propósito, sea privado, comercial o académico.

1.6 Métricas de validación del diseño

El IEEE “Standard Glossary of Software Engering Terms” define el término de métrica como “una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado” (IEEE, 2016). Esto permite evaluar la calidad durante el desarrollo del sistema. Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo (García Sanchez, 2010).

Las métricas están diseñadas a evaluar los siguientes atributos de calidad:

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelo de dominio o concepto, de la problemática propuesta.

- **Complejidad de implementación:** consiste en el grado de dificultad que tiene implementado un diseño de clases determinado.
- **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento:** consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de reutilización.
- **Complejidad del mantenimiento**: consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirectamente, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas:** consiste en el número de o el grado de esfuerzo para realizar pruebas de calidad (Unidad) del producto (componente, módulo, clase).

Para evaluar el diseño se utilizan diferentes métricas, atendiendo a las características y el lenguaje de programación que se esté empleando. Existen dos conjuntos de métricas fundamentales: Chidamber y Kemerer y el conjunto de Lorenz y Kidd. Del primer conjunto se utiliza Árbol de Profundidad de Herencia (APH) para medir la profundidad de herencia del componente, la métrica Acoplamiento entre Clases Objeto (ACO) para validar una de las variables del problema y del segundo grupo Tamaño Operacional de las Clases (TOC) y Relaciones entre Clases (RC) para ver si están bien distribuidas las responsabilidades entre clases. Se hace esta selección debido una característica de la solución: es un componente; por lo que es fundamental tener valores favorables para atributos como reutilización, complejidad y acoplamiento.

Métrica ACO

ACO mide el acoplamiento entre clases de objetos. Una clase esta acoplada a otra si los métodos de una clase usan métodos o atributos de la otra y viceversa. Dada esta definición, los usos pueden significar el tipo de miembro, el tipo de parámetro, variable de método local o el llamado a métodos de una clase. ACO es el número de clases con la que una clase dada esta acoplada. Incluye el acoplamiento basado en herencia (es decir, acoplamiento entre clases relacionadas vía herencia).
A medida que los valores crecen, es probable que la reusabilidad de la clase vaya descendiendo. Valores altos complican también las modificaciones y la comprobación que se produce cuando se efectúan modificaciones.

Métrica APH

Esta mide el nivel máximo en la jerarquía de herencia de clase; la raíz del árbol de herencia como no hereda de nadie está en el nivel cero. Ese es el conteo directo en una jerarquía de herencia. De acuerdo a Chidamber y Kemerer esta métrica se hizo con el propósito de medir la complejidad de la clase, la complejidad del diseño y el potencial de reutilización. Mientras más profunda es la clase, mayor será el número de métodos que debe heredar.

Una profundidad de herencia demasiado grande indica objetos complejos que pueden dificultar las pruebas y la reutilización. Lorenz y Kidd sugieren con un umbral de 6 niveles, indica una herencia excesiva para proyectos en C++. Para Java se agrega un nivel atendiendo que todas las clases herendan de la clase Object (Naboulsi, 2011).

Métrica RC

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- **Acoplamiento:** un aumento del RC implica un aumento del Acoplamiento de la clase.
- **Complejidad de mantenimiento:** un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** un aumento del RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para estos atributos están definidos los siguientes criterios y categorías de evaluación:

Tabla 2. Criterios y categorías para evaluar la métrica (RC).

Atributo	Categoría	Criterio
Acoplamiento	Ninguna	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<= Promedio
	Media	Entre promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre promedio y 2*Promedio
	Alta	>2*Promedio

Métrica TOC

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

- **Responsabilidad:** un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.

- **Complejidad de implementación:** un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización:** un aumento del TOC implica una disminución del grado de reutilización de la clase.

Para estos atributos de calidad están definidos los siguientes criterios y categorías de evaluación:

Tabla 3. Rango de valores para los criterios de evaluación de la métrica (TOC).

Atributo	Categoría	Criterio
Responsabilidad	Baja	<= Promedio
	Media	Entre promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<= Promedio
	Media	Entre promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre promedio y 2*Promedio
	Alta	<=Promedio

1.7 Pruebas

Las pruebas del software implican ejecutar una implementación del software con datos de prueba. Se examinan las salidas del software y su entorno operacional para probar que funcione tal y como se requiere. Las pruebas son una técnica dinámica de verificación y validación. Pueden ejecutarse en cualquier punto del proceso de desarrollo de software.

El objetivo de la etapa de prueba de componentes es descubrir defectos probando componentes de programas individuales. Estos componentes pueden ser funciones, objetos o componentes reutilizables. Durante estas pruebas, estos componentes se integran para formar subsistemas o el sistema completo. En esta etapa la prueba debería centrarse en establecer que el sistema satisface sus requisitos funcionales y no se comporta de forma inesperada (Somerville, 2015).

Sus dos objetivos principales son:

- Demostrar al desarrollador y al cliente que el software satisface sus requisitos.
- Descubrir defectos en el software en que el comportamiento de este es incorrecto, no deseable o no cumple su especificación.

Niveles de prueba (Zapata, 2013)

Pruebas unitarias o de componentes: son ejecutadas normalmente por el equipo de desarrollo, consisten básicamente en la ejecución de actividades que le permiten al desarrollador verificar que los componentes unitarios están codificados bajo condiciones de robustez.

Prueba de integración: es una técnica sistemática que se utiliza para confeccionar la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados con la interfaz.

Pruebas de Sistema: deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo. Este equipo deberá ejecutar actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementado de acuerdo a los documentos de especificación definidos en el proyecto.

Pruebas de Aceptación: independientemente de que se haya revisado el sistema por otro equipo y el responsable de esta actividad haya emitido un certificado de calidad sobre el sistema objeto de prueba, se hace indispensable, que el cliente designe a personal que haga parte de los procesos de negocio para la ejecución de las 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.

Para validar el componente realizado se le aplicarán las pruebas unitarias que son especificadas por la metodología de desarrollo, utilizando el método de caja negra y caja blanca. Las pruebas de aceptación serán realizadas por el cliente y que estarán centradas en las funcionalidades del componente.

Método de caja negra

Estas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa.

Pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Tipos de errores que se pretenden encontrar:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a base de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Técnicas de prueba de caja negra:

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

- Grafos de causa: le permite a la persona encargada de la prueba validar un conjunto de acciones y condiciones complejas (Edo, 2011).

Método de caja blanca

Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de pruebas.

Estas prueban intentan garantizar:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas (Carabali, 2013).

Para ejecutar caja blanca se utiliza la técnica de camino básico. Esta permite al diseñador de los casos de prueba, obtener una medida de la complejidad lógica del diseño y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

1.8 Conclusiones parciales

El estudio de los sistemas similares existentes demostró que estos no cumplen con las necesidades planteadas por lo que se hace necesario implementar un componente que se ajuste a las necesidades de XEGFORT. Este desarrollo será guiado por la metodología XP y se soporta en herramientas como: Maven como gestor de dependencias, Java como lenguaje de programación y Glassfish como servidor de aplicaciones favoreciendo el correcto desarrollo de la solución propuesta.

Capítulo 2: Componente para la gestión de notificaciones para el marco de trabajo XEGFORT.

2.1 Introducción

En el presente capítulo se propone la solución técnica de la investigación. Para lograr una mejor comprensión de la solución se describen las funcionalidades del componente. Se analizan los patrones de diseño utilizados en la solución propuesta relacionados con la arquitectura del marco de trabajo. Se exponen los requisitos funcionales y no funcionales permitiendo un mejor entendimiento del componente.

2.2 Requisitos

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de dispositivos, hacer un pedido o encontrar información. En algunos casos, un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este (Somerville, 2015).

Para la captura de requisitos se utilizaron las técnicas descritas a continuación:

- Tormenta de ideas: son reuniones en grupos con el objetivo de que cada participante muestre su idea libremente. Para esta se realizan talleres con los integrantes del equipo de desarrollo, así como sus respectivos tutores con el objetivo de debatir del tema y definir sus principales funcionalidades.
- Entrevista con el cliente: se realizar reuniones con el cliente para conocer sus exigencias con mayor claridad y sustentar las bases para la captura de los requisitos.

2.2.1 Requisitos funcionales (RF)

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Somerville, 2015).

Con la participación del equipo de desarrolladores y el cliente a través de las técnicas entrevista con el cliente y tormenta de ideas se identificaron los siguientes RF que debe cumplir el componente:

RF1 Adicionar cuentas de correo electrónico.

RF2 Eliminar cuentas de correo electrónico.

RF3 Modificar cuentas de correo electrónico.

RF4 Listar cuentas de correo electrónico.

RF5 Mostrar cuentas de correo electrónico.

RF6 Envío de correo electrónico.

RF7 Configurar envío de correo programado.

RF8 Crear plantillas de correos a partir de entidades de dominio.

RF9 Adicionar configuración de servidor de correo electrónico.

RF10 Eliminar configuración de servidor de correo electrónico.

RF11 Modificar configuración de servidor de correo electrónico.

RF12 Reporte de estado de envío de correos electrónicos por rango de fecha.

RF13 Reporte de estado de envío de correos electrónicos por plantilla.

2.2.2 Requisitos no funcionales (RNF)

Son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad (Somerville, 2015). A continuación son descritos los requisitos no funcionales que formaran parte de este componente.

Requisitos de Usabilidad

RNF1: la herramienta siempre tendrá visible la opción de Ayuda, lo que posibilita un mejor aprovechamiento de sus funcionalidades.

RNF2: la herramienta debe proporcionar mensajes que sean informativos y orientados al usuario final.

Requisito de Software

RNF3: se necesita tener instalado JDK en su versión 1.8.60 o superior a esta, Glassfish en su versión 4.1, JPA en su versión 2.1, EJB en su versión 3.1 y JavaMail en su versión 1.4.4.

2.3 Historias de usuarios (HU)

Son técnicas utilizadas en XP para especificar las tareas que debe realizar el sistema. Son escritas en lenguaje natural y guían la construcción de las pruebas de aceptación, además son utilizadas para estimarle tiempo de desarrollo. A continuación se muestran las HU para el componente.

Tabla 4. HU Adicionar cuentas de correo electrónico.

Número: 1	**Nombre del requisito**: Adicionar cuentas de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 8 horas

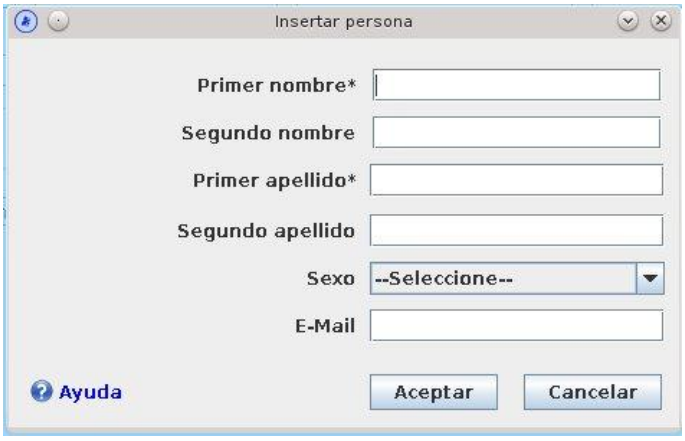
Descripción: Permite registrar una persona con su cuenta de correo electrónico teniendo en cuenta los siguientes datos: Primer nombre (obligatorio): cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula. Segundo nombre: cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula. Primer apellido (obligatorio): cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula. Segundo apellido: cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula. Sexo: campo seleccionable (Masculino/Femenino). E-mail: cadena de caracteres con formato usuario@domino.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 5. HU Eliminar cuentas de correo electrónico.

Número: 2	**Nombre del requisito:** Eliminar cuentas de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 6 horas

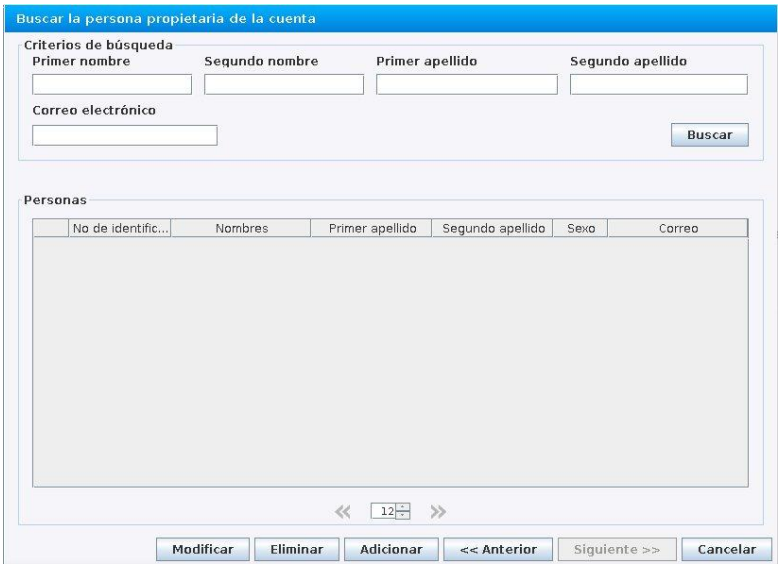
Descripción: Permite eliminar una cuenta de correo electrónico de un usuario determinado.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 6. HU Modificar cuentas de correo electrónico.

