Universidad de las Ciencias Informáticas
“Facultad 3”

Componente para la gestión de notificaciones del sistema

Ventanilla Única Aduanera.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:

Pedro Pablo Alvarez Martínez

Tutor:

Ing. Yorlen Guirado Más.

Co-tutor:
Ing. Yasmany Avila Sarmiento.

Julio 2016

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Pedro Pablo Alvarez Martínez — (Autor)

Ing: Yorlen Guirado Más. — (Tutor)

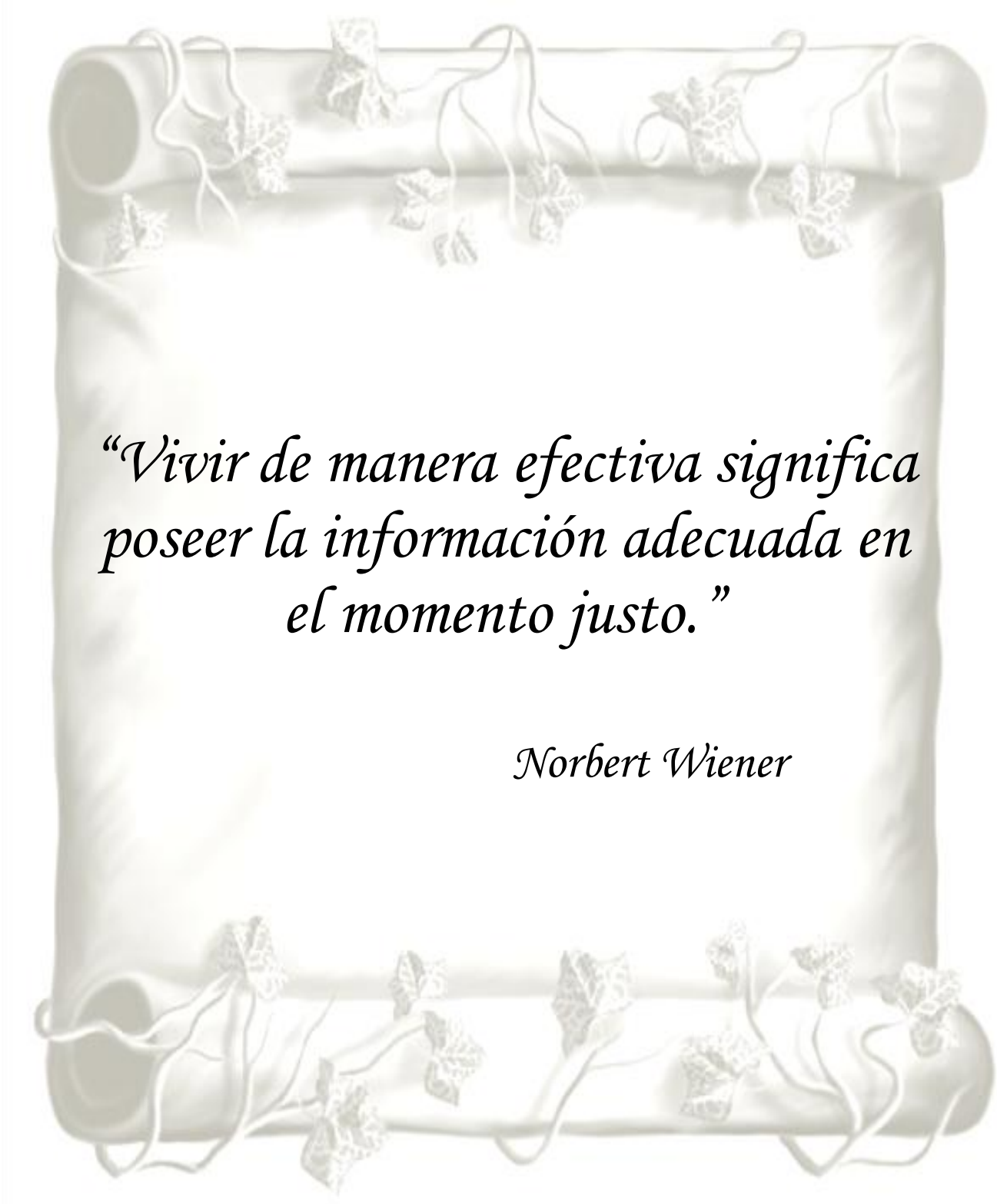
_____

Ing.: Yasmany Ávila Sarmiento
(Co-tutor)

DATOS DE CONTACTO

Ing. Yorlen Guirado Más (yorleng@uci.cu) Graduado de Ingeniero en Ciencias Informáticas en el año 2012. Actualmente labora como Jefe del proyecto Ventanilla Única.

PENSAMIENTO



AGRADECIMIENTOS

Primero a las tres mujeres de mi vida:

Mi madre porque en cada segundo de esta carrera hay una gota de tu sudor y amor. Y cuando hablo de la carrera me refiero al tiempo desde abril del 2003 cuando hable por primera vez de la UCI. Mi hermana por acompañarme en este camino desde que nació, por ser mi confidente y mi alegría. A mi novia Daimary por estar conmigo siempre, por seguirme, por aguantarme y por quererme.

A mi Familia que siempre se preocupo por mí.

A mi familia de Morón por cuidarme como si fuera uno más de sus hijos.

A mis nuevos hermanos Danay y Yoiler por acogerme, ayudarme, aconsejarme y hasta reganarme.

A mis tutores por el trabajo, tiempo y paciencia demostrados.

A todas las personas que hicieron posible mi traslado para la UCI.

A los estudiantes y profesores del quinto año de la facultad 3 que me hicieron sentir en familia y me facilitaron mucho la adaptación.

A todas las personas que han aportado a la realización de este trabajo y esta carrera.

A todos MUCHAS GRACIAS

DEDICATORIA

A mi madre María Esther que siempre ha estado a mi lado, siempre me ha sacado las mejores sonrisas en los peores momentos, ha preferido siempre mi felicidad antes que la suya y nunca me ha dejado caer, y si alguna vez he caído, siempre me ha dado la mano para levantarme.

A mi hermana Ana Carla porque el amor de hermana no tiene sustituto, porque me conoces tal y como soy, porque me aceptas a pesar de todas mis faltas, porque posiblemente pensarás que "no te queda de otra", pero siempre estás conmigo. Por todo eso: Eres mi hermana consentida, te quiero.

A mi novia Daimary por quererme, apoyarme, hacerme crecer como persona, porque sin ella esta carrera no hubiese sido lo mismo, porque no puedo pensar en lo que soy hoy sin ella, por desvelarse conmigo, por cargar conmigo y sobre todo por amarme como me ama y tener abierto el corazón para que yo la ame como nunca antes había amado a alguien.

RESUMEN

El presente trabajo tiene como objetivo desarrollar un componente para la gestión de las notificaciones automáticas y configurables en el sistema Ventanilla Única Aduanera.

Se realiza un estudio de algunas formas de notificación en tiempo real empleadas en aplicaciones web, con el objetivo de determinar los nuevos escenarios que se deberían cubrir. Además, se efectúa un estudio de las herramientas y tecnologías a utilizar a lo largo del desarrollo de la solución. Se identificaron y validaron los requisitos funcionales con que contaría la misma, mediante técnicas de captura y validación de requisitos.

Mediante el uso de estándares de codificación y patrones de diseño durante la implementación se logra un componente reutilizable, validado a través de las pruebas de software.

Con esta solución se pretende mejorar la forma en que los usuarios del sistema Ventanilla Única Aduanera reciben las notificaciones y darles opciones de crear nuevas y configurar las existentes.

Palabras Claves:

Notificación, componente, configuración, automáticas.

ÍNDICE

INTRODUCCIÓN

Las Tecnologías de la informática y las comunicaciones ilustran de la necesidad del ser humano de aumentar la calidad de la información que necesita para subsistir. La rapidez con que se es capaz de obtener la información juega un papel importante y realmente llega a marcar diferencia en un ámbito en que el más informado tiene ventaja sobre los demás (27).

Una de las formas de obtener informaciones es a través de la web y sus aplicaciones. A estas se acceden utilizando el navegador y se independizan del sistema operativo. Brinda, además, las facilidades de actualizar y de realizar mantenimientos sin tener que distribuir e instalar software a usuarios potenciales.

Para que la información interna de las aplicaciones web llegue a los usuarios se utilizan mensajes de error, advertencias del navegador y notificaciones creadas por la aplicación web.

El propósito de las notificaciones es informar de acciones entre usuarios dígase mensajes; envío de archivos o cambio en los estados en alguna entidad o componente del sistema. También se utilizan para informar errores de la aplicación o para anunciar la expiración de elementos dentro del sistema.

Dado el auge que tienen las Tecnologías de la Información y las Comunicaciones (TIC), las entidades que manejan gran cantidad de información y procesos necesitan para lograr una mayor eficiencia la informatización de sus mecanismos de trabajo. La Aduana General de la República de Cuba (AGR) es una de ellas.

Con el actual crecimiento continuo del comercio internacional y la interdependencia económica que vincula y norma las relaciones entre estados, se hace necesario para una buena aplicación de las políticas aduaneras, contar con personal altamente calificado junto a tecnologías y sistemas adecuados.

La Ventanilla Única Aduanera (VUA) es una herramienta digital donde se pueden realizar la totalidad de los trámites y pagos de las aplicaciones aduaneras. Permite a los involucrados gestionar a través de Internet y por una sola vía, los trámites requeridos por la AGR para la exportación, importación y tránsito de mercancías. (30)

La VUA se usa con los objetivos de:

- Ahorrar tiempo, dinero e incrementar la transparencia de los trámites.
- Aplicar las normas de control vigentes con mayor rapidez y eficacia.
- Lograr un almacenamiento centralizado de la información que se genera durante la ejecución del proceso.
- Aumentar nivel de gestión, control y seguimiento de los trámites.
- Facilitar la obtención de reportes que muestra el estado real en que se encuentran los trámites.
- Eliminar riesgo de destrucción de documentos.(31)

Para el funcionamiento de la herramienta se integran varios módulos, como portal, servicios, notificaciones, trámites y operaciones para los usuarios en general y para la administración los módulos control de acceso, envío/recepción, gestión de personas, gestión de contenidos y recepción de documentos.

El módulo de notificaciones está compuesto por 3 tipos de notificaciones: alertas, información y error; las mismas brindan información a los usuarios, que son generadas por la VUA y por el sistema de Gestión Integral de Aduanas (GINA) con el que interactúa. Este muestra a los usuarios mensajes resultantes de operaciones realizadas en el sistema. Según las especificaciones del componente, las notificaciones se predefinen directamente en el código fuente. Un ejemplo de ello es la definición de la notificación asociada a mensaje de contacto que se muestra a continuación.

```
//se crea el Evento y se lanza la notificación
$msg = '<strong>'.$mensajeContacto->getNombre().'</strong> ha escrito:<br>'.$mensajeContacto->getMensaje();
$event = new VUNotificacionesEvent('informacion', 'Nuevo mensaje de contacto recibido', $msg);
$dispatcher = $this->get('event_dispatcher');
$dispatcher->dispatch(StoreEvents::onContactMessageReceipt, $event);
return array(
    "textResponse" => $translator->trans('contacto.mensaje_guardado'),
    "form" => $this->createForm(new MensajeContactoType(), new MensajeContacto())->createView()
```

Figura 1: Código asociado a la notificación de mensaje de contacto

Al realizarse de esta forma se dificulta poder configurar el contenido de la información que se muestra, a quién o a quiénes van dirigidas, o la forma en la que van a ser enviadas. Además, los usuarios no cuentan con opciones de configuración, que les permita determinar las notificaciones que no desean recibir. En ocasiones se ven abrumados por la cantidad de notificaciones que reciben innecesariamente o que les llegan de forma repetida por correo electrónico y mensajes del propio sistema.

Actualmente las notificaciones que se emiten son las relacionadas con el envío de documentos a través de servicios web, contraseña próxima a expirar y por mensajes de contacto.

Como parte de la explotación del sistema se han identificados nuevos escenarios en los que podrían ser creadas notificaciones como son: errores ocurridos en el servidor, cambio de estado en las entidades del sistema, nomencladores próximos a expirar y acciones realizadas sobre las cuentas de usuarios.

En aras de la problemática se define como **problema a resolver**: ¿cómo gestionar las notificaciones automáticas en el sistema Ventanilla Única Aduanera con el fin de que sean configurables?

El **objeto de estudio** de investigación se enmarca en la gestión de notificaciones automáticas en sistemas web. Teniendo en cuenta el problema a resolver, se establece como **objetivo general** de la investigación:

Desarrollar un componente para la gestión de las notificaciones automáticas en el sistema Ventanilla Única Aduanera que permita su configuración.
A partir del objetivo general se derivan los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación relacionado con la gestión de notificaciones automáticas en sistemas web.
2. Realizar el análisis y el diseño del componente para gestionar y configurar las notificaciones automáticas en el sistema VUA.
3. Implementar un componente para gestionar y configurar las notificaciones automáticas en el sistema VUA.
4. Validación de la solución propuesta mediante pruebas funcionales y de aceptación.

Según lo planteado anteriormente se establece como **campo de acción:** la gestión de notificaciones en el sistema VUA.

En el transcurso de la investigación utilizan diferentes métodos científicos tales como:

- **Histórico - lógico:** se usó para comprender la estructura y funcionamiento de los sistemas de notificaciones tanto nacionales como internacionales. Además, permitió entender el desarrollo del envío de notificaciones en el tiempo.
- **Análisis - síntesis:** permitió mediante el análisis de la situación existente descomponer el problema en dos objetos a estudiar: la generación de notificaciones y la configuración de las mismas. Cada uno de ellos por separado fue descompuesto en: primeramente investigar sobre los diferentes tipos de notificaciones que se pueden generar y posteriormente en las eventualidades que se necesiten notificar. Finalmente, los resultados de todos estos elementos estudiados por separado se unen para dar una respuesta al problema en análisis a través de la creación de un componente de gestión de notificaciones.
- **Modelación:** la modelación es el método que opera en forma práctica o teórica con un objeto, no en forma directa, sino utilizando cierto sistema intermedio, auxiliar, natural o

artificial. La modelación empleada fue la teórica que utiliza símbolos para designar las propiedades del sistema que se desea estudiar, representa las características y las relaciones fundamentales del objeto.

- **Inductivo - deductivo:** se utilizó en la aplicación de casos de pruebas al sistema, llegando a diferentes resultados a partir de las respuestas proporcionadas por este.

El contenido del presente trabajo está estructurado en tres capítulos distribuidos de la siguiente manera:

Capítulo 1: Fundamentación Teórica. Se realiza un estudio del estado del arte sobre las técnicas de notificación en tiempo real empleadas en aplicaciones web. Se describen tanto las tecnologías, metodologías y herramientas definidas para el desarrollo de la propuesta de solución.

Capítulo 2: Propuesta de solución. Se resumen las bases del desarrollo del sistema propuesto, las funcionalidades que debe cumplir y su especificación, la arquitectura, los modelos de diseño y de datos, los patrones utilizados y los estándares de codificación empleados.

Capítulo 3: Implementación y pruebas. Se especifican aspectos de interés para el proceso de implementación tales como los estándares codificación para el código fuente del sistema, el diagrama de componentes que conforma la solución. Se especifican además los resultados las pruebas de software realizadas y finalmente se valora qué beneficios trae el resultado obtenido.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción.

En este capítulo se explican los conceptos relacionados con el desarrollo de la investigación. Se lleva a cabo un estudio acerca de las diferentes herramientas a nivel internacional y nacional, para la creación de un sistema que integre el envío de notificaciones. Se especifica la metodología de desarrollo de software escogida, así como las tecnologías, software y lenguajes utilizados para la implementación del sistema.

1.2. Sistemas de notificaciones

Según la Real Academia de la Lengua Española, "*las notificaciones son acciones o efectos de notificar.*

Notificar constituye comunicar formalmente una resolución o dar una noticia con propósito cierto" (1).

Un Sistema de notificación hace referencia a los procesos y la tecnología implicada en la estandarización, formato, comunicación, retroalimentación, análisis, aprendizaje, respuesta y difusión del aprendizaje generado por el registro de eventos. Es la acción que se adopta tras el registro, la respuesta constructiva, y no el registro en sí lo que provoca los cambios y aporta valor (2). Los sistemas de notificación de eventos son una parte de la "cultura de seguridad" donde se entienden los eventos adversos como una oportunidad para aprender y mejorar (3). Por lo que se llega a la conclusión de que un sistema de notificación de eventos es un conjunto organizado de información relacionada con las acciones que se realizan en una aplicación, es el encargado de mantener informados a los usuarios que interactúen con el sistema sobre las acciones que se efectúan.

1.3. Sistemas informáticos en tiempo real

Un **sistema en tiempo real** es un sistema informático que interactúa con su entorno físico y responde a los estímulos del entorno dentro de un plazo de tiempo determinado. No basta con que las acciones del sistema sean correctas, sino que, además, tienen que ejecutarse dentro de un intervalo de tiempo determinado. (37)

A continuación, se realiza un análisis de algunos sistemas de notificaciones utilizados en el mundo para identificar las opciones que brindan, cómo funcionan, qué tan configurables son. También se hace un análisis de las soluciones realizadas en la Universidad de las Ciencias Informáticas (UCI) que cuentan con sistemas o componentes de notificación.

1.3.1 Facebook notifications.

La red social Facebook contiene el módulo Notifications que permite obtener las notificaciones directamente al escritorio del usuario. Actualmente en caso de que no se tenga abierta la cuenta de Facebook, ni el correo para que notifique que se tiene una nueva notificación, no constituye un problema, gracias a este sistema. Este componente indica cuando se tienen nuevos mensajes, eventos e invitaciones. Además, permite actualizar el estado desde la aplicación. (4)

Las notificaciones son actualizaciones sobre la actividad de Facebook. Los tipos de notificaciones que existen dependen de la plataforma utilizada.

Ordenador y móvil

- Notificaciones de alerta: son los números de color rojo que aparecen sobre los iconos de mensaje, solicitud de amigo y notificación.
- Notificaciones por correo electrónico: notificaciones recibidas por correo electrónico.
- Notificaciones de inserción: son notificaciones que aparecen en los dispositivos móviles de los usuarios cuando no están usando Facebook de forma activa.

Solo en el ordenador

- Notificaciones emergentes: son las notificaciones que aparecen en la pantalla cuando se encuentra la sesión iniciada en Facebook.

Actualmente permite notificar los siguientes eventos:

- Algún amigo del usuario cambia su estado o su perfil.
- Envían al usuario una petición de amistad, evento o grupo.
- Alguien escribe en el muro de alguno de los amigos del usuario.
- Una foto es etiquetada con el nombre de uno de los amigos del usuario.

Además, el módulo Notifications de Facebook permite configurar de qué tema se reciben las notificaciones, editar la configuración de notificaciones para un grupo y activar o desactivar las notificaciones de personas, grupos o páginas que se siguen. (4)

1.3.2 Organización Mundial del Comercio (OMC):

La Organización Mundial del Comercio cuenta con un sitio para el trabajo con Documentos en línea que permite acceder a la documentación oficial de dicha institución. La base de datos de documentos del sitio, se actualiza diariamente, contiene más de 100.000 documentos en los tres idiomas oficiales, distribuidos a partir de 1995. Todos los documentos de la OMC están disponibles en Formato de Documento Portable (PDF) y en Formato de Microsoft Word (DOC). La aplicación incluye un registro descriptivo de cada documento archivado. Es posible consultar en línea todos esos documentos y descargarlos.

Tiene un componente de notificaciones que permite buscar documentos de notificación, por miembro que notifica o por prescripción jurídica de la OMC, utilizando la base de datos del Registro Central de Notificaciones (RCN). La función de Búsqueda avanzada incluye otros criterios, como la signatura[1], el tema de la prescripción y la clasificación. En la función de Búsqueda avanzada también se pueden realizar búsquedas dentro del propio texto del documento con la función de búsqueda de "Texto completo".

1.3.3 Tuenti Monitor

Tuenti Monitor muestra avisos de nuevos mensajes, comentarios e invitaciones que sean

[1] Señal que se pone en un objeto para distinguirlo de otros, especialmente la señal formada por números y letras que se pone en una parte visible de un libro o de un documento para clasificarlo e indicar el lugar que ocupa en una biblioteca o en un archivo

recibidas en Tuenti desde la barra de tareas. Una vez instalado, Tuenti Monitor solicita los datos de la cuenta de Tuenti. La configuración es muy sencilla y permite especificar el intervalo de tiempo con el que se mostrarán notificaciones y si se quiere que el mismo arranque con Windows. Las notificaciones de Tuenti Monitor se muestran en una discreta nota emergente. Si se hace un clic sobre el icono, mostrará otra vez las notificaciones, incluso si no se tiene ninguna. (6)

Por otro lado, un doble-clic abrirá la página de Tuenti en el navegador sin necesidad de volver a introducir usuario y contraseña, algo muy cómodo si se accede a Tuenti con frecuencia.

Para utilizar Tuenti Monitor se necesita estar en un sistema operativo de Microsoft, ya que entre sus requerimientos está tener instalado Microsoft Net Frenwork 3.5.

1.3.4 Módulo de notificaciones del Entorno Virtual de Aprendizaje

El Entorno Virtual de Aprendizaje (EVA) es una plataforma dentro de la UCI que permite a los estudiantes desarrollar actividades académicas y facilitar su estudio cotidiano; además, permite a los profesores controlar las tareas desarrolladas por cada uno de sus estudiantes; esta plataforma, tiene como método de notificación, enviar un correo, al inscribirse en un curso, cuando un profesor califica alguna tarea subida por el estudiante y también cuando responde un comentario publicado por este en algún foro. Este correo se genera de forma automática y el sistema brinda la posibilidad de desactivar este servicio.

1.3.5 Componente para la gestión de notificaciones en el marco de trabajo Sauxe.

El componente para la gestión de notificaciones en el marco de trabajo Sauxe brinda una solución que permite a los usuarios del marco de trabajo Sauxe conocer la ocurrencia de un evento determinado en el instante que suceda en el servidor de aplicaciones, sin que tengan estos que intervenir para que se actualice la información. Con esta solución se mejora el tiempo de respuesta que puede tener un usuario determinado para realizar una tarea relacionada con el desencadenamiento de dicho evento.

El componente permite:

- Crear un catálogo con las aplicaciones que se encuentran suscritas a cada evento y con los roles que *escuchan* los mismos.
- Gestionar a través de los roles, el código de mensaje que recibe cada rol.
- Gestionar las notificaciones que envía y recibe cada sistema.
- Mostrar historial de mensajes de un usuario.

Las notificaciones son enviadas directamente desde el código fuente y no se pueden configurar ni adaptar a otro marco de trabajo.

En los sistemas antes descritos se muestran elementos a tener en cuenta a la hora de seleccionar el comportamiento de una solución. Facebook Notifications permite que se configure cuándo y por cuál vía recibir las notificaciones al igual que Tuenti monitor, pero brindan pocas opciones a la hora de crear nuevas notificaciones no contempladas en el sistema. Los componentes de notificaciones del EVA, del marco de trabajo Sauxe y la OMC generan la notificación directamente en el código fuente. Esto no permite configurarlos y adaptarlos a nuevos sistemas.

Un elemento importante de los componentes analizados es que se envían notificaciones en tiempo real, esto permite a los usuarios obtener la información de lo que ocurre en el sistema sin la necesidad de actualizar la aplicación.

A continuación, se muestra una tabla en la que se presentan los resultados de la comparación entre los distintos sistemas estudiados. Los indicadores que se encuentran en la parte superior de la tabla y que sirven de base para la comparación fueron seleccionados de acuerdo a la necesidad existente de que la solución deseada cumpla con ellos.

Sistemas	Configurable	Forma de envío	Gestión de las notificaciones	Reutilización del código
Facebook notifications	Si	APP/Email/SMS	Si	No

OMC	Si	EMAIL	No	No
Tuenti monitor	Si	EMAIL/APP	No	No
EVA	No	EMAIL/APP	No	No
Notificaciones del marco de trabajo Sauxe	No	EMAIL/APP	No	Si

Tabla 1.1: Comparación entre sistemas estudiados

En la tabla 1.1 se muestra que todos los sistemas envían las notificaciones en tiempo real, solo Facebook notifications cumple los tres primeros indicadores por lo que se puede tomar como un referente, pero no se puede modificar su código y utilizarlo.

A través del estudio de las soluciones anteriormente mencionadas se obtienen una serie de elementos que presentes en un sistema de notificaciones y que deben estar presentes en la solución a desarrollar.

Entre los elementos con que cuentan los sistemas analizados están: la forma en que se envían las notificaciones y la forma en que se manejan los eventos que desencadenan el envío de notificaciones.

Entre las formas en que se envían las notificaciones una de las más usadas actualmente es a través de email, ya que todos los sistemas estudiados la utilizan, la misma tiene ventajas como: económico, ágil, versátil, medible, personalizable, eficaz, analizable, compartible y amigable. (31)

Otra manera de enviar notificaciones es directamente desde la aplicación; para ello se utilizan varias formas de las cuales algunas se describen a continuación.

1.4. Formas de enviar notificaciones en tiempo real

Las tendencias de envío de notificaciones utilizadas en aplicaciones web permiten a los usuarios recibir estas tan pronto como la información es publicada, sin necesidad de chequear

manualmente la fuente original para obtener actualizaciones. (7) A partir de esta definición se exponen a continuación las formas de envío de notificaciones estudiadas.

1.4.1 Web Sockets

Web Sockets según W3C (World Wide Web Consortium) es una tecnología de HTML5 que permite a las aplicaciones web mantener una comunicación bidireccional con procesos en el lado del servidor. Se realiza entre cliente y servidor utilizando un mecanismo para ponerlos en contacto. Tanto el servidor como el cliente o clientes utilizan una aplicación, la cual es responsable de interactuar con los sockets, dando responsabilidad al sistema operativo de utilizar un proceso para crear, utilizar y luego cerrar el socket cuando ya se ha transmitido el mensaje. El usar procesos del sistema operativo puede representar un gran consumo de recursos de ambas partes. Además de representar una relativa demora, la cual se puede incrementar considerablemente si se usa Web Socket Secureconnections (wss), porque el mismo utiliza TransportLayer Security (TLS) y representa un costo adicional en cuanto al tiempo que puede tardar para encriptar. (8)

Aunque la tecnología Web Sockets es indiferente a la conexión sobre servidores proxy o cortafuegos. Implementa una negociación compatible con Protocolo de Transferencia de Hipertextos (HTTP por sus siglas en inglés). Lo hace para que los servidores HTTP puedan compartir sus puertos HTTP y Protocolo de Transferencia de Hipertextos Seguro (HTTPS) por defecto (80 y 443 respectivamente) con una pasarela o servidor WebSocket. Los mensajes enviados corren el riesgo de ser bloqueados, provocando que la conexión falle. (8)

También es importante destacar que esta tecnología usada en navegadores web, presenta algunas desventajas. Existen versiones de navegadores que no lo implementan o lo hacen parcialmente. Esto unido a vulnerabilidades encontradas en la seguridad de esta tecnología hacen que los navegadores como Firefox y Opera anuncien que dejarán de ofrecer soporte.

1.4.2 COMET

El término COMET describe a las aplicaciones donde el servidor se mantiene enviando los datos, o manteniendo un cúmulo continuo de datos a la aplicación cliente, en vez de tener al

navegador realizando peticiones al servidor para actualizar el contenido, es decir COMET cambia fundamentalmente la naturaleza de la comunicación realizada entre la arquitectura Cliente-Servidor. (9)

COMET se basa en que el servidor envía datos, aunque el cliente no los pida (HTTP Push). Si se hace una llamada, el canal se queda abierto y el servidor va enviando información, o lo que es lo mismo, que el servidor va a devolver el resultado en partes. En otras palabras, todas las aplicaciones de COMET utilizan conexiones HTTP de larga duración para reducir la latencia con la cual los mensajes son enviados al servidor. En esencia no realizan pedidos ocasionalmente al servidor. COMET utiliza XMLHttpRequest para la entrega de datos entre el cliente y el servidor a través del protocolo HTTP. También es conocido como Server Push o HTTP Push. A continuación, algunas desventajas relacionadas con el uso de COMET. (9)

Este método podría significar un desperdicio en términos de sockets e hilos, y también representaría una sobrecarga para los cortafuegos, y los balanceadores de carga, ya que los mismos deben chequear frecuentemente los datos enviados por el canal de información que efectúa COMET.
Este método al mantener abierta la conexión por un largo período, puede presentar problemas de escalabilidad en una aplicación.

1.4.3 Jabber

Jabber es un sistema basado en el lenguaje XML para el intercambio en cuasi tiempo real de mensajes y *presencia*.