Número: 3	**Nombre del requisito:** Modificar cuentas de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 6 horas

Descripción:

Permite registrar una persona con su cuenta de correo electrónico teniendo en cuenta los siguientes datos:

Primer nombre (obligatorio): cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula.

Segundo nombre: cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula.

Primer apellido (obligatorio): cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula.

Segundo apellido: cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula. Sexo: campo seleccionable (Masculino/Femenino). E-mail: cadena de caracteres con formato usuario@domino.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

Tabla 7. HU Listar cuentas de correo electrónico.

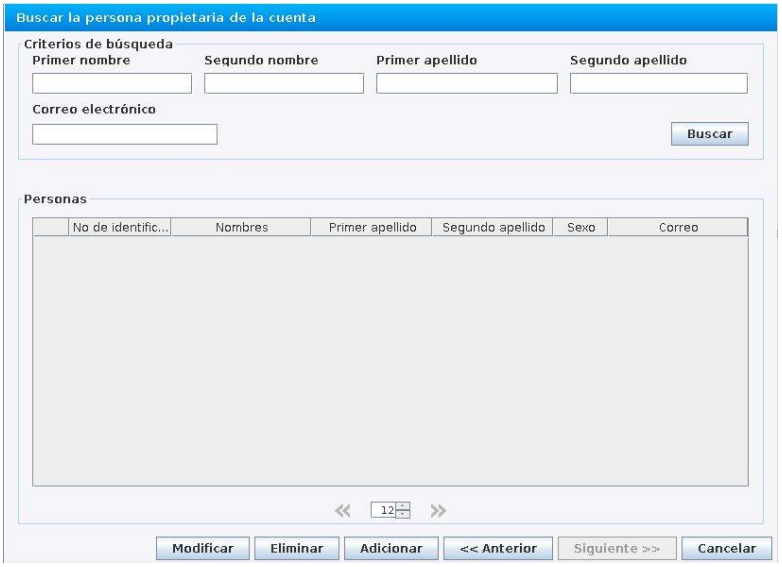
Número: 4	**Nombre del requisito:** Listar cuentas de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 8 horas
Descripción: Lista todas las cuentas de correos que se encuentran en el correo electrónico.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 8. HU Mostrar cuentas de correo electrónico.

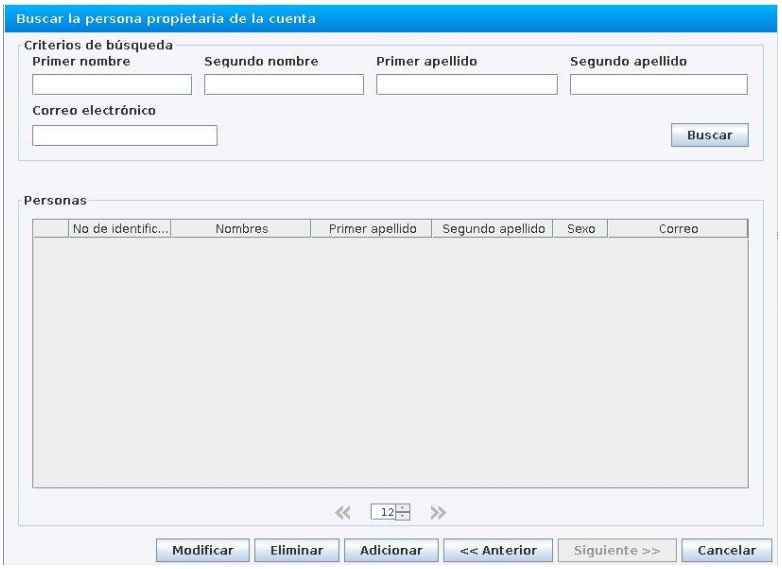
Número: 5	**Nombre del requisito:** Mostrar datos de cuentas de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada: 2**
Prioridad: Baja	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 8 horas
Descripción: Permite mostrar todos los datos de una cuenta de correo electrónico.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 9. HU Envío de correo electrónico.

Número: 6	**Nombre del requisito:** Envío de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 40 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 40 horas
Descripción:	

Destinatario (obligatorio): dirección de correo electrónico hacia el usuario que se desea enviar el correo. Contenido: cadena de caracteres que admite letras y números, puede contener un mínimo de 15 caracteres, debe empezar con mayúscula.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario: N/A

Tabla 10. HU Configurar envío de correo programado.

Número: 7	**Nombre del requisito:** Configurar envío de correo programado.
Programador: Esther García Naranjo.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 30 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 25 horas
Descripción: Destinatario (obligatorio): dirección de correo electrónico hacia el usuario que se desea enviar el correo. Hora: cadena de caracteres que presenta el siguiente formato HH: MM: SS. HH: representa la hora que debe ser válido desde las 01 hasta las 12. MM: representa los minutos que debe ser válido desde el 00 hasta el 59. SS: representa los segundos que debe ser válido desde el 00 hasta 59. Fecha: cadena de caracteres que presenta el siguiente formato AA MM DD. AA: representa el ano que debe ser válido desde 2015 hasta 3000. MM: representa los meses de un año que debe ser válido desde 01 hasta 12. DD: representa los días de un mes que debe ser válido desde 01 hasta 31 tener en cuenta que para el mes 02 hasta 28 y cada 4 años 29. Contenido: cadena de caracteres que admite letras y números, puede contener un mínimo de 15 caracteres, debe empezar con mayúscula.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: N/A	

Tabla 11. HU Crear plantillas de correos a partir de entidades de dominio.

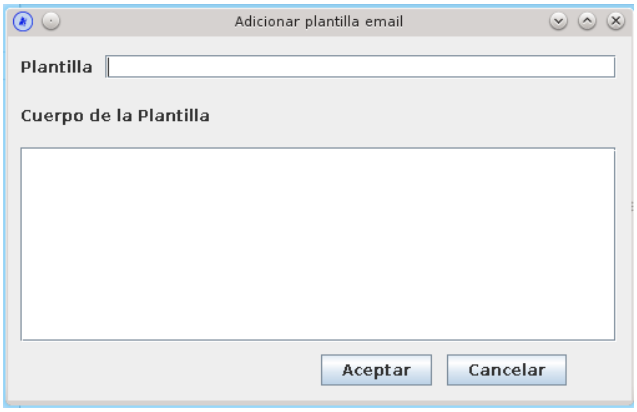
Número: 8	**Nombre del requisito:** Redactar plantillas de correos a partir de entidades de dominio.
Programador: Esther García Naranjo.	**Iteración Asignada: 2**
Prioridad: Baja	**Tiempo Estimado:** 32 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 32 horas
Descripción: Permite registrar una plantilla de correo electrónico teniendo en cuenta los siguientes datos: Plantilla (obligatorio): cadena de caracteres que solo admite letras, no máximo de 250 caracteres comenzando con mayúscula. Cuerpo de la Plantilla: cadena de caracteres.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 12. HU Adicionar configuración servidor de correo electrónico.

Número: 9	**Nombre del requisito:** Adicionar configuración de servidor de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 24 horas

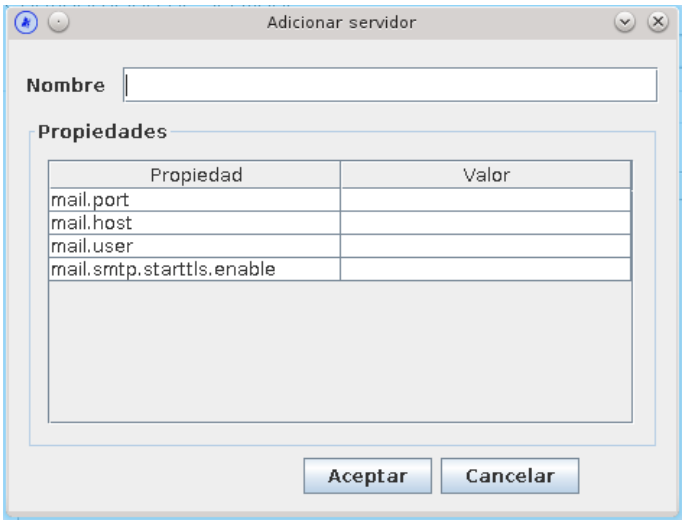
Riesgo en Desarrollo: N/A	**Tiempo Real:** 30 horas
Descripción: Permite adicionar un servidor de correo electrónico teniendo en cuenta los siguientes datos. Nombre: cadena de caracteres que solo admite letras no máximo de 250 caracteres que debe empezar con mayúscula. mail.host: cadena de caracteres que solo admite números y el caracter punto. mail.puerto: cadena de caracteres que solo admite números, no máximo de 4 caracteres. mail.user: cadena de caracteres que admite solo letras y números, no máximo de 15 caracteres. mail.smtp.starttls.enable: cadena de caracteres que solo admite true o false.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 13. HU Eliminar configuración de correo electrónico.

Número: 10	**Nombre del requisito:** Eliminar configuración de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 24 horas

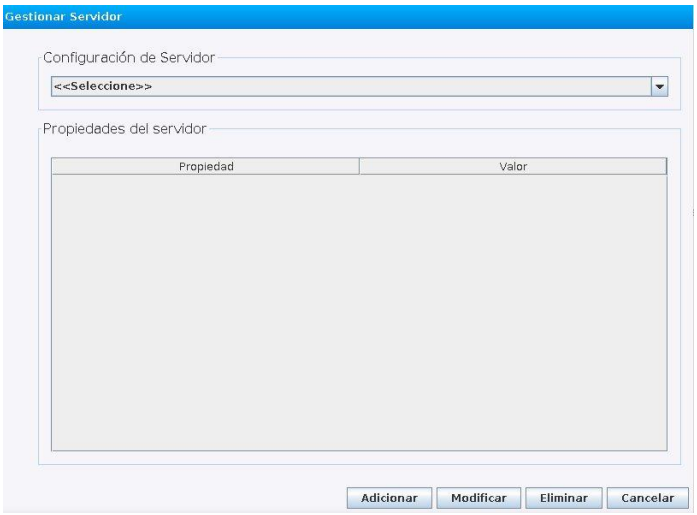
Riesgo en Desarrollo: N/A	**Tiempo Real:** 20 horas
Descripción: Permite eliminar un servidor de correo electrónico, seleccionando el nombre de la configuración deseada.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 14. HU Modificar configuración de correo electrónico.

Número: 11	**Nombre del requisito:** Modificar configuración de correo electrónico.
Programador: Esther García Naranjo.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 20 horas
Descripción: Permite modificar un servidor de correo electrónico teniendo en cuenta los siguientes datos. Nombre: cadena de caracteres que solo admite letras no máximo de 250 caracteres que debe empezar con mayúscula. mail.host: cadena de caracteres que solo admite números y el caracter punto.	

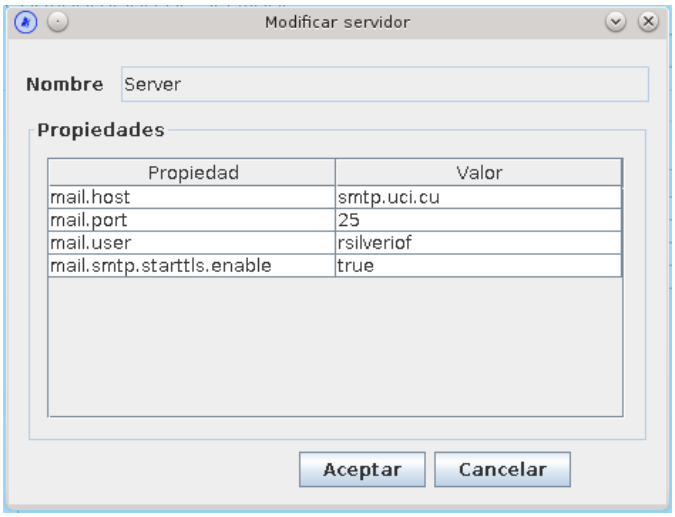
mail.puerto: cadena de caracteres que solo admite números, no máximo de 4 caracteres. mail.user: cadena de caracteres que admite solo letras y números, no máximo de 15 caracteres. mail.smtp.starttls.enable: cadena de caracteres que solo admite true o false.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 15. HU Reporte de estado de envío de correo electrónico por rango de fecha.