Presencia: Indica si una entidad Jabber está disponible para la comunicación. Incluye disponibilidad básica (o sea, conectado o desconectado) y también indicadores de estado que determinan tipos de disponibilidad. (10)

1.4.4 Protocolo XMPP/Jabber

Es el protocolo usado por Jabber, también el usado por otras aplicaciones como Google Talk y LiveJournalTalk, por lo que son interoperables. El protocolo está estandarizado por el Grupo de Trabajo de Ingeniería de Internet (IETF). Está basado en un conjunto de tecnologías XML

para aplicaciones en tiempo real. Fue desarrollado originalmente como un marco de trabajo de aplicaciones de mensajería instantánea y presencia para escenarios de empresas. Utiliza ficheros en formato XML para hacer las transferencias de mensajes entre sus entidades cliente-servidor o servidor-servidor. (10)

Características

- **Protocolo abierto:** con todas las ventajas del software libre, se puede programar un servidor o un cliente o ver el código.
- **Descentralizado:** se puede crear un servidor para Jabber y se puede separar o unir al resto de la red Jabber.
- **Extensible:** se puede ampliar con mejoras sobre el protocolo original. Las extensiones comunes son manejadas por la XMPP Standards Foundation**.**
- **Seguro:** cualquier servidor Jabber está aislado del exterior. El servidor de referencia permite SSL para comunicaciones cliente-servidor y algunos clientes aceptan GPG[2] como cifrado de las comunicaciones usando cifrado asimétrico. 1)

1.4.5 Servidores que utilizan el protocolo XMPP/Jabber

EJabberd

Es un servidor multiplataforma desarrollado en lenguaje Erlang y de código abierto. La instalación es sencilla, ya que la distribución de Erlang provee de todos los componentes que necesitará eJabberd. Soporta Ipv6. Implementa casi de forma completa el XMPP y varias de las extensiones J.E.P (JabberExtensionProtocol). El número de sitios que lo utilizan como solución no es muy elevado, no tiene una comunidad de usuarios muy amplia y eso se traduce en un soporte menos eficiente. En cuanto a la parte de desarrollo, el estar implementado en Erlang es lo que lo pone en desventaja con respecto a servidores como es el caso de Jive y Jabberd (hechos en java y C/C++ respectivamente), ya que está limitado el desarrollo a una comunidad más reducida de personas que conozcan el lenguaje. Por otro lado, la configuración de eJabberd no resulta ser tan sencilla. Pero posee una interfaz gráfica que

[2] **GNU Privacy Guard** (**GnuPG** o **GPG**) es una herramienta de cifrado y firmas digitales

permite configurarlo, la cual es accesible vía web a la dirección **http://[Url donde se ubica el servidor]/admin**, para acceso remoto. Esta interfaz permite administrar las ACL[3]. (11)

Openfire

Openfire antes llamado Servidor Wildfire, es un servidor desarrollado en Java que provee licencias comerciales y GNU. La administración del servidor se hace a través de una interfaz web, que establece la conexión por defecto utilizando el puerto 9090 (HTTP) y 9091 (HTTPS). Los administradores pueden conectarse desde cualquier lugar y editar la configuración del servidor, agregar y borrar usuarios, crear cuartos de conferencia permanentes. Openfire implementa las siguientes características: presenta un panel de administración web al cual se puede acceder por la dirección http://**[Url donde se ubica el servidor]**:9090, y una interfaz ajustable para agregar plugins para particularizar su funcionamiento, agregar o modificar funcionalidades. Utiliza SSL/TLS para la encriptación durante la transmisión de datos a través de las comunicaciones cliente-servidor o servidor-servidor. Puede interactuar con otros servicios de mensajería instantánea mundialmente conocidos como MSN, Google Talk, Yahoo Messenger, AIM, ICQ y es capaz de ofrecer estadísticas del servidor, mensajes y paquetes. Tiene soporte para la autenticación vía Certificados, Kerbeos, LDAP, MS SQL, MySQL, Oracle y Postgres. (12)

1.5. Valoración de las formas de enviar notificaciones

Al utilizar un servidor que implementa el protocolo XMPP\Jabber se obtiene de forma *nativa* la posibilidad de gestionar usuarios y grupos de usuarios, lo cual es una ventaja que debe ser aprovechada, ya que además representa una manera de organizar los emisores y destinatarios de las notificaciones.

El estudio realizado demuestra que las técnicas de notificación en tiempo real Web sockets y COMET, aunque cubren escenarios comunes, presentan algunas desventajas que pueden atentar contra el buen funcionamiento del componente como Web Sockets, donde las versiones antiguas de los navegadores no tienen soporte para esta técnica. En el caso de la técnica COMET puede sobrecargar el canal de comunicación. Por las razones antes

[3] Un **ACL** es una lista que especifica los permisos de los usuarios sobre un archivo, carpeta u otro objeto.

expuestas se determina que el protocolo XMPP sea la técnica que garantice el mecanismo requerido para enviar las notificaciones en tiempo real en el componente de notificaciones. El hecho de que este protocolo utilice el lenguaje XML para el proceso de comunicación entre sus extremos representa una ventaja, ya que el mismo es la tecnología que le permite compartir la información de una manera segura, fiable, fácil. Además, XML permite al programador dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, pues algunas son tediosas como la validación de estos o el recorrido de las estructuras son soportadaspor el lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello.

El metalenguaje aparece como un estándar que estructura el intercambio de información entre las diferentes plataformas. Por estas razones, no importa el lenguaje o plataforma que se encuentre en los extremos, los datos enviados o recibidos podrán ser tomados, comprendidos y manejados, lo cual asegura que el flujo de datos sea seguro y sobre todo rápido sin muchas complicaciones. Como servidor que implemente el protocolo XMPP/Jabber se utilizará el servidor Openfire, ya que el mismo está escrito en java, el cual es un lenguaje potente y con una gran comunidad que brinda soporte al mismo. Además, brinda una interfaz para extensiones o plugins y comparte con Symfony la característica de gestionar usuarios, lo cual ayuda a un mejor control y configuración del componente de notificaciones.

1.6. Manejo de eventos

1.6.1 Symfony2

Symfony2 cuenta con un componente para el manejo de eventos, el EventManager. Este componente consta de 3 partes, por una parte, los eventos que son los que se lanzan, por otra parte, están los escuchas o listeners que captan los eventos y finalmente está la clase EventDispatcher (despachador de eventos) que se encarga de lanzar los eventos y avisar a los listeners que están suscritos a un determinado evento.

El despachador de eventos de *Symfony2* implementa el patrón observador en una manera sencilla y efectiva para hacer todo esto posible y para realmente hacer extensibles los proyectos.

Un ejemplo simple desde el componente HttpKernel de Symfony2. Una vez creado un objeto Respuesta, puede ser útil permitir que otros elementos en el sistema lo modifiquen (por ejemplo, se añaden algunas cabeceras de caché) antes de utilizarlo realmente. Para hacer esto posible, el núcleo de *Symfony2* lanza un evento —kernel.response—. Así es como funciona:

- Un *escucha* (objeto *PHP*) le dice a un objeto *despachador* central que quiere escuchar el evento kernel.response;
- El núcleo de *Symfony2* dice al objeto *despachador* que difunda el evento kernel.response, pasando con este un objeto Evento que tiene acceso al objeto Respuesta.
- El despachador notifica a (es decir, llama a un método en) todos los escuchas del evento kernel.response, permitiendo que cada uno de ellos haga modificaciones al objeto Respuesta.

Los eventos se dividen en dos partes:

- **Nombre del evento:** es un nombre único y será el nombre que se le pase al EventDispatcher al lanzar el evento y el nombre al que se suscriben los *listeners*, este nombre debe seguir unas convenciones de nombres.
- **Objetos Event:** cuando se lanza un evento lleva asociado un objeto Event que tiene que extender de la clase Event o usar la propia clase (también se puede lanzar un evento sin objeto, pero internamente lo crea). Este objeto entre otras cosas tiene un método stopPropagation que permite parar la propagación. Lo habitual es extender esta clase para añadirle un atributo en la que se guarda una referencia algún objeto que pasar a los listeners, ya que cuando se les invoque recibirán este evento.

Para lograr un control de lo que ocurre en las entidades del sistema Symfony2 se apoya en Doctrine que es un Object Relational Mapping (ORM) que cuenta con un completo sistema de eventos que cubre casi todos los casos que se producen en el sistema, principalmente los del ciclo de vida, siendo el EventManager el encargado de controlarlos. Por nombrar algunos de estos eventos, preRemove se produce justo antes de que el EntityManager ejecute la operación de borrado en una entidad determinada, prePersist se produce en una entidad justo antes de que el EntityManager persista en base de datos esa entidad, postUpdate es el evento que se produce después de que se haya llevado a cabo la operación de actualización de los datos de la entidad en base de datos, entre otros. La captura de estos eventos permite mantener un control sobre lo que ocurre con las entidades del sistema y representa una opción a utilizar en el envío de las notificaciones.

1.7. Tecnologías utilizadas

El componente de notificaciones está enmarcado en el proyecto VUA del Centro de Informatización de Entidades (CEIGE) por tanto las tecnologías a utilizar son definidas por el centro. A continuación, se mencionan y se explican.

1.7.1 Lenguaje de modelado UML v2.1

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. (15)

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

1.7.2 Herramienta de modelado Visual Parading for UML v 8.0

Es una herramienta CASE[4], diseñada para contribuir al desarrollo de software. Soporta los principales estándares como UML, SysML, BPMN 16 y XML. Ofrece un conjunto de herramientas a los equipos de desarrollo de software, necesarios para la captura de requisitos, la planificación de software, la planificación de pruebas, el modelado de clases y el modelado de datos. (15)

1.7.3 Lenguajes de programación.

Un lenguaje de programación es aquel elemento que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que se ponen a disposición del programador para que éste pueda comunicarse con los dispositivos hardware y software existentes. (13) A continuación, se describen los lenguajes de programación utilizados.

PHP v5.4

PHP (acrónimo de "PHP: HypertextPreprocessor") es un lenguaje de código abierto interpretado, de alto nivel, embebido en páginas HTML y el lenguaje está orientado al desarrollo de aplicaciones web, que son interpretadas del lado del servidor. Su sintaxis es muy similar a lenguajes como C y PERL. Puede ser utilizado en casi todos los sistemas operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Tiene gran rapidez en la ejecución y bajos requerimientos de consumo en los sistemas donde es desplegado.

Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de poseer una comunidad de desarrolladores que intercambian experiencias, de esta forma cuando se presenta un problema es muy fácil obtener documentación para darle solución de forma rápida y sin costo alguno. (13)

[4] **Herramientas CASE** (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero.

Motor de plantillas Twig

Para el trabajo con las vistas se utiliza el motor de plantillas Twig. Es un motor y lenguaje de plantillas para PHP muy rápido y eficiente. Es utilizada por Symfony2. La sintaxis de Twig se ha diseñado para que las plantillas sean concisas, fáciles de leer y de escribir. Las plantillas de Twig se compilan a código PHP nativo, por lo que el rendimiento y el consumo de memoria son similares al de las plantillas PHP.

JavaScript v1.10.3

JavaScript es un lenguaje basado en objetos, que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que permite la interacción entre su contenido y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (14)

JQuery como biblioteca JavaScript

Se escoge JQuery como biblioteca JavaScript para crear una presentación estéticamente aceptable y amigable para los usuarios, porque permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. JQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de más códigos. Es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio (20).

1.7.4 Symfony2 v 2.8.6 como marco de trabajo de desarrollo

Symfony2 es un marco de trabajo PHP que facilita el desarrollo de las aplicaciones web. Se encarga de todos los aspectos comunes de las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto. Se

anunció por primera vez a principios de 2009 y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. Ha sido ideado para aprovechar al límite todas las nuevas características de PHP 5.3 y por eso es uno de los marcos de trabajo PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un proyecto. (16)

La principal razón por la cual se selecciona Symfony2 es que la herramienta VUA está desarrollada en este marco de trabajo.

También Symfony2 como marco de trabajo es independiente del sistema gestor de bases de datos, emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación Web, además que cuenta con una amplia documentación traducida al español. Por las razones expuestas anteriormente se considera que es la mejor opción para el desarrollo del módulo, ya que la calidad que brinda al código fuente y la cantidad de documentación disponible es favorable para desarrollar el mismo.

1.8. Metodología de desarrollo

Para el desarrollo de cualquier producto de software se realizan una serie de tareas entre la idea inicial y el producto final, un modelo de desarrollo establece el orden en el que se harán las cosas en el proyecto, provee requisitos de entrada y de salida para cada una de las actividades (17).

Una metodología impone un proceso de forma disciplinada sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente. Define una representación que facilita la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema.(39)

1.8.1 Metodología de desarrollo propuesta

El modelo de desarrollo propuesto a utilizar para desarrollar el componente de notificaciones está definido por la UCI para los proyectos o sistemas que se desarrollen en la misma, y fue creado teniendo en cuenta las principales características con las que cuenta la universidad.

Proporciona una guía para regir el proceso de desarrollo de software, centrado en la arquitectura. Dicho modelo está basado en principios, prácticas propuestas y algunos elementos de las metodologías AUP y CMMI-DEV v1.3. El mismo fue elaborado teniendo en cuenta las características especiales que presenta la UCI. (19)

1.9. Conclusiones parciales

Después de un estudio de los principales conceptos asociados a la investigación, así como de algunas aplicaciones similares existentes para una mejor comprensión del problema en cuestión se concluye que:

- A través del análisis de soluciones existentes, se determinó que ninguna podría ser utilizada para la solución del problema, por lo que se hace necesaria la implementación de un nuevo componente en el que se van a tener en cuenta los elementos similares que presentan las herramientas analizadas, como son la forma de envío de notificaciones a través de email y desde la propia aplicación web y el manejo de los eventos.
- Se determinó que las herramientas y metodología de desarrollo a utilizar serían las mismas del entorno de desarrollo del proyecto al que pertenece el componente, para garantizar su correcta integración.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción.

En el presente capítulo se desarrolla la propuesta de solución para el componente de notificaciones. Comienza con la realización de un modelo de dominio de la aplicación; se identifican y describen los requisitos funcionales y no funcionales, se muestran algunos artefactos y resultados generados por el modelo de desarrollo y que fueron obtenidos durante la fase de análisis y diseño del componente. Además, se da una breve descripción de los patrones de diseño y arquitectónicos utilizados en la solución.

2.2 Propuesta de solución

La solución propuesta consiste en el desarrollo de un componente que permita gestionar y configurar las notificaciones en el sistema VUA. Dicho componente se ha de desarrollar usando el marco de trabajo Symfony2 debido a que es la tecnología empleada en él. Como resultado se obtendrá un *bundle* utilizable en cualquier aplicación creada con el mencionado marco de trabajo, pero esencialmente en VUA.

El *bundle* en cuestión constará de cuatro partes, mediante las cuales se encontrarán en el código del sistema, se escucharán y gestionarán los eventos que los desarrolladores desean manejar. A continuación, se describe cada una de esas partes.

Parseador (“parser” en inglés): es la parte del *bundle* que se encargará de encontrar y almacenar en memoria todos los eventos configurados en el sistema, para ello los programadores deben vincular los eventos que desean manejar con los eventos que escucha este parseador. Existen dos clases fundamentales con las cuáles realizar el vínculo, estas son NotificablePersistenceEvent.php y NotificableEntityInterface.php. De la primera deben heredar todas las clases en las cuales se desee lanzar eventos no relacionados con el ciclo de vida de eventos (livecycle event en inglés) definidos por el Manejador de Objetos Relacionales (ORM por sus siglas en inglés) Doctrine2. Y de la segunda las entidades de las cuales sí desea escuchar esa clase de eventos.