Número:12	**Nombre del requisito:** Reporte de estado de envío de correo electrónico por rango de fecha.
Programador: Esther García Naranjo.	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 8 horas
Descripción: Muestra todos los correos agrupados por un rango de fecha.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

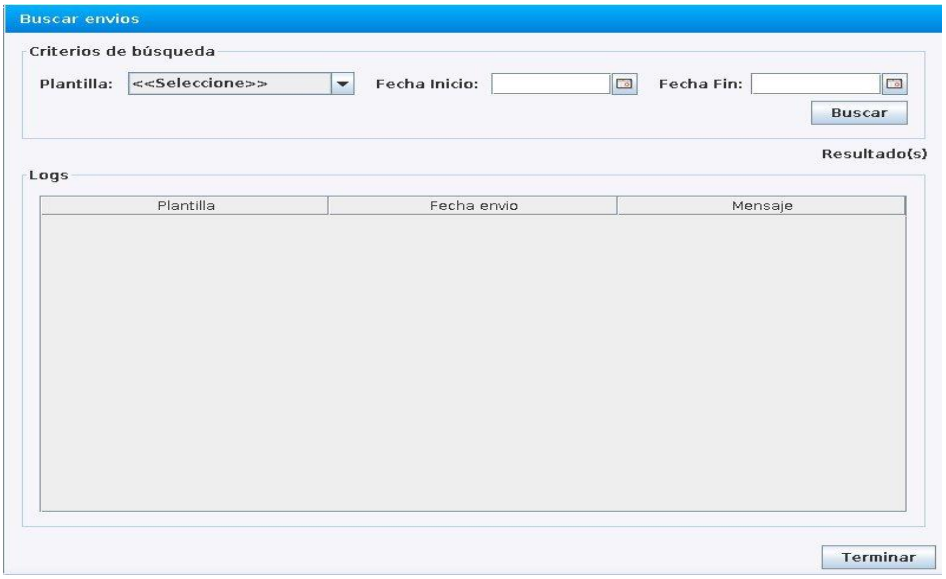
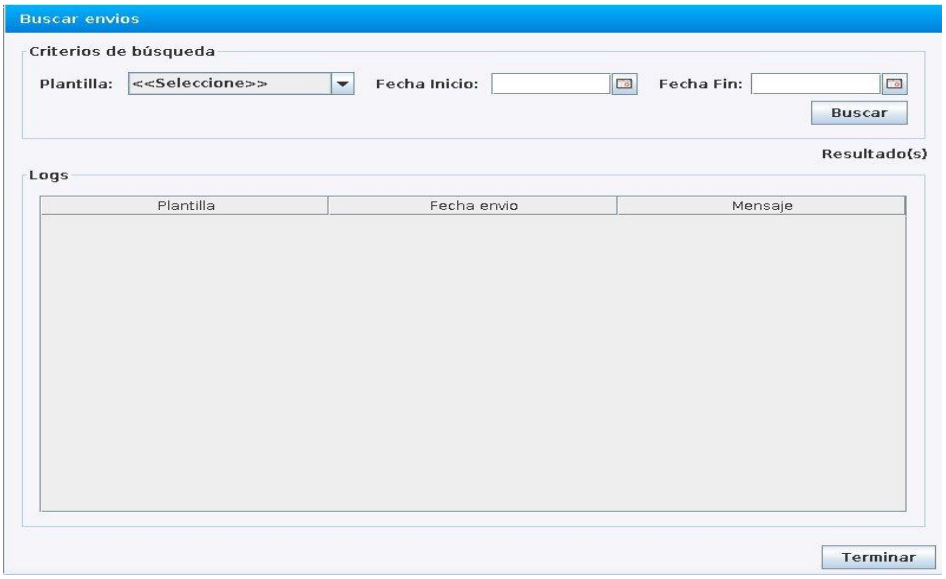


Tabla 16. HU Reporte de estado de envío de correos electrónicos por plantilla.

Número:13	**Nombre del requisito:** Reporte de estado de envío de correos electrónicos por plantilla.
Programador: Esther García Naranjo.	**Iteración Asignada:** 2
Prioridad: Baja	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 10 horas
Descripción: Muestra todos los correos agrupados por plantilla.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	



2.4 Estimación de esfuerzo por HU

Para lograr un desarrollo eficiente y satisfactorio, se realizó una estimación de esfuerzos para cada una de las HU identificadas en el proceso de planificación. La misma fue establecida por parte de los programadores utilizando como medida el punto, este equivale a una semana de programación. Una HU puede valer de 1 a 3 puntos, esto se da en dependencia de la capacitación y/o capacidad del programador que la realice o la carga del trabajo del equipo. A continuación se muestran los resultados en la Tabla 17.

Tabla 17. Estimación de esfuerzo.

Historias de usuarios	Puntos de estimación
Adicionar cuentas de correo electrónico.	1
Eliminar cuentas de correo electrónico.	1
Modificar cuentas de correo electrónico.	1
Listar cuentas de correo electrónico.	1
Mostrar datos de cuenta de correo electrónico.	1
Envío de correo electrónico.	1
Configurar envío de correo programado.	1
Redactar plantillas de correo a partir de entidades de dominio.	1
Adicionar configuración de servidor.	1
Eliminar configuración de servidor.	1
Modificar configuración de servidor.	1
Reporte estado de envío de correos electrónicos por rango de fecha.	1
Reporte de estado de envío de correo electrónico por plantilla.	1

2.5 Plan de iteraciones

Una vez que son identificadas las HU del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas, se procede a definir qué HU será implementada para cada iteración, dividido en 2 iteraciones.

Iteración 1: en la iteración número 1 se realiza el desarrollo de la HU 6 a la 11. Esta iteración es la de mayor complejidad en todo el desarrollo por lo que posee gran prioridad para el cliente.

Iteración 2: la iteración número 2 se encarga de la implementación de las HU de la 1 a la 5 y las HU 12 y 13.

2.6 Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto utilizando la metodología XP se crea el plan de duración de cada una de las iteraciones. Este plan se encarga de mostrar las HU que son complementadas en cada una de las iteraciones, así como la duración estimada de cada una y el orden en que se implementan.

Tabla 18. Plan de duración de las Iteraciones.

Iteración	Orden de la HU a implementar	Duración total por semanas.
1	Envío de correo electrónico. Configurar envío de correo programado. Adicionar configuración de servidor. Eliminar configuración de servidor. Modificar configuración de servidor.	8
2	Adicionar cuentas de correo electrónico. Modificar cuentas de correo electrónico. Eliminar cuentas de correo electrónico. Listar cuentas de correo electrónico. Mostrar cuentas de correo electrónico. Redactar plantillas de correo a partir de entidades de dominio. Reporte estado de envío de correos electrónicos por rango de fecha. Reporte de estado de envío de correo electrónico por plantilla.	6

2.7 Plan de entrega

En el plan de entrega se definen las HU que se entregan al final de cada iteración. A continuación se muestran las planificaciones finales que son publicadas.

Tabla 19. Plan de duración de las Iteraciones.

Iteración	Fecha de entrega
Iteración I	15/5/2016

Iteración II	20/6/2016

2.8 Arquitectura del componente

"La arquitectura de un sistema es la estructura del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente y las relaciones entre ellos" (Somerville, 2015).

El estilo arquitectónico que utiliza el marco de trabajo XEGFORT, es el Estilo N capas. Se identificaron 4 niveles o capas fundamentales distribuidas entre el cliente y el servidor como se describen a continuación:

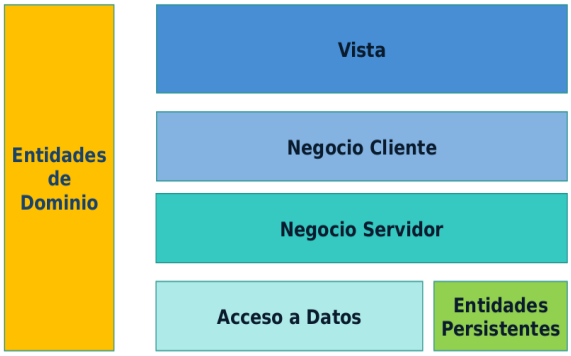


Figura 1. Modelo de arquitectura (Fuente: elaboración propia).

En el cliente se encuentran las capas de la Vista y Negocio Cliente, las cuales contienen las acciones, los formularios y los gestores de negocio del cliente. Las acciones se encargan de manejar la información existente en los formularios, permitiendo que los gestores de negocio del cliente realicen las validaciones necesarias.

En el servidor se encuentra la capa de Negocio Servidor, la de Acceso a Datos y las Entidades Persistentes. La primera contiene los gestores de negocio del servidor que se encargan de realizar validaciones más cercanas al negocio, delegando en los gestores de acceso a datos la persistencia de la información obtenida, a través de las entidades persistentes.

Transversal a ellas se encuentran las Entidades de Dominio que son las responsables de transportar la información desde el servidor hasta el cliente.

2.8.1 Patrones de diseño

Los patrones de diseño son soluciones a problemas repetidos en la construcción de software, y en ocasiones pueden incluir sugerencias para aplicar estas soluciones en diversos entornos. En el diseño que propone este trabajo, se utilizaron algunos patrones de diseño GOF y GRASP, para solucionar y/o evitar diferentes problemas durante el desarrollo de la solución.

Patrones GRASP

Patrones de Asignación de responsabilidades (por sus siglas en inglés *General Responsibility Assignment Software Patterns*, GRASP), este describe los principios fundamentales para asignar responsabilidades a los objetos. Esta responsabilidad consta de obligaciones o contratos de una clase que describe que comportamiento va a satisfacer.

Experto: es la clase que tiene la información necesaria para cumplir la responsabilidad. Este patrón se evidencia en la capa de negocio, un (EJB) que se encarga de manejar todo lo relacionado con el usuario como es el *GestorNSUsuario*.

Creador: expresa que los objetos deben hacer las cosas relacionadas con la información que poseen, expertos en crear instancias de otros objetos. En el componente se utiliza principalmente en la capa de Acceso a Datos, los *Converters* crean instancias de entidades de dominios y entidades persistentes.

Controlador: especialista experto que está encargado de controlar el flujo. Este se evidencia en las acciones (ver Anexo 6).

Bajo acoplamiento: asigna responsabilidades de manera que el acoplamiento permanezca bajo. El gestor de Negocio del Servidor de usuario utiliza Fachada para conectarse al servidor Gestor de Negocio del Servidor.

Alta cohesión: asigna responsabilidades de manera que la cohesión se mantenga alta. Encargado de que la clase tenga el número adecuado de dependencias para realizar. En el componente este se evidencia en la clase *GestorMarcoTrabajo*.

Patrones GoF

Instancia única (*Singleton*): este patrón asegura que solo se cree una instancia de la clase y provee un punto global de acceso a esta. Este patrón es muy útil cuando se quiere tener solamente un objeto único instanciado. En el componente se evidencia en la clase *GestorMarcoTrabajo.*

Factoría (*Factory*): su propósito es definir una interfaz para crear objetos donde se delega la creación de las instancias a las subclases. Crea objetos de una clase. Se evidencia en la clase FactoriaFachada pues es la que contiene la lógica de instanciación para crear una fachada.

Factoría Abstracta (*Abstact Factory*): este patrón proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas. En este caso el FactoriaFachada que crea instancias de actividades distintas según la configuración que se le pase.

Observador (*Observer*): define una dependencia de uno a muchos entre objetos así que cuando un objeto cambia de estado, todos los objetos que dependen de él son notificados para que se actualicen automáticamente. Se usa entre cliente-contexto y las diferentes ventanas de configuración que se utilizan cuando inicia el marco de trabajo.

Mediador (Mediator): la clase GestorMarcoTrabajo, le quita responsabilidades a las acciones.

2.9 Diseño de Base de Datos

Una base de datos permite almacenar información de forma ordenada con diferentes propósitos y usos en un sistema determinado. El diseño de esta consiste en definir la estructura de los datos que debe tener la misma en un sistema de información determinado.

A continuación se muestra el diagrama de base de datos asociado a las modificaciones componente. Las tablas destacadas en color azul fueron las que se agregaron para darle solución al sistema. Entre ellas se encuentra la tabla *t_propiedad_servidor* que almacena los datos y valores asociados a las propiedades del servidor de envío de correo. La tabla *t_email* que guarda los datos referentes al correo de la persona. La tabla *t_servidor_envio* guarda la denominación del servidor de envío al cual se asociarán los valores y las propiedades almacenadas en la tabla *t_valor_propiedad_servidor* y la tabla *n_plantilla* y *t_plantilla*. Fueron creadas para almacenar las plantillas de email que se utilizarán en el momento de hacer el envío.