Configuración de usuario: Esta parte va a estar dividida en una que se va a encargar de mostrar las notificaciones nuevas a los usuarios (las que no ha leído), para ello va a hacer uso

del servidor jabber. Y otro que va a mostrar todas las notificaciones recibidas por el usuario y va a permitir realizar operaciones de búsqueda y filtrado de las mismas.

Gestor de notificaciones: es la parte del *bundle* en la cual se gestionará y configurará de forma visual la creación de las notificaciones a partir de los eventos parseados. Es donde se ha de especificar una serie de elementos relacionados con la notificación, entre ellos el mensaje a mostrar, así como los implicados y las vías por las cuales se ha de enviar. El mensaje podrá ser personalizado a través de los parámetros provistos por el evento relacionado.

Escucha ("listener" en inglés): es la parte del bundle que se encargará de escuchar todos los eventos parseados y guardados en memoria, y crear las notificaciones relacionadas en dependencia de cómo se hayan gestionado estas.

Ha de contar con una serie de funcionalidades, entre ellas destacan las relacionadas con los componentes mencionados: crear evento, crear notificación y lanzar notificación. Con la solución propuesta se obtendrá un componente capaz de responder a la problemática que se desea resolver.

Para la realización del *bundle* antes descrito se transitará a través de las fases descritas por la metodología AUP y se generarán los artefactos correspondientes a cada una de ellas.

Para la realización del análisis del negocio se crea el siguiente modelo de dominio.

2.3 Modelo de dominio:

Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. El modelo de dominio es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir (28). A continuación, se muestra el modelo de dominio de la solución.

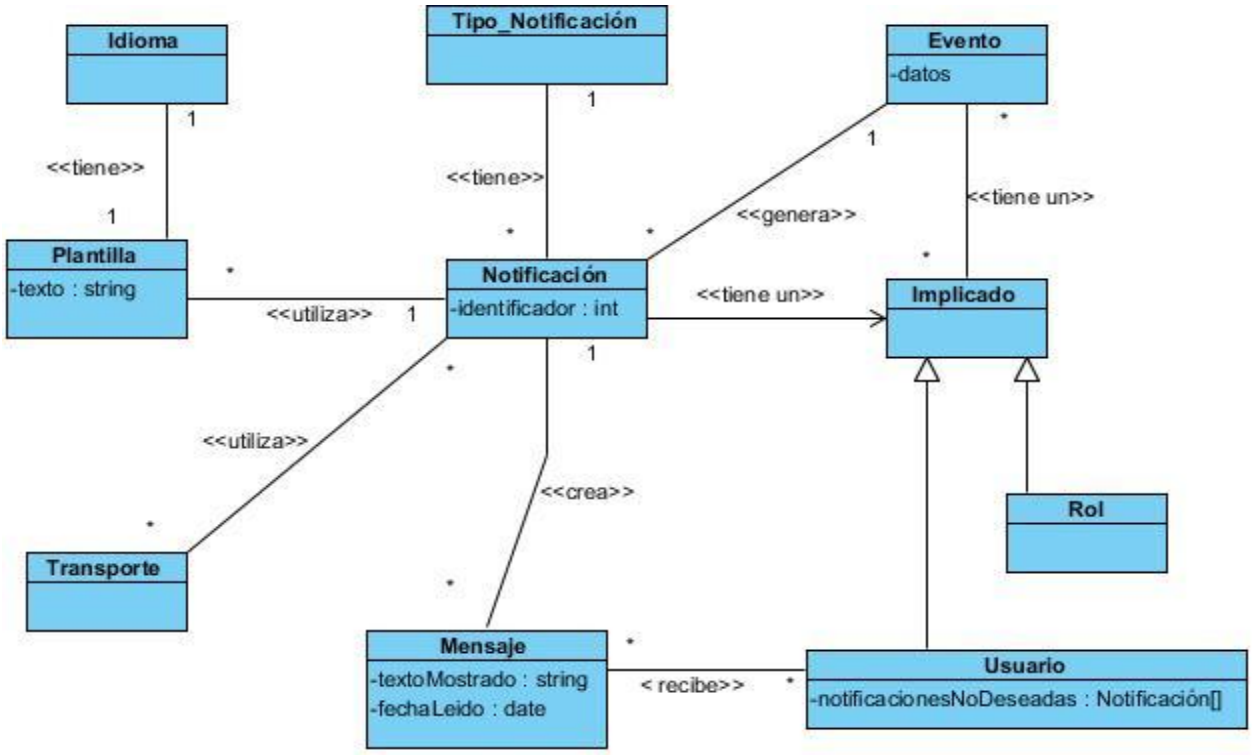


Figura 2.1 Modelo de dominio

Plantilla: se define para ser utilizada por las notificaciones. Contiene el mensaje predefinido que se le mostrará al usuario y los espacios para incluir las variables que personalicen el mensaje.

Idioma: define el idioma en el que se encontrará la plantilla para permitir que el sistema maneje las notificaciones en relación al idioma preferido por el usuario.

Transporte: define por qué vía será enviado el mensaje de notificación (Email, Aplicación).

Tipo_Notificación: permitirá definir los diferentes tipos en los que se podrán clasificar las notificaciones.

Evento: es la acción del sistema a partir de la que se podrá crear una notificación.

Implicado: es a quién o a quiénes será enviada la notificación, pueden ser un usuario o un grupo de usuarios agrupados por roles.

Notificación: es la configuración de la notificación a enviar cuando se ejecute el evento.

Mensaje: es el mensaje que se le muestra al usuario; él mismo decide si recibirlo o no.

2.4 Requisitos de software:

El profesor Ian Somerville presenta una definición acerca de lo que es un requisito: "*Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema.*"(18)

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué debe hacer el sistema para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados. (21)

2.4.1 Requisitos funcionales:

El proceso de definición de los requisitos funcionales es dirigido por el equipo de desarrollo, teniendo en cuenta la información brindada por el cliente. Posteriormente a partir de la información que brinda el cliente se elaboran los documentos de especificación de requisitos. Finalmente se valora y valida la información en busca de errores, inconsistencias o faltas para evitar omitir algún requerimiento del cliente. (22)

2.4.2 Captura de requisitos

Para la obtención de los requisitos que debe tener el sistema se usaron técnicas de captura de requisitos como fueron: entrevista, reuniones y observación. La entrevista se realizó para interpretar las necesidades del equipo de desarrolladores del sistema. Las reuniones con el grupo de trabajo permitieron mediante las tormentas de ideas definir posibles escenarios que enviar notificaciones. Las acciones necesarias para cumplir con las notificaciones en cada escenario conforman los requisitos funcionales del componente. La observación a los desarrolladores y clientes da la oportunidad de entender cómo el usuario interactúa con el software de esta forma se pueden recoger muchas tareas que son sutiles y que no son descritas fácilmente por los desarrolladores.

A partir de la captura de requisitos realizada se definieron los siguientes requisitos funcionales:

Número	Requisito Funcional
RF1	Crear evento
RF2	Crear notificación
RF3	Modificar notificación
RF4	Eliminar notificación
RF5	Listar notificación
RF6	Lanzar notificación
RF7	Listar usuarios
RF8	Filtrar usuarios por roles
RF9	Listar eventos
RF10	Capturar evento
RF11	Listar tipo de notificación
RF12	Crear tipo de notificación
RF13	Modificar tipo de notificación
RF14	Eliminar tipo de notificación
RF15	Listar transporte
RF16	Configurar Email
RF17	Configurar servidor Jabber
RF18	Listar plantilla
RF19	Obtener notificaciones vinculadas a evento
RF20	Listar entidades
RF21	Enviar notificaciones por Jabber
RF22	Enviar notificaciones por Email

Tabla 2.1: Requisitos funcionales.

2.4.3 Requisitos no funcionales.

Los requisitos no funcionales forman una parte significativa de la especificación. Son restricciones de los servicios o funciones ofrecidos por el sistema. Los requisitos no

funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. (27)

Número	Requisito no funcional
Usabilidad	
RNF1	Las interfaces del sistema deben ser entendibles, sencillas y amigables de manera que el usuario se sienta cómodo y pueda acceder a las funcionalidades de una forma rápida y cómoda.
Software	
RNF2	Para un correcto funcionamiento en la pc cliente se recomienda tener Navegador Moxila Firefox 3,6 o superior.
RNF3	Para el servidor es necesario Soporte para PHP 5.4.x Servidor web apache Oracle 11g
Hardware	
RNF4	Prestaciones de la pc cliente Procesador: 2.6 GHZ RAM: 1Gb Tarjeta de Red: 1
RNF5	El servidor debe contar con: Tarjeta de Red: 1, Procesador: 2.6 GHZ, RAM: 2GB, Disco duro: 120 GB

Seguridad	
RNF6	Para interactuar con el componente el usuario debe estar autenticado y tener los permisos necesarios.

Tabla 2.2 Requisitos no funcionales

2.4.4 Especificación de requisitos

La especificación de requisitos establece la base para el acuerdo entre usuarios y desarrolladores de software, quedando definido el comportamiento deseado del producto. (29) Según la metodología de desarrollo seleccionada (AUP) cuando se modela el negocio con modelo de dominio y además se realiza una descripción del proceso se deben especificar los requisitos utilizando descripción de requisitos por procesos (DRP) (19). A continuación, se muestra la descripción del requisito relacionado con el RF2 Crear notificación.
El resto de DRPs se encuentran en los anexos del documento.

Precondiciones	1. *El usuario encargado debe tener los permisos necesarios para acceder a la funcionalidad.* 2. *El usuario debe estar autenticado en el sistema.*	
Flujo de eventos		
Flujo básico adicionar notificación		
1.	*Se muestra un listado con las notificaciones existentes con los siguientes datos: Nombre, Título, Evento, Implicado y Cuerpo. Brinda también las siguientes opciones:* • Adicionar • Editar • Eliminar	
2.	Se selecciona la opción “Adicionar”. En caso de seleccionar la opción “Editar” ver requisito funcional Modificar Notificación	

	En caso de seleccionar la opción “Eliminar” ver requisito funcional Eliminar Notificación	
3.	Se debe mostrar una interfaz con los siguientes campos *Ver Ilustración1:* Nombre Título Tipo de notificación Evento Implicado Cuerpo Entidad Asociada Opción de idioma Permite además las opciones “Aceptar” y “Cancelar”	
4.	Se introducen los datos.	
5.	Se selecciona la opción “Aceptar”. En caso de que los campos requeridos no estén llenos ver flujo alterno *4.a Datos incompletos*. En caso de que los campos estén incorrectos ver flujo alterno *4.b Datos incorrectos*. En caso de que exista la categoría insertada ver flujo alterno *4.c Categoría repetida.* En caso de seleccionar la opción “Cancelar” ver flujo alternativo *4.d Cancelar.*	
6.	Se muestra muestra el listado de notificaciones y muestra el mensaje “Notificación creada con éxito”	
Pos-condiciones		
1.	Se adiciona correctamente la notificación	

Flujos alternativos		
Flujo alternativo 4.a Datos incompletos		
1.	Se muestra el mensaje de error “Este campo es requerido”.	
2.	Vuelve al paso 3 del flujo básico.	
Pos-condiciones		
1.	Se muestra un mensaje de error informando que es obligatorio el llenado de los campos marcados como requeridos.	
Flujo alternativo 4.b Notificación repetida		
1.	Se muestra el mensaje de error “La notificación ya existe en el sistema”.	
2.	Vuelve al paso 3 del flujo básico.	
Pos-condiciones		
1.	Se muestra un mensaje de error informando que ya existe la notificación.	
Flujo alternativo 4.c Cancelar.		
1.	Cancela la acción de Adicionar Notificación.	
2.	Vuelve al paso 1 del flujo básico	
Pos-condiciones		
1.	Cancela la acción de Adicionar Notificación.	
Validaciones		
1.	*Todos los campos son obligatorios*	

Conceptos	**Notificación**	*Visibles en la interfaz:* *Nombre* *Título* *Tipo de notificación* *Evento* *implicado* *Cuerpo* *Idioma* *Utilizados internamente:* *N/A*	
Requisitos especiales	*N/A*		
Asuntos pendientes	*N/A*		

Tabla 2.3 DRP Crear Notificación

2.4.5 Verificación de requisitos

Los requisitos una vez definidos necesitan ser verificados. Esto tiene como misión demostrar que los requisitos definen realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los artefactos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. Aun así, existen algunas técnicas que pueden aplicarse para ello (23):

Una de ellas es la creación de prototipos, que se basa en obtener prototipos que, sin tener la totalidad de la funcionalidad del sistema (24).

La construcción de prototipos ayuda al desarrollado de software y al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos. De esta manera, involucra al cliente más profundamente en el desarrollo del producto.

A continuación, se muestra el prototipo creado para el RF1 Crear notificación y el realizado para el RF5 Lanzar notificación.

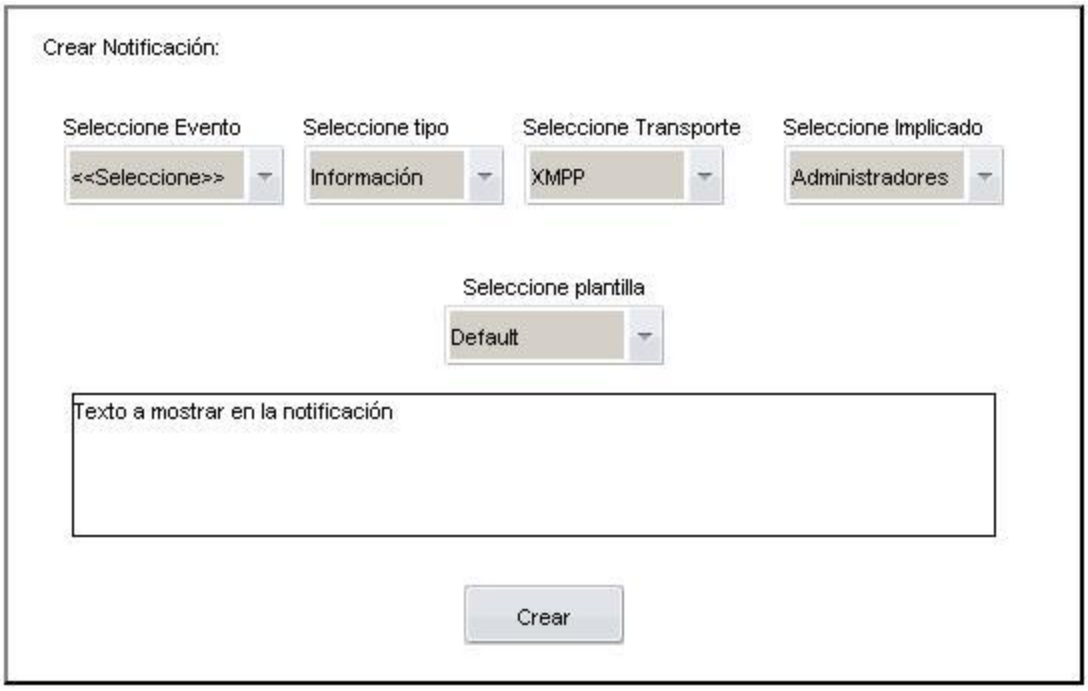


Figura 2.2: prototipo de crear notificación

Figura 2.3: prototipo de lanzar notificación

2.5 Análisis y diseño

Análisis y diseño en una de las disciplinas que se definen para la variación de la metodología empleada.

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura).
Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.

2.5.1 Patrones de diseño

Un patrón de diseño describe una estructura que resuelve un problema particular dentro de un contexto específico y en medio de fuerzas que pueden tener un impacto en la manera en que se aplica y se utiliza. (25)
Symfony2 implementa varios patrones de diseño que se utilizarán en la implementación de la solución propuesta, entre ellos se encuentran los patrones de Asignación de Responsabilidades (GRASP) y patrones de la pandilla de los cuatro (GoF).

Patrones GRASP

Controlador: Este patrón se evidencia en Symfony2, en las clases app_dev.php y app.php. Todas las peticiones son manejadas por un solo controlador frontal, que es el punto de entrada único de toda la aplicación en un entorno determinado.

Alta cohesión: Este patrón se utiliza con el propósito de asignar las responsabilidades a las clases de manera que permanezcan lo más cohesionadas posible, garantizando que ninguna realice un trabajo excesivo gracias a la colaboración con otras clases.

Decorador: este patrón se evidencia en la clase base.tiwg.html; este archivo, conocido también como plantilla global, guarda el código HTML, que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en la base, o si se mira desde el otro punto de vista, la base decora la plantilla.

Inyección de dependencia: Este es un patrón de diseño, en el que se suministran objetos a una clase, en lugar de ser la propia clase quien cree los objetos. Este se utiliza en todas las clases controladoras y se evidencia a la hora de obtener las peticiones en los controladores, al hacer las operaciones de la base de datos con doctrine.

Patrones GoF

Los patrones GoF se pueden clasificar según su propósito:

- De creación: conciernen al proceso de creación de objetos.
- De estructura: tratan la composición de clases y/u objetos.
- De comportamiento: caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos. (25)

El desarrollo del sistema empleando el marco de trabajo Symfony2 proporciona ventajas significativas para los desarrolladores de software. Symfony2 es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones de diseño y arquitectónicos más utilizados en la actualidad. Los patrones de diseño utilizados para esta solución son:

Decorador: aplicado a la generación de vistas, la solución que ofrece dicho patrón es la de añadir funcionalidad adicional a las plantillas. Por ejemplo, añadir el menú y el pie de página a las plantillas que lo requieran, se trata de decorar las plantillas con elementos adicionales reutilizables.

Observador: Define una dependencia de uno-a-muchos entre objetos, de tal forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él (Larman, 2004). Se evidencia en todo lo que se refiere a la notificación de eventos.

2.5.2 Patrones arquitectónicos

El patrón arquitectónico empleado para el desarrollo del trabajo fue el modelo vista controlador. El marco de trabajo Symfony2 está basado en este patrón arquitectónico, que está formado por tres niveles. En el controlador se encuentran las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica de la aplicación. Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación Symfony2, la URL define una acción y los parámetros de la petición. La vista es la encargada de organizar las páginas que son mostradas como resultado de las acciones, donde se encuentra el *Layout*[5], que es común para todas las páginas de la aplicación. La vista en Symfony2 está conformada por varias partes, preparadas cada una de ellas para ser transformables por la persona que trabaja con cada aspecto del diseño de las aplicaciones.

[5] Layout: voz inglesa que define la posición (y su presentación gráfica) de los componentes de un sistema.

En Symfony2, el acceso y la modificación de los datos que se almacenan en la base de datos, se realiza mediante objetos. Doctrine se encarga de esta generación automática para construir sus clases, creando la estructura y generando el código de las mismas. A medida que el desarrollo de un proyecto va avanzando, puede ser necesario agregar métodos y propiedades personalizadas en los objetos del modelo, esto trae consigo que se aumenten las tablas o columnas.

2.5.3Diagrama de paquetes

Una vez definidos los patrones a utilizar en el desarrollo de la solución, se realiza el diagrama de paquetes con el objetivo de delimitar las responsabilidades del negocio y agrupar las funcionalidades en el módulo. Los diagramas de paquetes son utilizados para reflejar la organización de los paquetes y sus elementos, para así proveer una visualización de sus correspondientes nombres de espacio. (32)

A continuación, en la figura 2.4, se muestra el diagrama de paquetes de la solución.

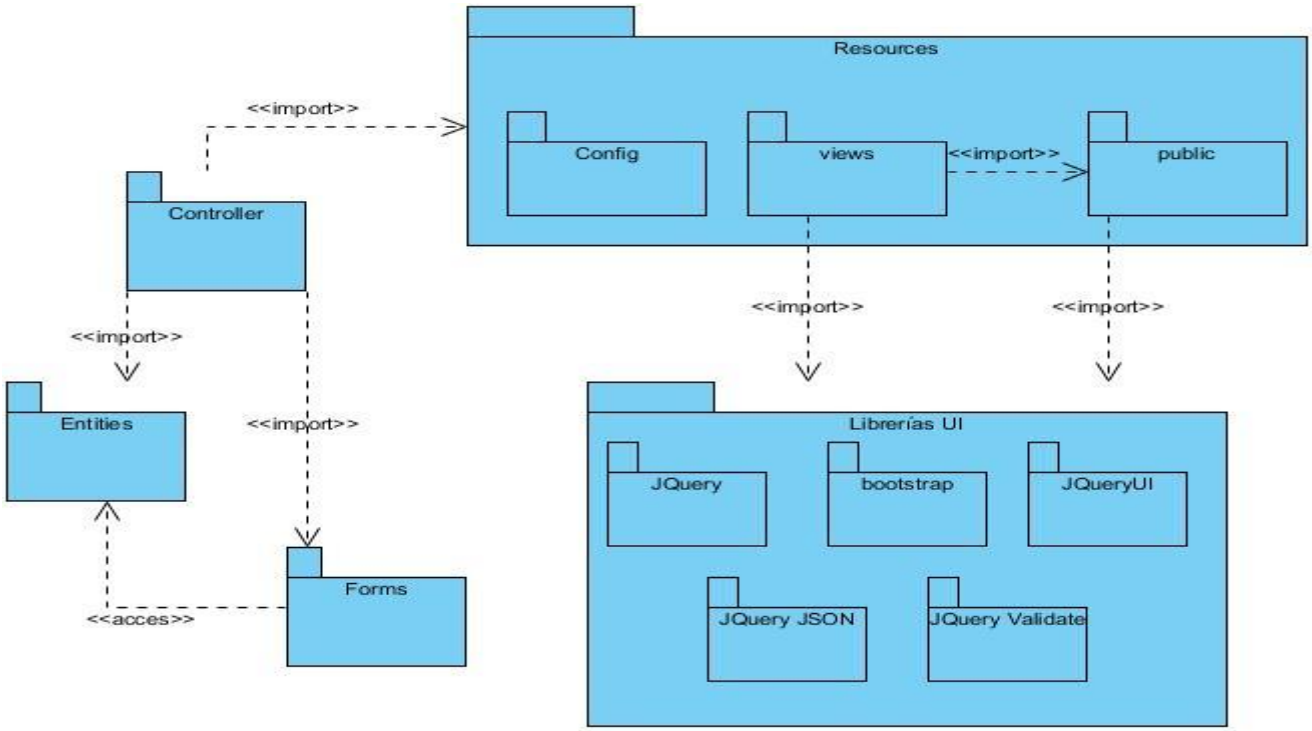


Figura 2.4 Diagrama de paquetes

El diagrama de paquetes (Figura 2.4) está enfocado a la estructura del *bundle*. Está compuesto por los paquetes:

- Entities en el que están agrupadas todas las clases del módulo.
- Formularios que contiene los formularios del sistema.
- Controller integrado por las clases controladoras del componente.
- Resource que agrupa todos los recursos que están asociados al *bundle* como los archivos públicos, las vistas y los de configuración.
- Librerías UI[6] que presenta aquellas librerías que no pertenecen a los componentes visuales del sistema.

2.5.4 Diagrama de clases de diseño

En la figura 2.5 se presenta el diagrama de clases de diseño asociado al requisito Adicionar Notificación. Mediante este diagrama, se puede observar el funcionamiento interno de este requisito en el sistema y se evidencia la aplicación del patrón arquitectónico MVC.

Está compuesto por las vistas del sistema, estas incluyen las páginas cliente Notificaciones y Crear Notificación.

La página servidora app.php y el formulario DatosNotificación. También componen este diagrama la clase controladora de la solución DefaultController y todas las clases pertenecientes al modelo del sistema que utiliza este requisito, representado en el paquete Entidades.

El diagrama de clases del diseño describe la estructura del sistema, mostrando sus clases, asociaciones, atributos, métodos y sus relaciones entre ellos.

En el paquete Entidades hay un total de 10 clases, representando las relaciones entre ellas y los atributos que presentan cada una.

[6] UI: Interfaz de usuario

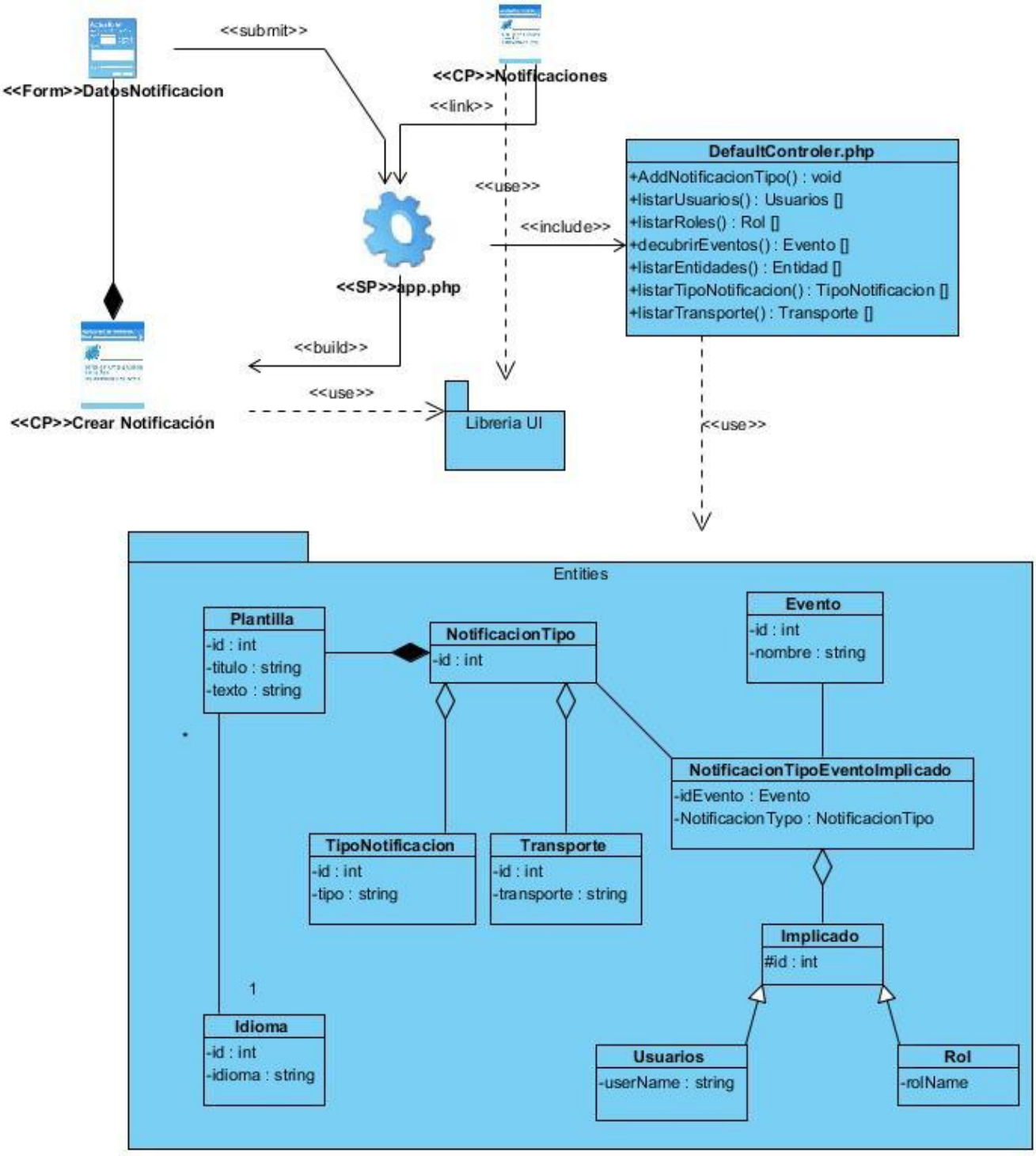


Figura 2.5 Diagrama de clases del diseño asociado al requisito Adicionar Notificación

2.5.5 Métricas para la verificación del diseño del sistema

Dentro del contexto de la ingeniería del software, una medida proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto. El IEEE[7] 17 define métrica como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. (32)

Las métricas suelen calificarse como instrumentos que cuantifican un criterio determinado. Estas son utilizadas para indicar la calidad de un producto de software, evaluar los beneficios en términos de productividad y de calidad, así como para justificar el uso de determinadas herramientas, entre otras. Lorenz y Kidd en su libro dividen las métricas basadas en clases en cuatro categorías, cada una con un diseño al nivel de componentes: (29)

- Tamaño: se centran en el conteo de atributos y de operaciones para una clase individual y promedian los valores para el sistema OO en su totalidad.
- Herencia: se centran en la forma en que se reutilizan las operaciones a lo largo y ancho de la jerarquía de clases.
- Valores internos: buscan cohesión y asuntos relacionados con el código.
- Valores externos: examinan el acoplamiento y la reutilización.

Las métricas concebidas como instrumento para evaluar la calidad del diseño y la relación con los atributos de calidad fueron:

- Tamaño Operacional de clases (TOC).
- Relaciones entre clases (RC).

Tamaño Operacional de clase (TOC)

El tamaño general de una clase puede medirse determinando las siguientes medidas: (29)

- El total de operaciones (tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- El número de atributos (tanto heredados como privados de la instancia), encapsulados por la clase.

[7] IEEE: Instituto de Ingeniería Eléctrica y Electrónica: . asociación mundial de ingenieros dedicada a la estandarización y el desarrollo en áreas técnicas.

Valores grandes de TOC representan gran responsabilidad de la clase. Esto implica la reducción de la reutilización de la clase y complica la implementación y las pruebas. De forma general, operaciones y atributos deben ser ponderados al determinar el tamaño de la clase. Para valores pequeños de TOC para una clase existe mayor posibilidad de que la clase pueda ser reutilizada.