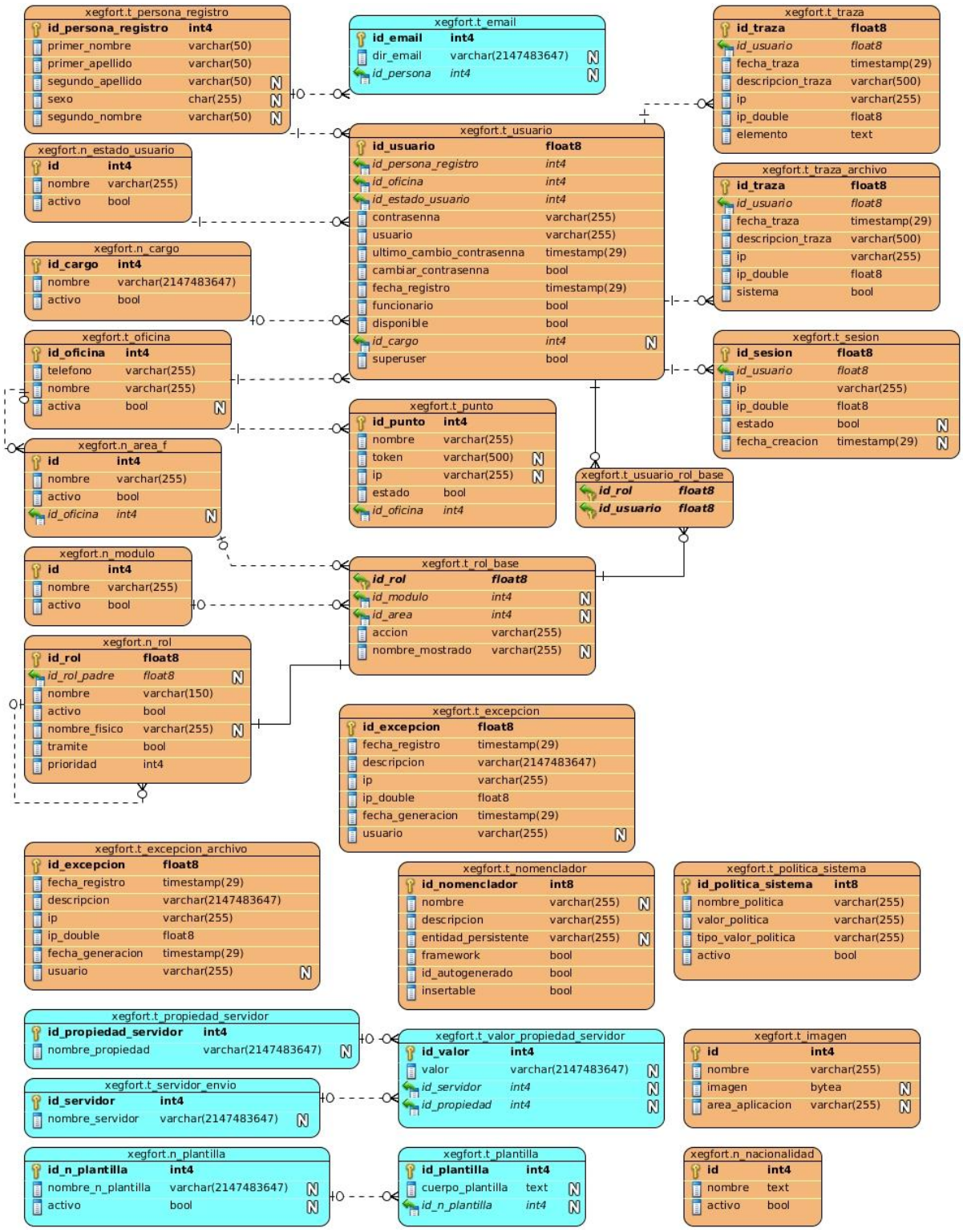


Figura 2. Diagrama de Base de Datos (Fuente: elaboración propia).

2.10 Diagrama de paquetes

Durante el desarrollo de software resulta muy conveniente agrupar clases y ficheros por diferentes criterios que ayudan a una fácil comprensión del componente, resultando conveniente el desarrollo de los diagramas de paquetes.

Un diagrama de paquetes en el UML representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica del sistema. Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

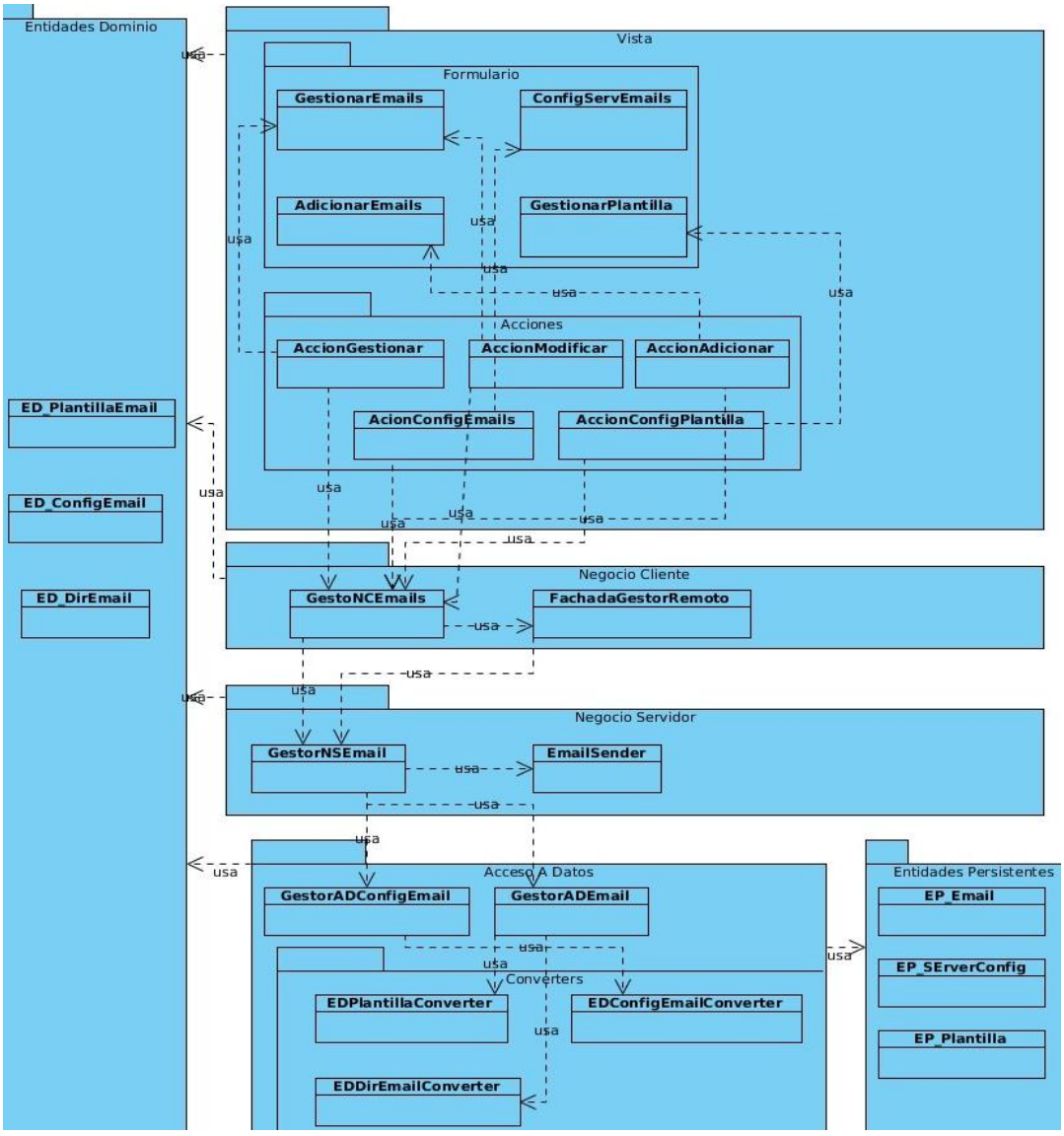


Figura 3. Diagrama de paquetes (Fuente: elaboración propia).

2.11 Tarjetas Clase, Responsabilidad y Colaborador (CRC)

Para el desarrollo de sistemas que utilizan como metodología de desarrollo XP no se requiere de la realización del diagrama de clases, debido a que la metodología contiene una variante denominada

tarjetas de Cargo o Clase, Responsabilidad y Colaboración (CRC). Estas tarjetas son una técnica simple e informal pero efectiva que ha sido propuesta tanto para el modelado conceptual como para el diseño detallado de sistemas Orientado a Objetos. Se analizan basándose en sus responsabilidades con respecto al sistema. Una tarjeta CRC establece 3 dimensiones las cuales identifican el rol de un objeto en análisis y/o diseño: nombre de la clase, responsabilidades y colaboraciones. Las mismas determinan el comportamiento de cada actividad. En la tabla 21 se muestra una tarjeta CRC, las demás se pueden ver en los Anexos (ver Anexo 2).

Tabla 20. Tarjeta CRC AccionGestionarServidor.

AccionGestionarServidor	
Responsabilidad	**Colaboraciones**
onBtnAdicionar	GestorMarcoTrabajo
onBtnSiguiente	GestorMensaje Gestorexcepcion

2.12 Estándar de codificación

Para un mejor entendimiento del código el componente se emplea el mismo estándar de codificación del marco de trabajo, *CamelCase* en su variante *lowerCamelCase y* UpperCamelCase*,* lo que permite un mayor entendimiento del código.

Convenciones de nomenclatura

General:

- Para las variables, clases o métodos que contengan tildes, serán utilizadas las propias palabras, pero sin la acentuación.
- Serán utilizados nombres relacionados con el contexto en que se desarrolla.

Identación:

Se utilizó UperCamelCase para:

- **Clases:** el nombre de las clases siempre empieza con letra inicial mayúscula. En caso de que sea una palabra compuesta, cada una de las palabras empezará con letra mayúscula (ver Figura 4).

```
public class AccionGestionarServidor implements AccionPanel {
```

Figura 4.Estilo UperCamelCase para las clases (Fuente: elaboración propia).

Se utilizó lowerCemelCase para:

- **Nombre de las variables:** no utilizar nombres ambiguos, siempre empezará con letra inicial minúscula, si es compuesto, cada una de las palabras siguientes es escrita con letra mayúscula (ver Figura 5).

```
private String titulo;
private final Map<String, Object> configutacionesBotones;
private PnlConfigurarServidor formulario;
private Callback acciones;
private GestorNCEmail gestor;
```

Figura 5. Estilo lowerCamelCase para las variables (Fuente: elaboración propia).

- **Nombre de los métodos:** el nombre de los métodos siempre empezará con letra inicial minúscula, si es compuesto, cada una de las palabras siguientes es escrita con letra mayúscula (ver Figura 6).

```
public void inicializarFormulario() throws Exception {
```

Figura 6. Estilo lowerCamelCase para los métodos (Fuente: elaboración propia).

El correcto uso de estas nomenclaturas es de gran importancia para el desarrollo del componente informático, estos sirven de guía para un mayor entendimiento en el código, facilitando una mejor comprensión a la hora de darle un posterior mantenimiento.

2.13 Conclusiones parciales

En este capítulo se describieron las principales características del sistema a través de los artefactos definidos por la metodología. Se definieron los requisitos funcionales y no funcionales, brindándole al desarrollador una visión más clara del componente que se desea desarrollar. Los requisitos funcionales fueron descritos y organizados a través de las historias de usuarios, las cuales fueron escritas por el cliente, en un lenguaje natural y sencillo, lo que permitirá llevar a cabo su rápida implementación, además de permitir estimar el esfuerzo de desarrollo de estas.

Capítulo 3: Validación de la solución propuesta

3.1 Introducción

En el presente capítulo se presentan las pruebas de software al componente así como los resultados obtenidos. Específicamente validación de las variables y requisitos, pruebas unitarias y de aceptación. Además de los métodos de las pruebas unitarias tales como: caja blanca y caja negra.

3.2 Validación de las variables de la investigación

En este epígrafe se validan las variables de la presente investigación (simplicidad y efectividad). A continuación en la Tabla 21 se muestran las dimensiones definidas para estas variables, así como los indicadores e índices para validar el cumplimiento de cada indicador.

Tabla 21. Operacionalización de variables.

Variable	Dimensión	Indicador	Índice
Simplicidad	Mantenibilidad	- Acoplamiento entre clases objeto (ACO)	< 14
Efectividad	Proceso de notificación.	- Capacidad de envío a varios usuarios.	Sí o no.
		- Capacidad de envío desde varios servidores.	Sí o no.
		- Capacidad de envío automático de notificaciones.	Sí o no.
		- Capacidad de envío mediante plantillas predefinidas	Sí o no.

Resultados obtenidos en la aplicación de la métrica ACO para medir la variable simplicidad

Se aplica la métrica ACO a la clase AccionGestionarServidor, representada por la tarjeta CRC AccionGestionarServidor (ver Tabla 20).

Según plantean García Sánchez, Ana María en su artículo *"Evaluación de métricas de calidad del software sobre un programa Java"* es favorable que el valor de ACO se encuentre por debajo de 14, para así asegurar la mantenibilidad y reusabilidad del diseño propuesto (García Sanchez, 2010). Teniendo en cuenta este criterio y que las múltiples colaboraciones sobre una misma clase son tratadas como un único acceso, se obtuvo un valor de ACO igual a 3, demostrando así que la variable simplicidad posee valores aceptables según las métricas aplicadas.

Resultados obtenidos para medir la variable efectividad

Tabla 22. Resultado de evaluar variable efectividad.

Indicador	Antes	Después
Capacidad de envío a varios usuarios.	No	Sí
Capacidad de envío desde varios servidores.	No	Si

Capacidad de envío automático de notificaciones.	No	Si
Capacidad de envío mediante plantillas predefinidas	No	Si

Teniendo en cuenta los resultados expuestos en la Tabla 22 se concluye que después de desarrollado el componente de notificaciones, cada uno de los indicadores definidos para la investigación se cumplen satisfactoriamente.