Parámetros de Calidad	**Valores Grandes de TOC**
Reutilización	Reduce la reutilización de la clase
Implementación	Complica la implementación
Responsabilidad	La clase debe tener bastante responsabilidad

Tabla 2.4 Métrica Tamaño operacional de las clases

Medidas y umbrales aplicados:

Parámetros	**Criterio**	**Umbral**
Responsabilidad	Pequeño	< =Prom.
	Medio	Entre Prom. y 2* Prom.
	Grande	> 2* Prom.
Complejidad de implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Prom
	Alta	<= Prom.

Tabla 2.5: Umbrales utilizados en TOC

Se les aplicó la métrica de TOC a un total de 14 clases para un total de 52 atributos, obteniéndose un promedio de atributos de 3.72 y a un total de 32 operaciones con un promedio de operaciones de 2.29. De las clases analizadas se tienen que, de acuerdo con la responsabilidad, 10 tienen responsabilidad baja, 4 media y 0 alta; respecto a la complejidad 10 clases la tienen baja, 4 media y 0 alta; y referente a la reutilización 0 la presentan baja, 4 media y 10 alta. Este resultado puede verse reflejado en el gráfico de barra presentado a continuación (ver figura. 2.6).

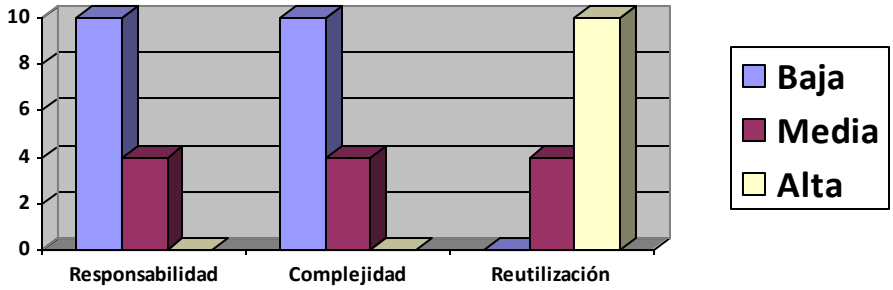


Figura 2.6 Resultado de la métrica TOC

A partir de estos resultados, se concluye que la complejidad y la responsabilidad del diseño realizado son generalmente bajas; no se presentarán problemas con la implementación y las clases dentro del sistema se podrán reutilizar.

Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otras. Los atributos de calidad que esta afectan son Acoplamiento, Complejidad de mantenimiento, Cantidad de Pruebas y Reutilización.

Parámetros de Calidad	**Valores grandes de RC**
Acoplamiento	Implica un aumento del acoplamiento de la clase
Complejidad de mantenimiento	Implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Implica una disminución en el grado de reutilización de la clase.
Cantidad de Pruebas	Implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 2.6: Métrica Relaciones entre clases

Medidas y umbrales aplicados:

Parámetros	**Criterio**	**Umbral**

Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	> 2*Prom.
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom
	Alta	<= Prom
Cantidad de Pruebas	Baja	<= Prom
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla 2.7: Umbrales utilizados en RC

Una vez aplicada la métrica RC (ver Anexo 7), se obtiene un total de 17 relaciones de uso para 14 clases, siendo un promedio de 0.82 relaciones. Con estas relaciones se determina que el acoplamiento es generalmente bajo en las clases, al igual que la complejidad de mantenimiento y la cantidad de pruebas, además de que la reutilización de las clases es alta. Este resultado puede verse reflejado en el gráfico de barra presentado a continuación (ver fig. 2.7).

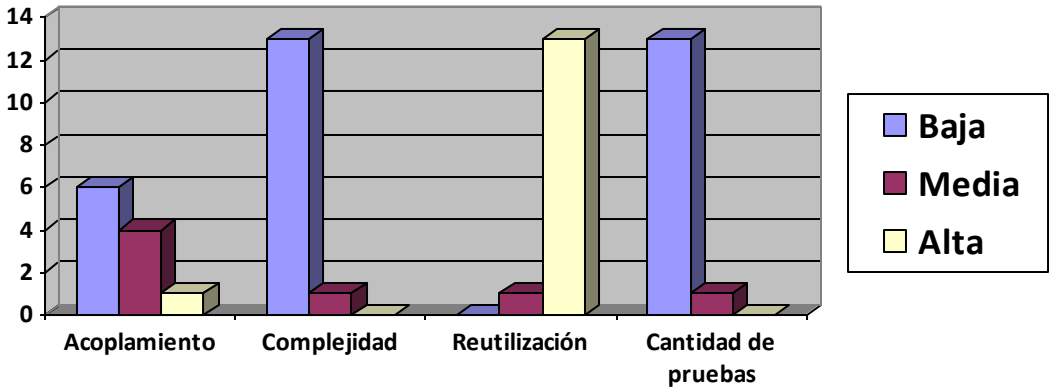


Figura 2.7 Resultado de la métrica RC

2.5.6 Diagrama de base de datos

El modelo de base de datos propuesto cuenta con 14 tablas.

La tabla Notificación integra los datos de las tablas transporte que almacena el medio por el que será enviada la notificación.

La tabla plantilla que contiene el texto o mensaje predefinido para la notificación y el tipo de notificación que se envía. Se relaciona con la tabla evento y con la tabla Implicado a través de la tabla Evento_Notificación y es donde se guarda el envío realizado de notificación.

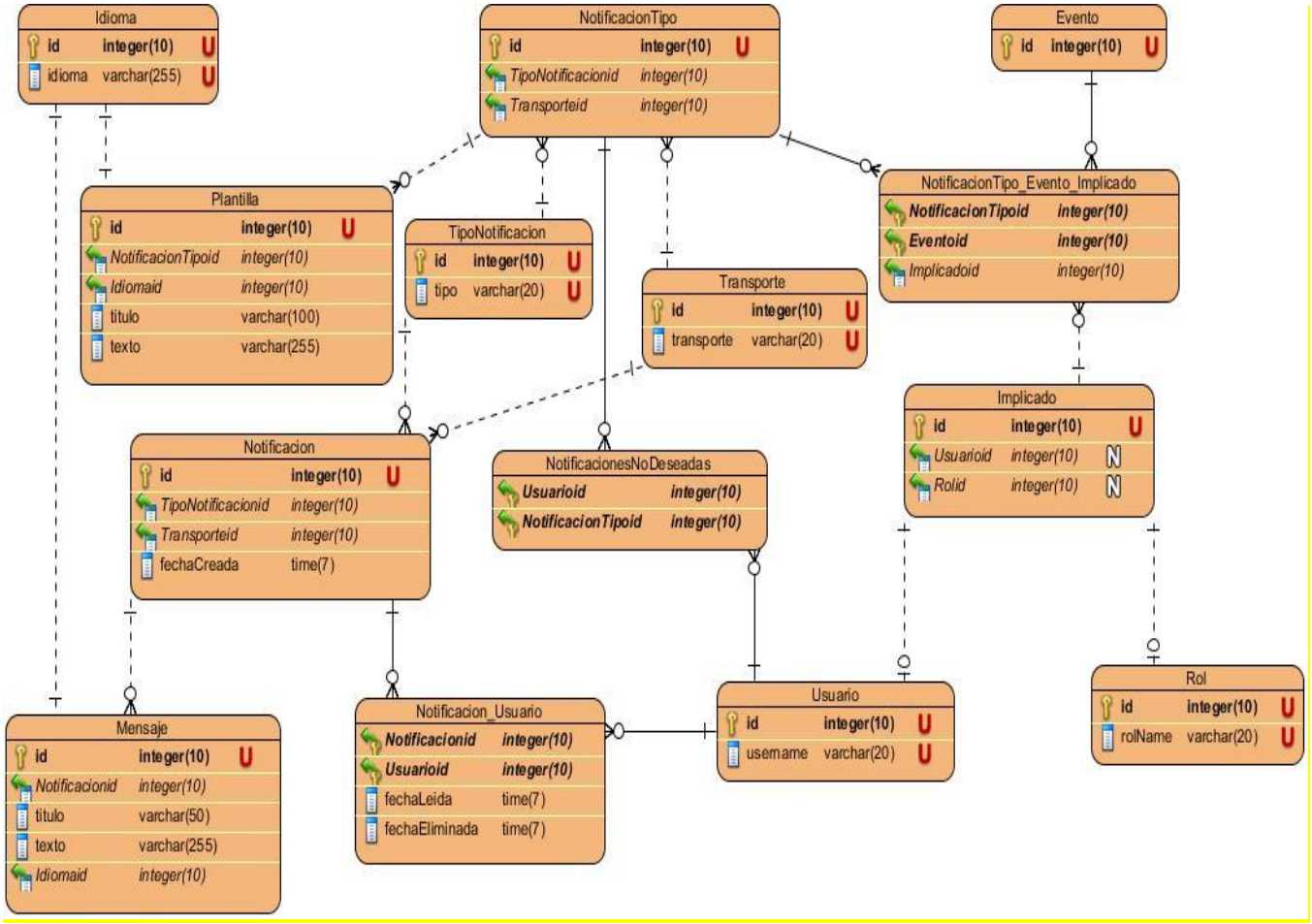


Figura 2.6 Diagrama entidad relación del componente notificaciones.

2.6 Conclusiones del capítulo

Con la captura de requisitos, análisis y diseño de la solución se obtuvieron 22 requisitos funcionales para el sistema modelado y se generaron los artefactos definidos por el proyecto Ventanilla Única Aduanera, obteniéndose entre ellos el diagrama de clases de diseño, con un total de 14 entidades. Además, se validó el diseño utilizando las métricas Tamaño Operacional de Clases y Relaciones entre Clases. La validación del diseño, los artefactos obtenidos, unido al uso de los patrones de diseño, permitieron sentar las bases para la implementación del software con un alto grado de calidad.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

En el presente capítulo se abordan los temas referentes a la implementación de la solución anteriormente analizada y modelada. Se muestran algunos de los artefactos generados en esta etapa, además de los estándares de codificación utilizados en la construcción de la solución se detalla el trabajo acerca del tratamiento de errores y se plasma el diagrama de componentes. Se presentan además las pruebas que se aplican para verificar que el comportamiento observado del software cumple con los requisitos establecidos en etapas anteriores.

3.2 Implementación

La implementación es una de las etapas fundamentales en el desarrollo de un software. En ella se construyen los componentes, su organización y dependencia entre nodos físicos en los que funcionará la aplicación. (33) En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. (33) Durante el desarrollo de esta etapa se planifican las integraciones necesarias del sistema, se distribuyen en componentes, se implementan las clases y vistas definidas en la fase del diseño y se prueban los componentes individualmente.

A partir de los resultados y artefactos obtenidos en el diseño, se realiza un análisis de los diagramas de clases para conocer las interrelaciones y responsabilidades de las mismas. Luego de este estudio se desarrolla cada una de las funcionalidades descritas.

3.2.1 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. (32) Un código fuente completo debe reflejar un estilo armonioso de la implementación de un sistema.

Como estándar de codificación para la implementación de la solución se utilizó el definido en el documento Propuesta de un estándar de codificación: Ventanilla Única Dpto. Aduana

CEIGE. En el mismo se muestran los diferentes estándares a seguir tanto en la implementación de las pantallas como en las clases definidas para las pruebas. A continuación, se exponen algunos de los principales elementos presentes en dicho documento.

Nombre de las clases:

- Los nombres de las clases deben estar expresados en notación *UpperCamelCase* .
- No se deben utilizar guiones bajos en su nombre "_".
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- Los nombres de las clases no deben estar atados a las clases de las que se derivan, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.

> **Ejemplo:**
>
> DatosGenerales, OmiDeclaracionGeneral, etc.

Nombre de las funciones

- Todas las funciones definidas por los desarrolladores deben seguir la nomenclatura CamelCase, a no ser que para cierto ámbito se especifiquen características específicas.
- Los nombres de las funciones deben dejar reflejado claramente cuál es la acción que realiza el mismo.

> **Ejemplo:**
>
> manejarErrorBD, descagarArchivoFTP, etc.

- Se debe apoyar en los prefijos para expresar la acción que realiza sobre un elemento determinado:

> **Ejemplo:**
>
> Set: para cambiar el valor de una variable.
>
> Get: para devolver un valor determinado.
>
> Is: devuelve un valor booleano de si un elemento cumple con una condición dada.

Variables

- Los nombres de las variables deben expresar claramente el contenido de las mismas.
- Pueden estar referidas en singular o plural.
- Se definen al principio de las estructuras donde son utilizadas.

Ejemplo:

Archivo, clase, función, IF, SWITCH, WHILE, etc.

- En caso de que no se le asigne un valor inicial se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.
    1. Los tipos de datos cadena son definidos con comillas dobles (“);
    2. Los tipos de datos de caracteres se definen con comillas simples (‘);
    3. En caso de que se espere almacenar tipos de datos diversos no se inicializa.

Ejemplo:

```
$numeroBajas = 0;
$nombreAcotado = “”;
$arreglo_temp = array();
$objeto = new Clase;
$mixta;
```

Política de llaves

- Política tradicional de Unix de colocar la primera llave en la misma línea del comienzo del bloque separada por un espacio y la llave de cierre del bloque en la última línea.

- La estrategia debe ser mantenida en todo el código escrito por el programador y siempre colocar las llaves aunque sea para una sola línea de código dentro del bloque.

Ejemplo:

```
if ($condition) {
...
}
```

```
while ($condition) {
...
}
```

Políticas de () paréntesis

- No utilizar paréntesis seguido de palabras claves del lenguaje, se debe dejar un espacio por el medio.
- Los paréntesis se sitúan seguidos de los nombres de las funciones.
- No se debe utilizar paréntesis en las sentencias return de no ser necesario.

Ejemplo:

```
if (condition) {
}
strcmp($s, $s1);
return 1;
```

```
while (condition) {
}
```

Políticas de Sangría, Tabulación y Espacios.

- La sangría de cada nivel será de 4 espacios o un tabulador.
- Para el caso de los ficheros YAML solo se debe usar la sangría con espacios.

3.2.2 Tratamiento de errores

En el desarrollo de software el tratamiento de errores es muy importante para garantizar el correcto funcionamiento del sistema que no afecte la calidad del mismo. Es fundamental identificar y controlar los posibles problemas que puedan presentarse a la hora de interactuar con el software.