3.3 Validación de los requisitos

La aplicación de técnicas de validación está encaminada a demostrar que los requisitos previamente definidos cumplen con las expectativas del cliente. En este caso para el desarrollo del componente, se emplearon las siguientes:

Revisiones formales: se realizaron revisiones formales a cada uno de los requisitos por parte del cliente y del equipo de desarrollo, obteniéndose un total de 2 no conformidades (NC) de tipo técnica, 3 de redacción y 2 de ortografía, las que fueron corregidas en tiempo, generándose un Acta de Aceptación por parte del cliente (ver Anexo 3).

Generación de casos de prueba: como parte del proceso de validación de los requisitos funcionales del componente se diseñaron un conjunto de casos de pruebas, los cuales fueron creados a partir de cada historia de usuario definida. En los Anexos (ver Anexo 5) se muestran todos los casos de pruebas.

3.3.1 Métrica aplicada a los requisitos

Con el propósito de medir la calidad de la especificación de los requisitos de software se aplicó la métrica Calidad de la Especificación (CE).

Para obtener el nivel de entendimiento y precisión de los requisitos se debe calcular, en primer lugar, 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, capturados para el desarrollo del componente, se obtiene:

$$Nr = 13+3$$

$$Nr = 16$$

Finalmente, para calcular la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación:

$$ER = Nui/Nr$$

Nui: número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

Teniendo en cuenta que de los requisitos obtenidos para el desarrollo del componente solo uno causó contradicción en las interpretaciones y mientras más cercano a uno se encuentre el valor de ER, menor será la ambigüedad, a continuación se hace la sustitución de las variables:

ER= 15/16

ER=0,94

Como resultado final se obtiene 0,94 indicando que existe muy poca ambigüedad en los requisitos capturados ya que el 94% de ellos son entendibles.

3.4 Validación del diseño

A continuación se muestran los resultados de la aplicación de las métricas aplicadas para la validación del diseño.

3.4.1 Resultados de aplicar la métrica APH

La siguiente tabla muestra los resultados de calcular la métrica APH en las clases diseñadas para la solución del componente. El valor APH representa el número de ancestros que posee en la jerarquía de la clase en cuestión.

Tabla 23. Resultado de aplicar la métrica APH (Fuente: elaboración propia).

Clase	Valor de APH
AccionAdicionarPlantillaEmail	1
AccionGestionarPlantilla	1
AccionModificarPLantillaEmail	2
AccionAdicionarServidor	1
AccionGestionarServidor	1
AccionModificarServidor	2
GestorADConfiguracionEmail	1
GestorADEmail	1
GestorNSEmail	1
GestorNCEmail	1
GestorADConfiguracionEmailLocal	1
GestorADEmailLocal	1
GestorADPlantillaEmail	1
GestorADPlantillaEmailLocal	1

3.4.2 Resultados de aplicar la métrica RC

En la siguiente tabla se muestran como datos las 13 clases a las cuales se le aplicó la métrica RC en la columna **Clase**, la cantidad de relaciones que tiene una con respecto a las demás clases en la columna **Cantidad de relaciones de uso**, junto a los atributos de calidad: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas en sus columnas

correspondientes. Al promediar la cantidad de relaciones que tiene cada clase se obtuvo un resultado de dependencia por clase de 2,14.

Tabla 24. Resultados de aplicar la métrica RC (Fuente: elaboración propia).

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
AccionAdicionarPlantillaEmail	3	Alta	Media	Media	Media
AccionGestionarPlantilla	3	Alta	Media	Media	Media
AccionModificarPLantillaEmail	1	Bajo	Baja	Alta	Baja
AccionAdicionarServidor	2	Medio	Baja	Alta	Baja
AccionGestionarServidor	3	Alta	Media	Media	Media
AccionModificarServidor	1	Bajo	Baja	Alta	Baja
GestorADConfiguracionEmail	2	Medio	Baja	Alta	Baja
GestorADEmail	1	Bajo	Baja	Alta	Baja
GestorNSEmail	3	Alta	Media	Media	Media
GestorNCEmail	3	Alta	Media	Media	Media
GestorADConfiguracionEmail Local	2	Bajo	Baja	Alta	Baja
GestorADEmailLocal	2	Bajo	Baja	Alta	Baja
GestorADPlantillaEmail	2	Bajo	Baja	Alta	Baja
GestorADPlantillaEmailLocal	2	Bajo	Baja	Alta	Baja

La gráfica mostrada en la Figura 7 guarda relación con la Tabla 24 y representan el resultado de haber aplicado (RC).

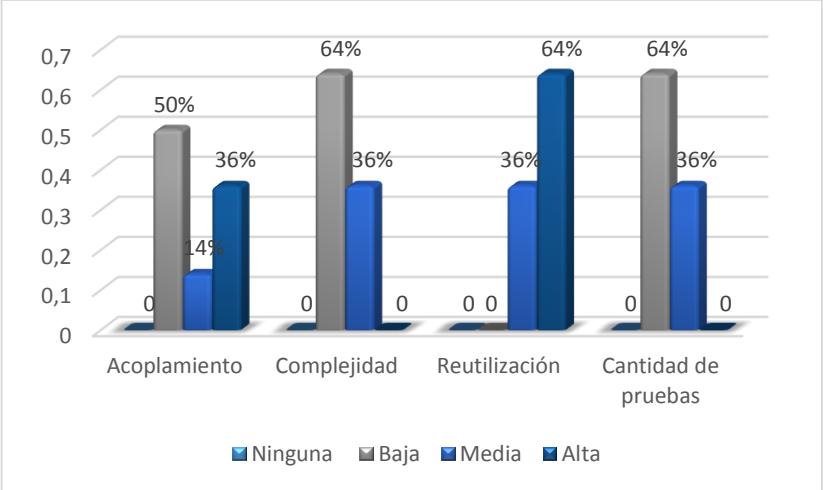


Figura 7. Resultado de la métrica RC (Fuente: elaboración propia).

Análisis de los resultados obtenidos al evaluar la métrica RC

De la aplicación de las métricas al conjunto de clases propuestas en el diseño, se obtuvieron como resultados que el 50% de las clases poseen un acoplamiento bajo, resultando en una solución que evidencia su calidad. La complejidad de mantenimiento se conserva baja para el 64% de las clases por tanto son fácilmente mantenibles y reutilizables para un 64% con valor alto de este atributo.

Además el diseño facilita la aplicación de pruebas reduciendo su número pues el 64% de las clases poseen un valor bajo.

3.4.3 Resultados de aplicar la métrica TOC

En la siguiente tabla se recoge como datos las 11 clases a las cuales se le aplicó la métrica TOC en la columna **Clase**, en la columna **Cantidad de operaciones** la cantidad de métodos que tiene la clase, y en las siguientes columnas los atributos de calidad que utiliza esta métrica y en ellas se representa la categorización en baja, media o alta, de lo que se obtuvieron los siguientes resultados:

Tabla 25. Resultados de aplicar la métrica TOC (Fuente: elaboración propia).

Clase	Cantidad de operaciones	Responsabilidad	Complejidad	Reutilización
AccionAdicionarPlantillaEmail	12	Media	Media	Media
AccionGestionarPlantilla	16	Media	Media	Media
AccionModificarPlantillaEmail	12	Media	Media	Media
AccionAdicionarServidor	12	Media	Media	Media
AccionGestionarServidor	16	Media	Media	Media
AccionModificarServidor	12	Media	Media	Media
GestorADConfiguracionEmail	4	Baja	Baja	Alta
GestorADEmail	4	Baja	Baja	Alta
GestorNSEmail	13	Media	Media	Media
GestorNCEmail	14	Media	Media	Media
GestorADConfiguracionEmailLocal	4	Baja	Baja	Alta
GestorADEmailLocal	4	Baja	Baja	Alta
GestorADPlantillaEmail	4	Baja	Baja	Alta
GestorADPlantillaEmailLocal	4	Baja	Baja	Alta

La gráfica mostrada en la Figura 8 guarda relación con la Tabla 25 y representan el resultado de haber aplicado (TOC).

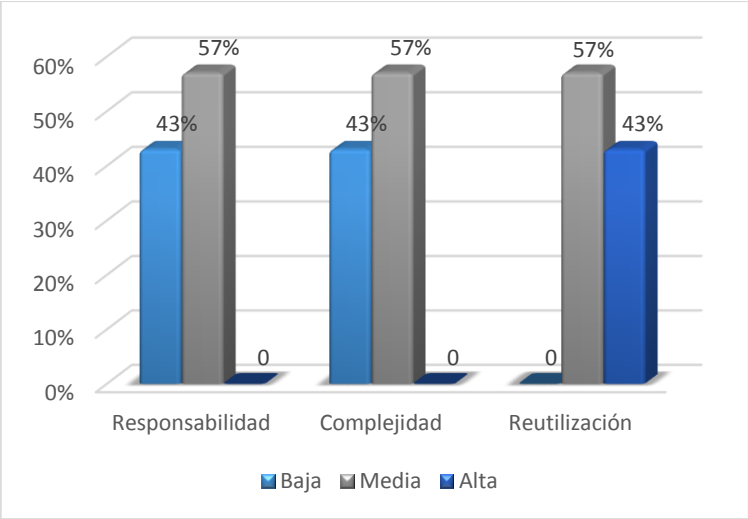


Figura 8. Resultado de la métrica TOC (Fuente: elaboración propia).

Análisis de los resultados obtenidos al evaluar la métrica TOC

Como resultados de la aplicación de la métrica se obtuvo que el 57% de las clases presentan valores en la media para todos los atributos evaluados (Responsabilidad, Complejidad y Reutilización). Como aspecto positivo no existen valores altos para la Responsabilidad y la Complejidad, así como valores bajos asociados a la Reutilización.

3.5 Pruebas

A continuación se muestra los resultados de las pruebas realizadas al componente de la presente investigación.

3.5.1 Pruebas unitarias

Estas fueron establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema.

3.5.1.1 Técnica de caja blanca

Para demostrarla aplicación de la técnica se tomó como ejemplo el método *onBtnTerminar*, perteneciente a la clase *AccionGestionarServidor*. A los efectos de la lógica del componente este método es el encargado de enviar lo datos a ser procesados por el servidor. Este método permite eliminar una configuración de servidor de envío con todas sus propiedades.

Para realizar las pruebas de caja blanca se utilizó la técnica de camino básico, para la cual primero se procede a enumerar las sentencias de código seleccionado (ver Figura 9) y a partir del mismo se construye el grafo de flujo asociado (ver Figura 10).

```
@Override
public void onBtnTerminar() {
    if (getFormulario().cmbConfiguracionServidor.getSelectedIndex() != 0) {//1
        try {
            int confirmar = GestorMensajes.MensajeConfirmar("¿Esta seguro que desea eliminar la configuracion seleccionada?");//2
            if (confirmar == AdministradorConstante.RET_OK) {//3
                EDConfiguracionEmail itemConfiguracionEmail = (EDConfiguracionEmail) getFormulario().cmbConfiguracionServidor.getSelectedItem();
                gestor.eliminarConfiguracionEmail(itemConfiguracionEmail);
                ((DefaultComboBoxModel) getFormulario().cmbConfiguracionServidor.getModel()).removeElement(itemConfiguracionEmail);
                getFormulario().cmbConfiguracionServidor.setSelectedIndex(0);
                GestorMensajes.MensajeAviso("La configuracion ha sido eliminada correctamente");//4
            }
        } catch (Exception ex) {//5
            GestorExcepcion.reportarExcepcion(ex);//6
        }
    } else {
        GestorMensajes.MensajeAviso("Debe seleccionar una configuracion para eliminarla");//7
    }
}//8
```

Figura 9. Código fuente de la funcionalidad numerada (Fuente: elaboración propia).

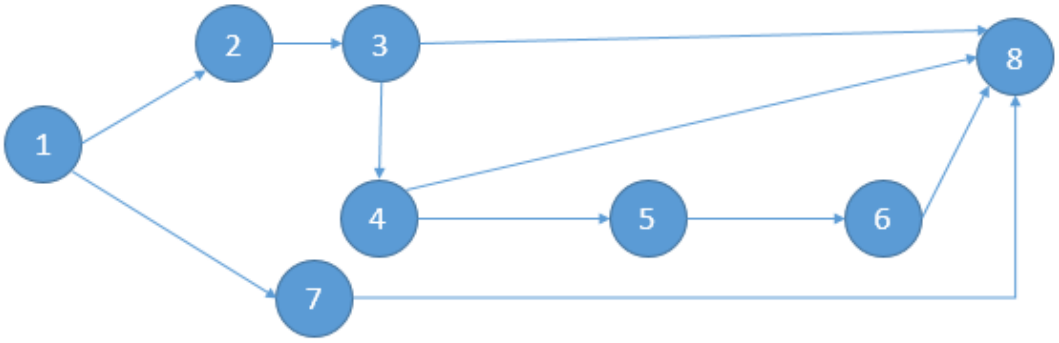


Figura 10. Grafo de flujo asociado al código fuente (Fuente: elaboración propia).