Para lograr lo mencionado anteriormente se establecieron mecanismos de validación que comprueban la corrección de los datos a tratar tanto en el lado del cliente como en el lado del servidor. Uno de estos son las validaciones de formulario y objetos haciendo uso de los métodos isValid() y validate() (figura 3.1). Estos hacen uso de los Assert en las entidades para validar individualmente cada uno de los atributos. Aplicando todo esto, se evitan las inyecciones de código SQL y garantiza la validación de los datos que se introducirán al sistema.

```
$notificacionTipo->addImplicado($implicado);
$notificacionTipo->assignarIdNotificacionTipoAPlantillas();
if($form->isValid()){
    $em->persist($notificacionTipo);
    $em->flush();
    $rpta['success'] = true;
    $rpta['data']['textResponse'] = $translator->trans('notificacion_tipo.add.msg');
}else{
    $errors = $this->get('validator')->validate($notificacionTipo);
    $msg = $translator->trans('notificacion_tipo.eliminar.failure.title').':<br><ul>';
    if(count($errors) !=0 ){
        foreach ($errors as $e){
            $msg .= "<li>" . $e->getMessage() . "</li>";
        }
        $msg .= '</ul>';
    }
    $rpta['data']['errors'] = $msg;
}
return new JsonResponse($rpta);
```

Figura 3.1: Fragmento de validación de adicionar notificación

Además, se insistió en que el usuario introduzca la menor cantidad posible de datos, evitando así incoherencias e incorrecciones en los mismos. En el caso de la entrada de datos por parte del usuario se implementaron funciones que validaron dicha entrada para que, de existir errores, se muestren mensajes de alerta indicando los errores detectados.

3.2.3 Manejo de idiomas en el *bundle*

El manejo de idiomas es el conjunto de acciones encaminadas a traducir y adaptar el sitio web a diferentes idiomas sin necesidad de cambiar el código. (33) Este proceso se realiza fundamentalmente para asegurar que el producto sea funcional y que sea aceptado en los mercados internacionales.

El uso de los distintos idiomas trae un conjunto de ventajas, como son:

- Aumentar el mercado.
- No es necesario hacer un desarrollo internacional del producto una vez acabada la primera versión.
- Los recursos se utilizan más eficientemente.

En el proceso de traducción del software se incluyen un conjunto de elementos como los textos, botones, títulos, mensajes, menús y diálogos.

En Symfony2 las traducciones se gestionan mediante catálogos, que son archivos de texto en formato XLIFF, PHP o YAML. Estos archivos son los que contienen las traducciones a cada idioma de las diferentes cadenas de texto de las plantillas. En el caso del *bundle* Notificaciones, se utiliza el formato YAML (ver fig. 3.2).

Figura. 3.2: Ubicación del archivo para la internacionalización

3.2.4 Diagrama de componentes

Un diagrama de componentes muestra dependencias entre dichos elementos. Cada componente ofrece algunas interfaces y utiliza otras. Si las dependencias entre componentes se hacen a través de interfaces, los componentes se pueden sustituir por otros componentes que realicen las mismas interfaces. (28) Con ellos se estructura el modelo de implementación y las relaciones entre los elementos a implementar.

El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando:

- Los subsistemas de implementación y sus dependencias a la hora de importar código.
- Organizar los subsistemas de implementación en capas.

A continuación, en la figura 3.3 se puede apreciar el diagrama de componentes de la solución a desarrollar. En el mismo se muestra la relación entre el componente Notificaciones y el resto de los componentes que utiliza el sistema, además de aquellos que provee Symfony2, como **Translator** para la internacionalización, **Validator** para la validación de las entidades, **Forms** para la creación de formulario, EventManager para la captura de eventos y **Twig** para la creación de las plantillas.

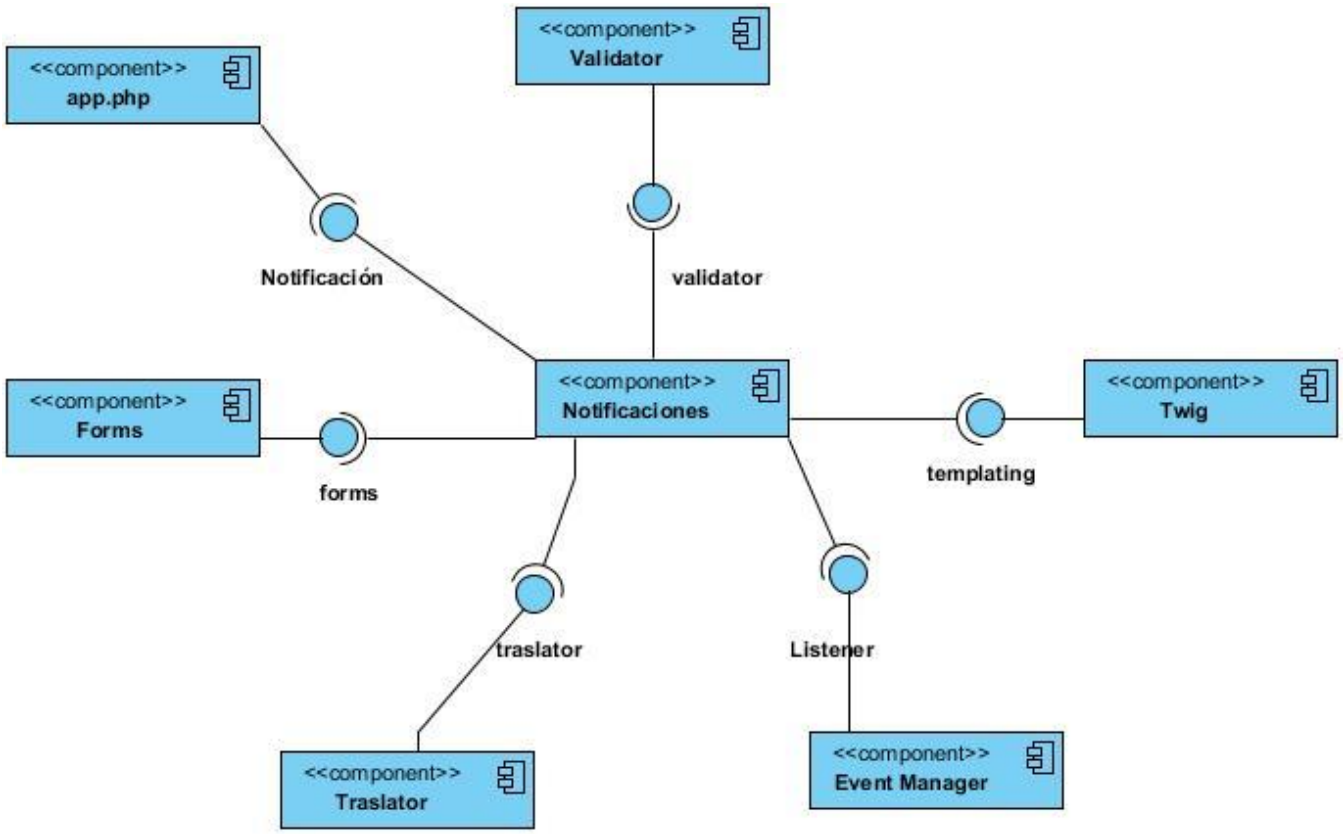


Figura 3.3 Diagrama de componentes.

3.3 Pruebas aplicadas

Las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. (35) Una vez terminada la implementación del sistema, es necesario probar el software con el objetivo de identificar y corregir el máximo de errores posibles antes de ser entregados al cliente.

3.3.1 Pruebas de caja negra

Estas pruebas pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Se centran en los requisitos funcionales del software y permiten encontrar errores de tipo de funciones incorrectas o ausentes, de interfaz, en la estructura de datos, de rendimiento y de inicialización y terminación. (34)

Para la realización de este método se utilizó la técnica de partición equivalente, la cual divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. (34)

A continuación se muestra el diseño de un caso de prueba para el requisito adicionar notificación.

SC Adicionar notificación

Escenario	Descripción	Tipo	Transporte	Evento	Usuarios	Roles	Título	Nombre	Mensaje	Respuesta del sistema
EC 1.1 Adicionar notificación	*Se adiciona correctamente la notificación*	V *INFORMACIÓN*	V *XMPP*	V *Evento completo*	V *federico*	V *ROLE_USER*	V *Nueva*	V *Nueva*	V *Se ha creado un*	*El sistema muestra el cartel notificación creada satisfactoriamente y*
EC 1.2: Datos incompletos	Se muestra un mensaje indicando que no se han introducido los datos obligatorios.	I Vacio	V Email	V Evento completo	V Federico	V ROLE_USER	V error en el sistema	V error en el sistema	V Ha ocurrido un error en el sistema x	Muestra un mensaje indicando que faltan datos por introducir y señala los datos que faltan por introducir.
		V ERROR	I Vacio	V	V Federico	V ROLE_USER	V error en el sistema	V error en el sistema	V Ha ocurrido un error en el sistema x	
		V ERROR	V EMAIL	I Vacio	V Federico	V ROLE_USER	V error en el sistema	V error en el sistema	V Ha ocurrido un error en el sistema x	
		V ERROR	V EMAIL	V Evento completo	I Vacio	I Vacio	V error en el sistema	V error en el sistema	V Ha ocurrido un error en el sistema x	
		V ERROR	V EMAIL	V Evento completo	V Federico	V ROLE_USER	I Vacio	V error en el sistema	V Ha ocurrido un error en el sistema x	
		V ERROR	V EMAIL	V Evento completo	V federico	V ROLE_USER	V error en el sistema	I Vacio	V Ha ocurrido un error en el sistema x	
		V ERROR	V EMAIL	V Evento completo	V federico	V ROLE_USER	V error en el sistema	V error en el sistema	I Vacio	
EC 1.4: Cancelar	Cancela la acción de Adicionar notificaciones.	NA	NA	NA	NA	NA	NA	NA	NA	Cancela la acción de Adicionar notificación y retorna a la interfaz Configurar Notificaciones

Figura 3.4. Diseño de caso de prueba adicionar notificación

En la fase de pruebas se realizaron tres iteraciones al componente de gestión de notificaciones, en la primera se detectaron cuatro no conformidades. Se realizó posteriormente una segunda iteración de pruebas para verificar que se habían eliminado los problemas encontrados en estos requisitos o si se encontraban nuevas no conformidades en el funcionamiento del módulo. Esta tercera iteración concluyó satisfactoriamente ya que no se encontró ningún otro problema en el componente de gestión de notificaciones de eventos.

3.3.2 Pruebas Aceptación

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada 'prueba de aceptación'. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. (18)

Las pruebas realizadas se aplicaron al concluir el desarrollo de todas las funcionalidades. Para la realización de las pruebas, se colocó a los usuarios finales en su ambiente de trabajo y a medida que fueron utilizando el sistema estos fueron registrando todos los problemas que detectaron.

En la primera iteración fueron detectadas tres no conformidades (NC), en la segunda iteración se corrigieron las NC encontradas en la primera y se detectaron dos NC que fueron corregidas en la tercera iteración.

Finalmente, el cliente expresó su satisfacción por la calidad de la solución brindada a través de una carta de aceptación del producto.

3.4 Validación de la solución

3.4.1 Panel de expertos

Se entiende por experto a un especialista en una materia. Persona experimentada, que posee una gran experiencia o habilidad en una actividad Para la validación y aceptación del

componente para la gestión de notificaciones en el sistema Ventanilla Única Aduanera, se utilizó el criterio de un Grupo de Expertos.

Este panel se conformó con especialistas con experiencia trabajando en el sistema Ventanilla Única Aduanera y en los temas relacionados con la configuración en los sistemas web.

En esta técnica se realiza una selección del grupo de expertos que participará en el proceso de evaluación, teniendo en cuenta que ningún experto conoce las respuestas individuales de los otros que componen el panel. Esto posibilita, que un miembro pueda cambiar sus opiniones y defender sus argumentos con la tranquilidad desde su propio punto de vista, sin ser influenciado por el prestigio o el conocimiento de los demás panelistas.

La correcta elección de los expertos propicia obtener resultados con calidad y una opinión grupal con un alto grado de aceptación. A continuación, se hará la descripción de los pasos utilizados en la selección del Panel de Expertos y los resultados obtenidos.

Proceso de selección de los expertos

Los expertos deben ser capaces de ofrecer valoraciones conclusivas de un problema en cuestión y hacer recomendaciones al respecto.

En el desarrollo de este proceso se consideraron tres etapas cruciales:

Confeccionar el listado de expertos.

El listado se realizó atendiendo a la posibilidad real de participación, a través de un correo de confirmación. Todos ellos trabajan directamente sobre el sistema VUA. Poseen amplios conocimientos en temas relacionados con el componente a evaluar que deben ser:

1. Conocimientos del Sistema Ventanilla Única Aduanera.
2. Trabajo con el marco de trabajo Symfony2.
3. Conocimientos de los lenguajes de programación utilizados en el proyecto Ventanilla Única Aduanera del centro CEIGE.

Se utilizaron 5 expertos para la validación de la propuesta debido a que es el número de desarrolladores del proyecto VUA que cumplen con las características antes establecidas.

Para lograr la familiarización de los expertos con el componente de notificaciones se les presentó un caso de estudio en el que bebían crear notificaciones en los escenarios que se realizan actualmente y en los nuevos escenarios donde se necesita enviar notificaciones, una vez utilizando el componente de notificaciones y otra utilizando el sistema tal y como esta.

Para resolver el caso de estudio los expertos debieron realizar las siguientes tareas:

Sin el componente de notificaciones:

1. Seleccionar a que clase controladora correspondía la tarea de lanzar el evento que desencadenara la notificación.
2. Definir nombre del evento y texto de la notificación en la clase controladora seleccionada.
3. Subscribir al evento a un escucha (listener) del sistema.
4. Crear en la vista el lugar en el que se mostrara la notificación.

Todo este proceso se realiza directo en el código de la aplicación utilizando el IDE de la preferencia de los expertos.

Con el componente de notificaciones:

1. Acceder a la página de configuración de notificaciones.
2. Seleccionar la opción añadir notificación.
3. Definir mediante listas de selección y áreas de texto los parámetros de la notificación.
4. Aceptar.

El tiempo de realización del caso de estudio se recogió quedando que para el desarrollo del caso de estudio los 5 expertos emplearon como promedio para crear notificaciones sin el componente 4 horas y utilizando el componente 10 minutos.

Elaboración del cuestionario.

Para recoger las impresiones y propuestas de los expertos se creó un cuestionario

Para la elaboración del cuestionario se tuvieron en cuenta los objetivos que debería cumplir el componente propuesto para su implantación en el sistema VUA.

La encuesta establece una serie de preguntas de enfoque investigativo, que permitió evaluar la posibilidad real de utilización de la propuesta. Contando con una serie de temáticas y una escala que se propuso del uno al diez, siendo uno el de menor escala y diez el de mayor.

Se le facilitó al panel la posibilidad de modificar aspectos que ellos consideraban necesario cambiar y presentar su opinión general a favor o en contra del procedimiento propuesto, con la libertad de expresar todo lo que se pudo agregar al cuestionario.

Todos los expertos recibieron a través del correo electrónico un informe con la propuesta a evaluar, el cuestionario y un plazo de tiempo determinado, para enviar sus respuestas o realizar las preguntas pertinentes, que les surgieran al estudiar el documento.

Cuestionario para los expertos.

1. ¿Cree usted que el componente propuesto aumenta la configurabilidad de las notificaciones del sistema Ventanilla Única Aduanera?

   _Si _No ¿Por qué?

2. ¿Cree usted que se aumente la adaptabilidad del sistema frente a nuevos escenarios que requieran el uso de notificaciones?