Cálculo de la complejidad ciclomática: proporciona una medición cuantitativa de la complejidad lógica del programa. El valor que se calcula define el número de caminos independientes del conjunto básico del programa.

V (G)= A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

V (G)= A-N+2 =10-8+2=4

Determinar un conjunto básico de caminos linealmente independientes: el valor de V (G) da el número de caminos linealmente independientes de la estructura de control, a continuación se definen los siguientes caminos:

Camino básico # 1:1-2-3-8

Camino básico # 2:1-2-3-4-8

Camino básico # 3: 1-2-3-4-5-6-8

Camino básico # 4: 1-7-8

Determinar un conjunto básico de caminos linealmente independientes: para obtener los casos de prueba se crea un conjunto de datos que permitan recorrer los caminos identificados. En este caso se obtuvieron 4 casos de pruebas de los cuales se muestra el caso de prueba asociado a camino básico #1 (ver Tabla 26).

Tabla 26. Caso de prueba de caja blanca para el camino básico #1

Caso de pruebas: Camino básico # 1	
Entrada	Configuración del servidor seleccionada.
Resultados esperados	Ninguno.
Condiciones	Haber registrado al menos una configuración.

Una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico para todos los métodos del componente, se concluye que el código generado no presenta ciclos infinitos y no existe código innecesario.

3.5.1.2 Técnica de caja negra

Las pruebas de caja negra son pruebas funcionales que se realizan al software. Se elaboraron un total de 14 casos de pruebas. Para ver los demás casos de pruebas ver Anexo 5.

Tabla 27. Caso de prueba de Caja negra de la HU Crear plantillas de correos a partir de entidades de dominio.

Escenario	Descripción	Plantilla	Cuerpo de la plantilla	Respuesta del sistema	Flujo central
EC 1.1: Redactar plantilla.	Se adiciona una nueva plantilla correctamente.	V Nuevo	V N/A	Se muestra un mensaje confirmando que se adicionó satisfactoriamente la plantilla.	1. Seleccionar la opción del “Gestionar plantilla”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
EC 1.2: Redactar Plantilla incorrectamente.	Se muestra un mensaje indicando que hay datos escritos incorrectamente.	I nuev@#go	V N/A	Se muestra un mensaje con los campos que contienen errores.	1. Seleccionar la opción “Gestionar plantilla”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
EC 1.3: Redactar plantilla de forma incompleta.	Se muestra un mensaje indicando que no se han introducido los datos.	I (vacío)	V N/A	Se muestra un mensaje con los campos obligatorios que faltan por introducir.	1. Seleccionar la opción “Gestionar plantilla”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
		V Nuevo	I N/A		

Con el objetivo de comprobar las funcionalidades del sistema se implementaron correctamente y responden a las necesidades del cliente aplicando los casos de pruebas antes descritos. Se realizaron pruebas funcionales por parte de los especialistas del grupo de Calidad del Centro de Gobierno Electrónico.

Las pruebas se realizaron en tres iteraciones (ver Anexo 7). En la primera se detectaron un total de 12 (NC), clasificadas en 7 de ortografía, 2 de funcionalidad, 2 de validación y 1 de interfaz, al finalizar la iteración todas las NC quedaron resueltas. Para una segunda iteración fueron resueltas todas las

NC detectadas en la primera iteración y se identificaron 5 NC, clasificadas en 4 de ortografía y 1 de funcionamiento. En la tercera iteración los resultados fueron satisfactorios, obteniéndose cero NC, generándose así el Acta de Liberación Interna de Productos de Software (ver Anexo 8). La Figura 11 ilustra los resultados de aplicar el método de caja negra, teniendo en cuenta los tipos de NC identificadas (ortografía, validación, funcionalidad, interfaz).

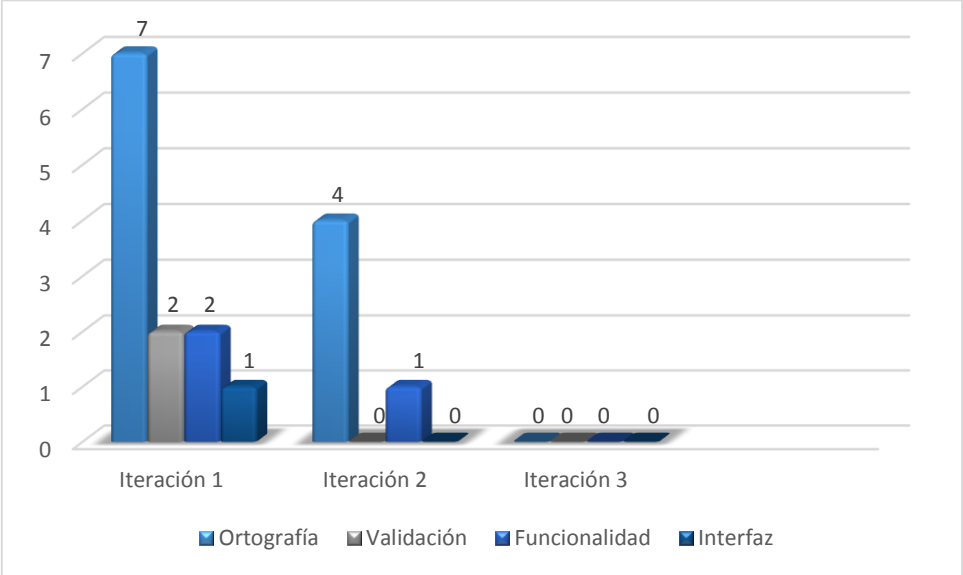


Figura 11. No conformidades detectadas al aplicar el método de caja negra (Fuente: elaboración propia).

3.5.2 Pruebas de aceptación

Estas pruebas en la metodología XP son especificadas por el cliente, los cuales son los responsables de verificar que los resultados obtenidos sean correctos. Son realizadas por el cliente y los usuarios finales de la aplicación. En esta serán probadas las funcionalidades exigidas por el cliente. Una vez superada esta prueba se considera que la aplicación esta apta para su uso y despliegue.

3.5.2.1 Casos de prueba de aceptación

La evaluación del caso de prueba de aceptación es clasificada en satisfactorio o insatisfactorio, para que una prueba sea satisfactoria tiene que cumplir con todas las condiciones que define el caso de prueba, que se cumpla con todo lo descrito en la prueba, que se verifiquen las condiciones de ejecución y los pasos de la misma, además que se obtengan los resultados esperados. En caso de que no se cumpla el caso de prueba sería evaluado de insatisfactorio. A continuación se muestra un caso de prueba de aceptación aplicado al componente.

Tabla 28. Caso de prueba de Aceptación de la HU Adicionar configuración de servidor.

Casos de prueba de Aceptación

<table>
<tr><td colspan="2">Código de caso de prueba: 01</td><td>Nombre de la HU: Adicionar configuración de servidor.</td></tr>
<tr><td colspan="3">Nombre de la persona que realiza la prueba: Esther García Naranjo.</td></tr>
<tr><td colspan="3">Descripción de la prueba: Comprobar que la configuración del servidor se guarda correctamente y se lista entre las opciones de gestionar servidor.</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Debe estar logueado con un usuario que tenga permisos de ejecutar esa acción.</td></tr>
<tr><td colspan="2">Entrada/Pasos de ejecución: N/A</td><td rowspan="2">Resultados esperados: Si se adiciona correctamente el componente lanza un mensaje diciendo que se ha insertado satisfactoriamente la configuración.</td></tr>
<tr><td>Acción: Se pulsa el menú Configuración servidor de envía y luego al botón adicionar.</td><td>Entrada: Texto</td></tr>
<tr><td colspan="3">Evaluación de la prueba: Satisfactoria.</td></tr>
</table>

3.6 Conclusiones parciales

Las técnicas de validación de requisitos y métricas de diseño utilizadas permitieron detectar la mayor cantidad de errores que presenta el componente, para darle así solución a los problemas detectados. Las pruebas de forma general arrojaron resultados satisfactorios, siendo corregidos en tiempo todos los errores.

Conclusiones generales

- Luego del estudio realizado sobre algunos sistemas que realizan el envío de notificaciones se identificaron algunas debilidades y posibles puntos a reutilizar, arrojando como resultado que los sistemas estudiados no solucionan la situación problemática expuesta al inicio de la investigación, por lo que conlleva a implementar un componente para el envío de notificaciones a través del correo electrónico.
- La implementación del componente de notificaciones se sustentó en la arquitectura basada en capas, donde se identificaron 4 capas fundamentales. El componente presenta una arquitectura cliente servidor y se determinaron los patrones de diseño debidamente fundamentados; los cuales contribuyeron a que se obtuvieron resultados favorables en los atributos de responsabilidad, complejidad algorítmica y reutilización de componentes.
- Se implementó el componente para el marco de trabajo XEGFORT, soportando el envío de notificaciones vía correo electrónico, facilitando además el manejo de las mismas.
- Las técnicas de validación de los requisitos, métricas de validación del diseño y las pruebas unitarias demostraron el cumplimiento de diferentes atributos de calidad, lo que demuestra que el componente cumple satisfactoriamente con los requisitos definidos por el cliente.

Bibliografía referenciada

Barroso Pérez, Javier. 2011. *Sistema de notificación de eventos.* La Habana : s.n., 2011.

Carabali, Mauricio. 2013. Prezi. [En línea] 26 de Septiembre de 2013. [Citado el: 20 de Mayo de 2016.] https://prezi.com/sjwfwmix7slk/pruebas-de-caja-negra-y-caja-blanca/ .

Correa Selamé, Danilo. 2006. *Nuevo procedimiento laboral.* 2006 : PuntoLex, 2006.

Cruz Figueredo, Denis Henry. 2013. *Módulo de notificaciones del Sistema Integral de Análisis de Información.* La Habana : s.n., 2013.

Dayne, anthony y Philips, Nicola. 2012. *Desarrollo.* s.l. : Alianza, 2012.

Díez de los Ríos, Almudena. 2012. Glosario explicativo. [En línea] 1 de Septiembre de 2012. [Citado el: 16 de Diciembre de 2015.] http://pendientedemigracion.ucm.es/info/contratos/wikiglo/index.php/Categor%C3%ADa:Notificaci%C3%B3n_electr%C3%B3nica.

Edo, Mauri. 2011. Testing funcional. [En línea] 12 de Marzo de 2011. [Citado el: 20 de Mayo de 2016.] https://testingfuncional.wordpress.com/2011/03/12/%C2%BFtesting-funcional-o-pruebas-de-caja-negra/.

Española, Real Academia. 2015. Diccionario de la lengua española. [En línea] 2015. [Citado el: 15 de diciembre de 2015.] http://dle.rae.es/?id¡Qeb4gld.

Gálvez Rojas, Sergio y García Sucino, Ignacio. 2006. *Java a tope: JavaMail (JavaMail con ejemplos).* Malaga : s.n., 2006.

García Sanchez, Ana María. 2010. *Evaluación de métricas de calidad del software sobre un programa Java.* Madrid : s.n., 2010.

García Sánchez, Ana María. 2010. *Evaluación de métricas de calidad del software sobre un programa Java.* Madrid : Universidad Complutense de Madrid, 2010.

Google. 2015. Gmail Notifier. [En línea] 2015. [Citado el: 20 de Diciembre de 2015.] http://support.google.com.

Group, The PostgreSQL Global Development. 2015. PostgreSQL. [En línea] 2015. [Citado el: 8 de Diciembre de 2015.] http://postgresql.org.es..

Groussard, Thierry. 2012. *Java 7: Los fundamentos del lenguaje Java. .* s.l. : Ediciones ENI, 2012.

Hinojosa Tinoco, Diego Andésñ. 2013. [En línea] 2013. [Citado el: 15 de Enero de 2016.] http://repositorio.puce.edu.cu/bitstream/handle/22000/6379/9.21.001774.pdf?sequence¡04.

IEEE. 2016. IEEE Standar Glossary of Software Engineering Terminology. [En línea] 2016. [Citado el: 20 de Mayo de 2016.]

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=159342&filter=AND%28p_Publication_Number:2238%29.

Lamarca Lapuente, María Jesús. 2013. *Hipertexto: El nuevo concepto de documento en la cultura y la imagen.* Madrid : s.n., 2013.

López, Gallardo. 2011. *Tema 1: Introducción a los lenguajes de programación.* 2011.

Matos Rodríguez, Lenis y Sarmiento Sastrieques, Daniel. 2011. *Propuesta de mecanismo de seguridad para el marco de trabajo Kairos.* La Habana : s.n., 2011.

Miranda, Cano, Jhans. 2013. [En línea] 2013. [Citado el: 17 de Diciembre de 2015.] http://herramientascase1141.blogspot.com/2013/05/herramientas-case.html.

Montenegro, MSc. Manuel. 2012. *Interfaces gráficas con swing.* 2012.

Naboulsi, Zain. 2011. Microsoft Developer. [En línea] 19 de Mayo de 2011. [Citado el: 15 de Mayo de 2016.] https://blogs.msdn.microsoft.com/zainnab/2011/05/19/code-metrics-depth-of-inheritance-dit/.

Navarro Cadavid, Andrés, Fernández Martínez, Juan y Morales Vélez, Jonathan. 2013. *A review of agile methodologies for software development.* 2013.

Netbeans, Comunidad. 2015. NetsBeans. [En línea] 2015. [Citado el: 7 de Diciembre de 2015.] https://netbeans.org/features/index.html.

Núñez Ponce, Julio. 2016. *Implicaciones jurídicas de la Notificación Enviada por Medios Informáticos y el Domicilio virtual.* s.l. : Revista Electrónica de Derecho Informático, 2016.

Océano, Grupo. 2015. *Diccionario Ilustrado Océano de la Lengua Española.* Barcelona, España : Océano, 2015.

Oracle Corporation. 2011. *GlassFish Server Open Source Edition 3.1 Add-On Component Development Guide.* Redwood City : 500 Oracle Parkway, 2011.

Oracle. 2014. *Java Platform, enterprise Edition.* 2014.

Paradigm, Visual. 2015. Software Design tools for Agile Teams, with UML, BPMN and more. [En línea] 2015. [Citado el: 17 de Diciembre de 2015.] https://www.visual-paradigm.com/.

Pérez, Juan y Millalba, Maríañ. 2005. *Diccionario Ócéano de Sinónimos y Antónimos.* Barcelona, España : Océano, 2005.

Pressman, Roger S. 2014. *Ingeniería de Software. Un enfoque práctico.* México : s.n., 2014.

Ramírez Giraldo, Mauricio. 2013. *Desarrollo de software para la gestión y control eficiente de la cartera de créditos educativos de la Universidad libre de Pereira.* Universidad libre de Pereira : s.n., 2013.

Samuel. 2013. Prezi. [En línea] 19 de Enero de 2013. [Citado el: 20 de Diciembre de 2015.] https://prezi.com/zpwy1wkclv_4/metodologias-de-ingenieria-de-software/.

Somerville, Ian. 2015. *Software Engineering, 10th Edition.* University of St Andrews, Scotland : PEARSON EDUCACIÓN, 2015.

Zapata, Javier. 2013. Pruebas de software. [En línea] 21 de 1 de 2013. [Citado el: 15 de Mayo de 2016.] https://pruebasdelsoftware.wordpress.com/.

Zara. 2012. *Apache´s Maven Tutorial.* 2012.

Zuckerberg, Mark. 2016. Facebook. [En línea] 2016. [Citado el: 15 de Diciembre de 2015.] http://facebook.com.

Anexos

Anexo 1. Encuesta.

Encuesta realizada con el objetivo de obtener un criterio válido que demuestre la necesidad de implementar un componente de notificación:

1. Al enviar un correo a varios destinatarios el proceso es:
   ___ fácil ___ normal ___ difícil
2. Implementar un componente de correo orientado a un dominio atendiendo a las características de XEGFORT es:
   ___ fácil ___ normal ___ difícil
3. Cuantas veces desarrollando XEGFORT ha tenido que implementar un componente de correo: _____
4. Cuando ha tenido que implementar unte de correo ha tenido que empezar de cero:
   ___ Sí ___ No

Anexo 2. Tarjetas CRC.

Tabla 29. CRC AccionGestionarServidor

AccionGestionarServidor	
Responsabilidad	**Colaboraciones**
onBtnAdicionar	GestorMarcoTrabajo
onBtnSiguiente	GestorMensajes Gestorexcepcion

Tabla 30. CRC AccionAdicionarServidor

AccionAdicionarServidor	
Responsabilidad	**Colaboraciones**
onBtnAceptar	Gestormensajes GestorExcepcion
inicializarFormulario	GestorMarcoTrabajo

Tabla 31. CRC AccionModificarServidor

AccionModificarServidor	
Responsabilidad	Colaboraciones
onBtnAceptar	GestorMensajes GestorExcepcion

Tabla 32. CRC AccionGestionarPlantilla

AccionGestionarPlantilla	
Responsabilidad	**Colaboraciones**
onBtnAnterior	GestorMarcoTrabajo GestorExcepcion
onBtnSiguiente	GestorMensajes

Tabla 33. AccionModificarPlantillaEmail

AccionModificarPlantillaEmail	
Responsabilidad	**Colaboraciones**
onBtnAceptar	GestorMensajes GestorExcepcion

Tabla 34. AccionAdicionarPlantillaEmail

AccionAdicionarPlantillaEmail	
Responsabilidad	**Colaboraciones**
onBtnAceptar	GestorMensajes GestorExcepcion
inicializarFormulario	GestorMarcoTrabajo

Anexo 3. Acta de aceptación de los requisitos.

Universidad de las Ciencias Informáticas

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento del desarrollo de la Tesis: **Componente para la gestión de notificaciones para el marco de trabajo XEGFORT.** Se hace entrega de los productos que se relacionan a continuación:

- Especificación de Requisitos de Software
- Historias de Usuarios

La Parte Cliente, luego de haber revisado los productos de trabajo determina que los mismos cumplen con los estándares establecidos para el diseño y redacción de estos artefactos, quedando aceptados de esta forma.

Entrega	**Recibe**
Nombre y apellidos: Esther García Naranjo	**Nombre y apellidos:** Ing. Reinier Silverio Figueroa
Cargo: Estudiante	**Cargo:** Jefe de proyecto XEGFORT

Fecha: 6/05/2016

Anexo 4. Acta de aceptación del componente.

Universidad de las Ciencias Informáticas

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento del desarrollo de la Tesis: **Componente para la gestión de notificaciones para el marco de trabajo XEGFORT.** Se hace entrega de los productos que se relacionan a continuación:

- Componente para la gestión de notificaciones para el marco de trabajo XEGFORT.

La Parte Cliente, luego de haber revisado el componente, considera que la solución cumple con cada uno de los requisitos que originaron su desarrollo y constituye un aporte significativo al macro de trabajo XEGFORT.

Entrega	**Recibe**
Nombre y apellidos: Esther García Naranjo	**Nombre y apellidos:** Ing. Reinier Silverio Figueroa
Cargo: Estudiante	**Cargo:** Jefe de proyecto XEGFORT

Fecha: 28/06/2016

Anexo 5. Diseño de casos de prueba.

Tabla 35. Caso de prueba HU Adicionar cuenta de correo electrónico.

Escenario	Descripción	Primer nombre	Segundo nombre	Primer apellido	Segundo apellido	Documento de entidad	Sexo	Nacionalidad	Correo electrónico	Respuesta del sistema	Flujo central
EC 1.1: Adicionar cuentas de correo electrónico	Se adiciona una nueva cuenta de correo electrónico correctamente	V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@dominio	Se muestra un mensaje confirmando que se adicionó satisfactoriamente la persona.	1. Seleccionar la opción “Cuentas de usuario”. 2. Seleccionar el botón “Registrar cuenta”. 3. Click en el botón “Adicionar” 4. Introducir los datos solicitados. 5. Seleccionar la opción “Aceptar”.
EC 1.2: Adicionar cuentas de correo electrónico incorrectamente	Se muestra un mensaje indicando que hay datos escritos incorrectamente	I nuev@3#o	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@dominio	Se muestra un mensaje con los campos que contienen errores.	1. Seleccionar la opción “Cuentas de usuario”. 2. Seleccionar el botón “Registrar cuenta”. 3. Click en el botón “Adicionar” 4. Introducir los datos solicitados. 5. Seleccionar la opción “Aceptar”.
		V Nuevo	I nueidpdv@3#o	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@dominio		
		V Nuevo	V Nuevo	I nuev@3#o	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@dominio		
		V Nuevo	V Nuevo	V Nuevo	I nuev@3#o	V 89021145781	V F/M	V cubana	V usuario@dominio		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	I wfwf$%%fdgdf	V F/M	V cubana	V usuario@dominio		

		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	I N/A	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	I N/A	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	I adf·$c(/·%”		
EC 1.3: Adicionar cuentas de correo electrónico de forma incompleta	Se muestra un mensaje indicando que no se han introducido los datos.	I (vacío)	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@domino	Se muestra un mensaje con los campos obligatorios que faltan por introducir.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
		V Nuevo	V Nuevo	I (vacío)	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	I (vacío)	V F/M	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	I (vacío)	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	I (vacío)	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	I (vacío)		

Tabla 36. Caso de prueba HU Eliminar cuenta de correo electrónico.

Escenario	Descripción	Correo electrónico	Respuesta del sistema	Flujo central
EC 1.1: Eliminar cuenta de correo electrónico	Se elimina correctamente una cuenta de correo electrónico	NA	Retorna a la página que le dio origen a la acción y se muestra un mensaje de que se ha eliminado la cuenta de correo electrónico correctamente.	1. Seleccionar la opción “Cuentas de usuario”. 2. Seleccionar el botón “Registrar cuenta”. 3. Introducir criterio de búsqueda y click al botón “Buscar”. 4. Seleccionar la persona y click en el botón “Siguiente”. 5. Actualizar el campo “Estado de la cuenta” a “Desactivado”.

Tabla 37. Caso de prueba HU Modificar cuenta de correo electrónico.

Escenario	Descripción	Primer nombre	Segundo nombre	Primer apellido	Segundo apellido	Documento de entidad	Sexo	Nacionalidad	Correo electrónico	Respuesta del sistema	Flujo central
EC 1.1: Modificar cuentas de correo electrónico	Se adiciona una nueva cuenta de correo electrónico correctamente	V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@dominio	Se muestra un mensaje confirmando que se modificó satisfactoriamente la persona.	1. Seleccionar la opción “Cuentas de usuario”. 2. Seleccionar el botón “Registrar cuenta”. 3. Click en el botón “Adicionar” 4. Introducir los datos solicitados. 5. Seleccionar la opción “Aceptar”.
EC 1.2: Modificar cuentas de correo electrónico incorrectamente	Se muestra un mensaje indicando que hay datos escritos incorrectamente	I nuev@3#o	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@dominio	Se muestra un mensaje con los campos que contienen errores.	1. Seleccionar la opción “Cuentas de usuario”. 2. Seleccionar el botón “Registrar cuenta”.
		V Nuevo	I nueidpdv@3#o	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@dominio		

		V Nuevo	V Nuevo	I nuev@3#o	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@domino		3. Click en el botón “Adicionar” 4. Introducir los datos solicitados. 5. Seleccionar la opción “Aceptar”.
		V Nuevo	V Nuevo	V Nuevo	I nuev@3#o	V 89021145781	V F/M	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	I wfwf$%%fdgdf	V F/M	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	I N/A	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	I N/A	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	I adf·$c(/·%”		
EC 1.3: Modificar cuentas de correo electrónico de forma incompleta	Se muestra un mensaje indicando que no se han introducido los datos.	I (vacío)	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@domino	Se muestra un mensaje con los campos obligatorios que faltan por introducir.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
		V Nuevo	V Nuevo	I (vacío)	V Nuevo	V 89021145781	V F/M	V cubana	V usuario@domino		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	I (vacío)	V F/M	V cubana	V usuario@domino		

		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	I (vacío)	V cubana	V usuario@dominio		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	I (vacío)	V usuario@dominio		
		V Nuevo	V Nuevo	V Nuevo	V Nuevo	V 89021145781	V F/M	V cubana	I (vacío)		

Tabla 38. Caso de prueba HU Listar cuentas de correo electrónico.

Escenario	Descripción	Listado	Respuesta del sistema	Flujo central
EC 1.1 Listar : Listar cuentas de correo electrónico	Se muestran en el sistema los datos correctos de todas las cuentas de correo electrónico	V 1	Se listan todas las cuentas de correo electrónico	1. Seleccionar la opción “Cuentas de usuario”. 2. Seleccionar el botón “Registrar cuenta”. 3. Introducir criterio de búsqueda y click al botón “Buscar”.

Tabla 39. Caso de prueba HU Mostrar cuenta de correo electrónico.

Escenario	Descripción	Mostrar cuentas	Respuesta del sistema	Flujo central
EC 1.1 Listar : Mostrar cuenta de correo electrónico	Se muestran en el sistema los datos de la persona incluyendo la cuenta de correo electrónico	V 1	Se muestran los datos de la persona.	1. Seleccionar la opción “Cuentas de usuario”. 2. Seleccionar el botón “Registrar cuenta”. 3. Introducir criterio de búsqueda y click al botón “Buscar”. 3. Seleccionar un elemento y click en el botón “Siguiente”.

Tabla 40. Caso de prueba HU Adicionar configuración de servidor.

Escenario	Descripción	Nombre	mail.host	mail.port	mail.user	mail.smtp.starttls.enable	Respuesta del sistema	Flujo central
EC 1.1: Adicionar configuración de servidor correctamente.	Se adiciona una nueva configuración de servidor correctamente.	V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	V true/false	Se muestra un mensaje de que se ha adicionado la configuración correctamente y retorna a la página que le dio origen a la acción.	1. Seleccionar la opción “Configurar servidor de envío”. 2. Click en el botón “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”.
EC 1.2: Adicionar configuración de servidor incorrectamente.	Se muestra un mensaje indicando que hay datos escritos incorrectamente.	I un$e%vo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	V true/false	Se muestra un mensaje con los campos que contienen errores.	1. Seleccionar la opción “Configurar servidor de envío”. 2. Click en el botón “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”.
		V Nuevo	I yu&hn%n	V 25	V egnaranjo	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	I mljk	V egnaranjo	V true/false		

		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	I nk”l&(ñ'¡	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	I sta$rtl%e&s		
EC 1.3: Adicionar configuración de servidor de forma incompleta	Se muestra un mensaje indicando que no se han introducido los datos obligatorios.	I (vacío)	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	V true/false	Se muestra un mensaje con los campos obligatorios que faltan por introducir.	1. Seleccionar la opción “Configurar servidor de envío”. 2. Click en el botón “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”.
		V Nuevo	I (vacío)	V 25	V egnaranjo	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	I (vacío)	V egnaranjo	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	I (vacío)	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	I (vacío)		

Tabla 41. Caso de prueba HU Eliminar configuración de servidor.

Escenario	Descripción	Servidor	Respuesta del sistema	Flujo central

EC 1.1: Eliminar configuración de servidor	Se elimina correctamente una configuración de servidor.	V (servidor seleccionado)	Retorna a la página que le dio origen a la acción y se muestra un mensaje de que se ha eliminado la configuración satisfactoriamente.	1. Seleccionar la opción “Configurar servidor de envío”.
EC 1.2: Eliminar configuración de servidor.	Cancela la acción de eliminar configuración de servidor.	I (vacío)	Muestra un mensaje que debe seleccionar una configuración para eliminar.	1. Seleccionar la opción “Configurar servidor de envío”.

Tabla 42. Caso de prueba HU Modificar configuración de servidor.

Escenario	Descripción	Nombre	mail.host	mail.port	mail.user	mail.smtp.starttls.enable	Respuesta del sistema	Flujo central
EC 1.1: Adicionar configuración de servidor correctamente.	Se adiciona una nueva configuración de servidor correctamente.	V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	V true/false	Se muestra un mensaje de que se ha modificado la configuración correctamente y retorna a la página que le dio origen a la acción.	1. Seleccionar la opción “Configurar servidor de envío”. 2. Selecciona la configuración a modificar y click en el botón “Modificar” 3. Introducir los datos solicitados. 4. Seleccionar la opción “Aceptar”.
EC 1.2: Adicionar configuración de servidor incorrectamente.	Se muestra un mensaje indicando que hay datos escritos incorrectamente.	I un$e%vo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	V true/false	Se muestra un mensaje con los campos que contienen errores.	1. Seleccionar la opción “Configurar servidor de envío”. 2. Selecciona la configuración a

		V Nuevo	I yu&hn%n	V 25	V egnaranjo	V true/false		modificar y click en el botón “Modificar” 3. Introducir los datos solicitados. 4. Seleccionar la opción “Aceptar”.
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	I mljk	V egnaranjo	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	I nk”l&(ñ'¡	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	I sta$rtl%e&s		
EC 1.3: Adicionar configuración de servidor de forma incompleta	Se muestra un mensaje indicando que no se han introducido los datos obligatorios.	I (vacío)	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	V true/false	Se muestra un mensaje con los campos obligatorios que faltan por introducir.	1. Seleccionar la opción “Configurar servidor de envío”. 2. Selecciona la configuración a modificar y click en el botón “Modificar” 3. Introducir los datos solicitados. 4. Seleccionar la opción “Aceptar”.
		V Nuevo	I (vacío)	V 25	V egnaranjo	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	I (vacío)	V egnaranjo	V true/false		

		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	I (vacío)	V true/false		
		V Nuevo	V 10.3.2.1/ smtp.uci.cu	V 25	V egnaranjo	I (vacío)		

Tabla 43. Caso de prueba HU Reportar estado de envío de correo electrónico por rango de fecha.

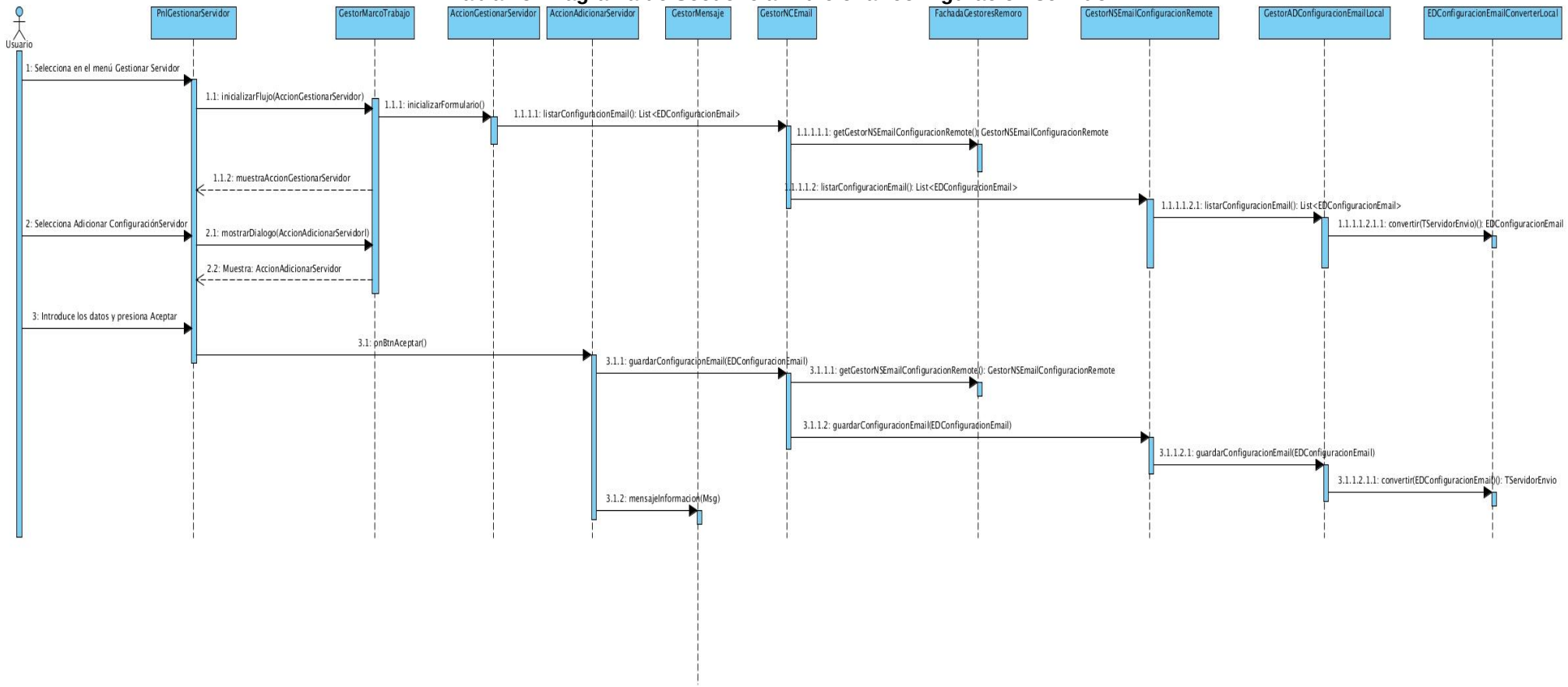
Escenario	Descripción	Reporte rango fecha	Respuesta del sistema	Flujo central
EC 1.1 Listar : Listar reporte de envío de correo electrónico	Se muestran en el sistema un listado de los reportes de envío de correos electrónicos por rango de fecha.	V 1	Se listan todos la información de los envíos de correos electrónicos	1. Seleccionar del menú la opción “Buscar envios”. 2. Introducir parámetro de búsqueda y dar click en el botón “Buscar”.

Tabla 44. Caso de prueba HU Reportar estado de envío de correo electrónico por plantilla.

Escenario	Descripción	Reporte rango fecha	Respuesta del sistema	Flujo central
EC 1.1 Listar : Listar reporte de envío de correo electrónico	Se muestran en el sistema un listado de los reportes de envío de correos electrónicos por plantilla.	V 1	Se listan todos la información de los envíos de correos electrónicos	1. Seleccionar del menú la opción “Buscar envios”. 2. Introducir parámetro de búsqueda y dar click en el botón “Buscar”.

Anexo 6. Diagrama de secuencia: Adicionar configuración de servidor.

Tabla 45. Diagrama de Secuencia: Adicionar configuración servidor.



Anexo 7. Lista de no conformidades.

Tabla 46. Lista de no conformidades.

Elemento	No.	No Conformidad	Ubicación de la No Conformidad	Etapa de Detección	Significativa	Estado NC	Clasificación de las NC	Responsable de la NC
App	1.	Al modificar los datos no se muestra un mensaje de confirmación. El mensaje que se muestra no cumple ninguna función.	App/Administracion Local/Cuentasdeusuario/registrarCuenta/Modificar	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Funcionalidad	Esther García Naranjo.
App	2.	La ventana Modificar no actualiza los datos en el sistema.	App/Administracion Local/Cuentasdeusuario/registrarCuenta/Modificar	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Funcionalidad	Esther García Naranjo.
App	3.	El campo E-mail admite caracteres especiales.	App/Administracion Local/CuentasdeUsuario/registrarCuenta/Adicionar/	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Validación	Esther García Naranjo.
App	4.	En el enunciado aparece Configuración de servidor deberá ser Configuración de servidor	App/AdministracionLocal/Configurarservidordeenvio	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	5.	En el mensaje de aviso dice que el campo Nombre puede tener 4 de 250 y en realidad es de 5 a 250. Además de debe ser 4 y 250 o de 4 hasta 250.	App/AdministracionLocal/Configurarservidordeenvio/Adicionar	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	6.	Al eliminar una configuración de servidor en el mensaje de aviso falta el punto final.	App/AdministracionLocal/Configurarservidordeenvio/Adicionar	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	7.	Al adicionar un servidor los campos en la tabla aparecen cambiados a como aparecen en la interfaz.	App/AdministracionLocal/Configurarservidordeenvio/Adicionar	1era Iteración	X	Abierta 30/06/2016 Cerrada	Interfaz	Esther García Naranjo.

Elemento	No.	No Conformidad	Ubicación de la No Conformidad	Etapa de Detección	Significativa	Estado NC	Clasificación de las NC	Responsable de la NC
						30/06/16		
App	8.	La palabra envío aparece sin tilde.	App/AdministracionLocal/Configurarservidordeenvio/Adicionar	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	9.	En el vínculo aparece e-mail y en otra interfaz aparece E-mail. Estandarizar en todo el componente.	App/Administracion Local/CuentasdeUsuario/registrarCuenta/Adicionar/	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	10	Validar las fechas y mostrar un mensaje de aviso.	App/AdministracionLocal/Probarnotificarporcorreo	1era Iteración	X	Abierta 30/06/2016 Cerrada 30/06/16	Validación	Esther García Naranjo.
App	11	Al introducir el nombre mal en el mensaje de aviso aparece la palabra configuración sin tilde, además de que no se refiere a eso si no a la plantilla.	App/AdministracionLocal/gestionarplantilla	1era Iteración	x	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	12	El aviso advierte que no debe contener caracteres especiales y comenzar con letra mayúscula, peo no dice que no puede contener números.	App/AdministracionLocal/gestionarplantilla	1era Iteración	x	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	13	En el mensaje modificar al no seleccionar ninguna persona el mismo no tiene punto final.	App/AdministracionLocal/gestionarplantilla/Modificar	2da Iteración	x	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	14	Eliminar no funciona.	App/AdministracionLocal/gestionarplantilla/Eliminar	2da Iteración	x	Abierta 30/06/2016 Cerrada	Funcionalidad	Esther García Naranjo.

Elemento	No.	No Conformidad	Ubicación de la No Conformidad	Etapa de Detección	Significativa	Estado NC	Clasificación de las NC	Responsable de la NC
						30/06/16		
App	15	En configuración de servidor al adicionar poner punto final en el mensaje.	App/AdministracionLocal/gConfigurarplantilla/Adicionar	1era Iteración	x	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	16	En configuración de servidor al adicionar, el mensaje tiene falta de ortografía.	App/AdministracionLocal/gConfigurarplantilla/Adicionar	1era Iteración	x	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.
App	17	En configuración de servidor al adicionar, falta el punto final.	App/AdministracionLocal/gConfigurarplantilla/Adicionar	1era Iteración	x	Abierta 30/06/2016 Cerrada 30/06/16	Ortografía	Esther García Naranjo.

Anexo 8. Acta de liberación del sistema por el grupo de calidad de CEGEL.

FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: *30*/06/2016

Emitida a favor de: Tesis *"Componente para la gestión de notificaciones para el marco de trabajo XEGFORT"*

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
App: Componente para la gestión de notificaciones para el marco de trabajo XEGFORT	*1.0*	*0*	*3*	*Evaluación dinámica Pruebas de Funcionalidad*	*30/06/2016*

Ing. Yordanis García Leiva
Asesor de Calidad CEGEL

Ing. Jorlen Machado Suástegui
Responsable de la liberación

Esther García Naranjo
Autora

Centro de Gobierno Electrónico

1