   _Si _No ¿Por qué?

3. ¿Cree usted que la implantación del componente de gestión de notificaciones del sistema VUA es beneficiosa para los usuarios del sistema?

   _Si _No ¿Por qué?

   Si cree preciso eliminar o agregar alguno menciónelo y explique brevemente.

4. En una escala de 1 al 5 otorgue una evaluación al componente propuesto de acuerdo a los criterios siguientes.

   _Nivel de configurabilidad.

   _Posibilidades reales de aplicación.

_Adaptabilidad ante nuevos escenarios.

_Repercusión en el sistema Ventanilla Única Aduanera.

5. ¿Cuáles serían los argumentos que usted expondría a favor y en contra del componente de notificaciones?
6. Realice alguna observación o aporte sobre la propuesta.

2.1. Resultado de la evaluación.

Para la obtención del resultado de las encuestas realizadas a los expertos se calculó el coeficiente de concordancia entre las respuestas de los expertos a través de la fórmula siguiente:

$$Cc = \left(1 - Vn/Vt\right) \times 100$$

Donde:

Cc: Coeficiente de concordancia expresado en porcentaje.

Vn: Cantidad de expertos en contra del criterio predominante.

Vt: Cantidad total de expertos.

Si resulta $Cc \geq 60$ % se considera aceptable la concordancia. (Medina, 2007)

Después de calculado el coeficiente de concordancia de los expertos **Anexo II** quedaron con un valor aceptable (≥ 60 %) las siguientes:

- El componente propuesto aumenta la configurabilidad de las notificaciones del sistema Ventanilla Única Aduanera, de 5 expertos que validaron la propuesta 5 estuvieron de acuerdo con esta competencia, obteniendo un valor de concordancia de 100%.
- Se aumente la adaptabilidad del sistema frente a nuevos escenarios que requieran el uso de notificaciones, de 5 expertos que validaron la propuesta 4 estuvieron de acuerdo con esta competencia, obteniendo un valor de concordancia de 80%.
- La implantación del componente de gestión de notificaciones del sistema VUA es beneficiosa para los usuarios del sistema, de 5 expertos que validaron la propuesta 5

estuvieron de acuerdo con esta competencia, obteniendo un valor de concordancia de 100%.

- El componente tiene posibilidades reales de aplicación, de 5 expertos que validaron la propuesta 4 estuvieron de acuerdo con esta competencia, obteniendo un valor de concordancia de 80%.
- Puede repercutir positivamente en el sistema Ventanilla Única Aduanera., de 5 expertos que validaron la propuesta 4 estuvieron de acuerdo con esta competencia, obteniendo un valor de concordancia de 80%.

Además de los elementos mostrados anteriormente los expertos hicieron una serie de valoraciones en contra de la aplicación del componente que se enuncian a continuación.

En contra:

- La forma de insertar las etiquetas en la plantilla puede ser difícil para usuarios inexpertos.
- Para que se logre mayor efectividad en la internacionalización del componente se debe entrar el texto de la notificación en todos los idiomas que maneja el sistema.

Determinar si la muestra de los expertos es significativa.

Conocer si la muestra de expertos con la que se trabajó es significativa para la validación del procedimiento propuesto es de gran importancia, pues confirma la correcta elaboración del mismo.

Para llevar a cabo este paso se tomaron las competencias con nivel de concordancia (*Cc*) mayor de 60%.

Validándose a través del cálculo del coeficiente W de Kendall:

$$W = \frac{S}{\frac{1}{12} K^2 (N^3 - N)} \qquad \text{Dónde} \quad S = \sum \left(R_j - \frac{\sum R}{N} \right)^2$$

Donde:

N: Número de competencias.

K: Número de expertos.

Rj: Se suman todos los R de cada fila, luego se suma la columna que se forma con los nuevos valores.

R: Los valores de cada competencia, es decir, la celda que se forma entre la competencia y cada experto. (Medina, 2007)

Si se obtiene un valor W ≥ 0.65, se concluye que la muestra de los expertos es satisfactoria para la validación de la propuesta.

Luego del cálculo del coeficiente se obtiene un valor de 0.66, por lo que se considera que la muestra es significativa.

37.4 **Conclusiones del capítulo**

En el presente capítulo se generaron los artefactos pertenecientes a la etapa de implementación, como el diagrama de componentes. Se aplicaron los estándares de codificación definidos en el proyecto Ventanilla Única para la implementación de la solución anteriormente modelada, con el objetivo de facilitar el entendimiento del mismo por el equipo de desarrollo del proyecto. También se definió cómo realizar el tratamiento de errores en la solución y el uso de la internacionalización en la misma.

Se aplicaron las pruebas funcionales permitiendo la obtención de no conformidades en la solución, dando la posibilidad de realizarle correcciones al sistema a tiempo y obtener un módulo que responda completamente a los requisitos funcionales. Se comprobó a partir de las pruebas realizadas que la solución desarrollada contribuye a disminuir en tiempo y la veracidad de los datos.

El proceso de validación por parte de los expertos y analizando los resultados que arrojaron sus respuestas en los cuestionarios que les fueron aplicados. Se concluye que el componente es consistente y aplicable al sistema VUA, por lo que se propone su aplicación en la línea productiva.

CONCLUSIONES

La concepción y desarrollo del componente para la gestión de notificaciones del sistema Ventanilla Única Aduanera facilitó arribar a las siguientes conclusiones

- Mediante el estudio de sistemas se logró establecer el marco teórico de la investigación que permitió identificar las posibles funcionalidades que pueden integrar la solución.
- A partir del análisis y diseño se obtuvieron los requisitos funcionales del componente que fueron correctamente validados.
- A partir de la implementación realizada se obtuvo un componente que permite gestionar y configurar las notificaciones del sistema Ventanilla Única Aduanera.
- Mediante la realización de pruebas funcionales y de aceptación se demostró que la solución proporciona un soporte confiable para el almacenamiento y la integridad de la información que se genera durante la ejecución de los procesos de configuración y envío de las notificaciones.

RECOMENDACIONES

Se recomienda que:

- Futuros proyectos que necesiten un sistema de notificaciones automáticas y configurables tengan en cuenta la solución desarrollada en este trabajo, ya que la misma es genérica y necesita solo pequeños cambios en los sistemas para integrarse.
- Se adicione una funcionalidad que permita a un usuario recibir un reporte temporal de lo que ocurre con una entidad específica del sistema.

BIBLIOGRAFÍA

1. Real Academia de la Lengua Española. *Real Academia de la Lengua Española.* [En línea] Real Academia de la Lengua Española. [Citado el: 15 de 10 de 2016.] http://rae.es..

2. **Evento seguridad 2012.** Notificación de incidentes. 2012.

3. **Banners.** *Cultura de Seguridad.* 2013.

4. Notificaciones Facebook en tu escritorio. *WEBADICTOS.* [En línea] [Citado el: 17 de 10 de 2015.] http://www.webadictos.com.mx..

5. Sistema de Bibliotecas de la Universidad de Concepción. [En línea]

6. **tuentimonitor., Equipo de desarrollo de.** Tuentimonitor. [En línea] [Citado el: 15 de 10 de 2015.] http://www.tuentimonitor.com/.

7. **Werdmuller, Ben.** *Build a web-based notification tool with XMPP.* 2010.

8. **Barros, Laura.** *Programación Concurrente y Distribuida.* 2012.

9. **Oviedo Chaparro, Luis Enrique.** *COMET: UN SIGUIENTE PASO AL AJAX MOVIENDO DE LAS APLICACIONES WEB TRADICIONALES A UN NUEVO ESTILO. S.l.: .* Universidad Católica de Asunción. : Facultad de Ciencias y Tecnología, , 2014.

10. **jabber.org.** Mensajeria Instantanea Libre . [En línea] marzo de 2013. [Citado el: 23 de 10 de 2015.] http://www.jabberes.org/glosario.

11. **www.fing.edu.uy.** [En línea] [Citado el: 2015 de 11 de 3.] http://www.fing.edu.uy/~asabigue/prgrado/2004eofgl/contenido/archivos/Anexo-III.pdf..

12. **sagt.cnti.gob.ve .** [En línea] [Citado el: 2016 de 3 de 27.] http://sagt.cnti.gob.ve/otrs/public.pl.

13. *Manual de PHP.* La Habana : s.n., 2011. S.n.

14. Manuales. [En línea] [Citado el: 23 de 10 de 2015.] http://max-alva.webs.com/javascript.htm..

15. **Visual Parading.** UML tool for software application development. *Visual Paradigm for UML.* [En línea] [Citado el: 2015 de 10 de 23.] http://www.visual-paradigm.com/product/vpuml/..

16. Symfony en pocas palabras. [En línea] [Citado el: 25 de 11 de 2015.] http://librosweb.es/libro/symfony_2/capitulo_1/symfony_en_palabras.html.

17. *Ecured.* [En línea] [Citado el: 12 de 10 de 2015.] http://www.ecured.cu/index.php/Modelo_de_dominio.

18. **Sommerville.** *Ingeniería de Software.* S.l. : Pearson Educación, septima edición 2005.

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

20. Jquery para JavaScript. [En línea] [Citado el: 25 de 10 de 2015.] https://jquery.com/ .

21. **Sawyer.** *Requirements Engineering: A good practice guide.* 2010.

22. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de software.* 2000.

23. **María José Escalona, Nora Koch. .** *Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo. .* Sevilla : s.n., 2002. S.n.

24. **Tedeschi, Nicolás.** ¿Qué es un Patrón de Diseño? . *Microsoft MSDN.* [En línea] [Citado el: 13 de 3 de 2016.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

25. **Alexander, Christopher.** *The Timeless Way of Building.* 1979.

26. **Mendoza, Yusniel Cedeño.** *Trabajo de diploma Componente para la gestión de notificaciones en el marco de trabajo Sauxe.* Habana : s.n., 2012.

27. **IEEE. .** *Guide to the Software Engineering Body of Knowledge (SWEBOK).* 2004.

28. **Aduana General de la República de Cuba.** Ventanilla Única Aduanera. [En línea] [Citado el: 3 de 4 de 2016.] sua.aduana.cu/vua/.

29. Ventajas de la comunicación por email. [En línea] [Citado el: 20 de 3 de 2016.] http://www.teenvio.com/es/consejos/ventajas-de-la-comunicacion-por-email/.

30. **Jose, M.C.** Spark System. [En línea] 2007 . [Citado el: 2 de 3 de 2016.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_packagediagram.html..

31. **Rumbaugh.** *El proceso unificado de desarrollo de software.* Wesley : s.n., 1999.

32. **Eguiluz, J.** *Desarrollo Ágil con Symfony2.* 2012.

33. [En línea] [Citado el: 12 de 4 de 2016.] http://gitnacho.github.io/symfony-docs-es/components/event_dispatcher.html.

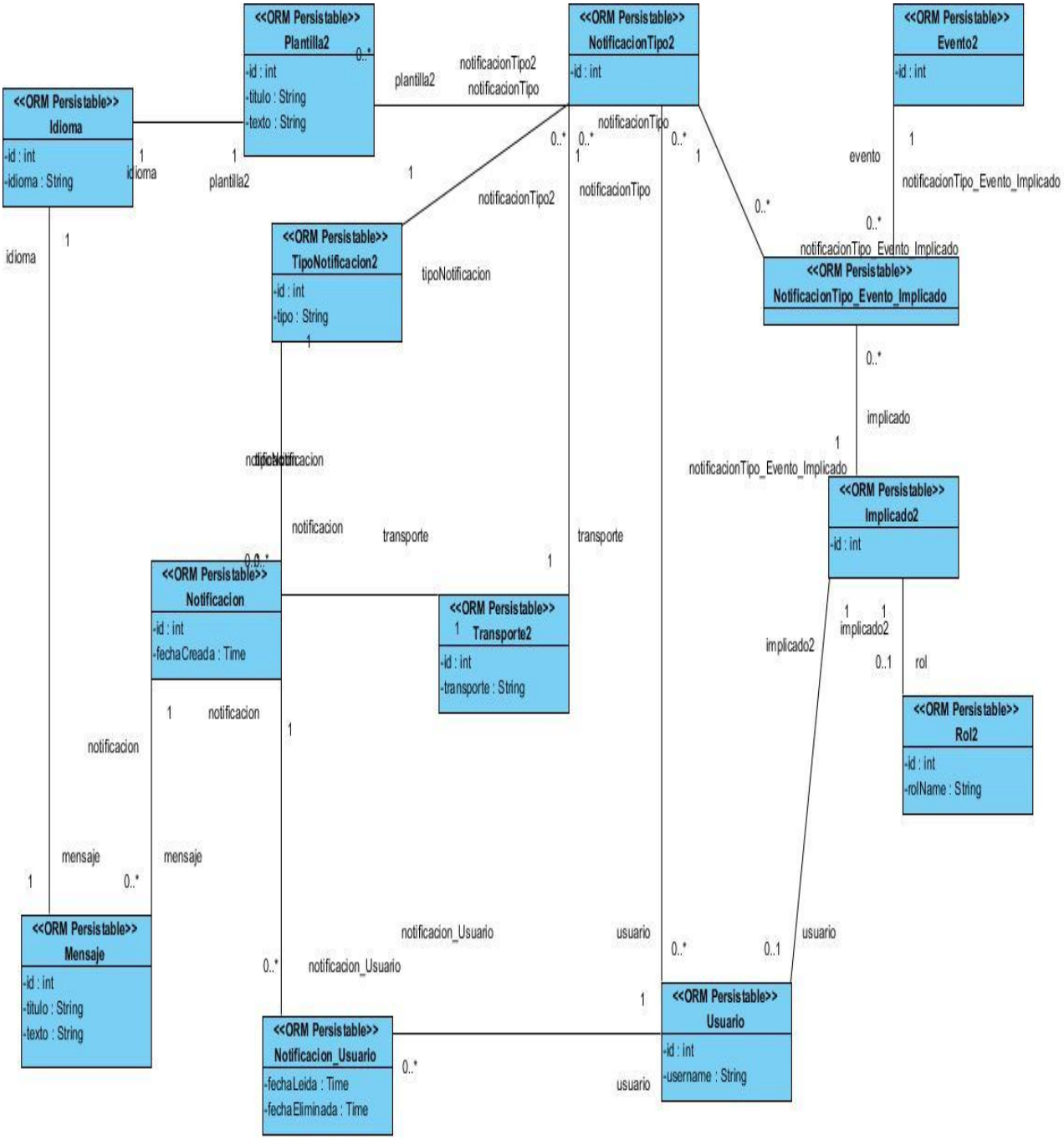
34. **Pressman, Roger S.** *Ingeniería de software. Un enfoque práctico.* La Habana : Felix Varela, 2005.

35. **Villarroel Salcedo, José Luis. .** *Sistemas de Tiempo Real.* 2015.

37. **Larman, C.** *UML y patrones. Introducción al análisis orientado a objetos.* Mexico : D.S.White, 2001.

Anexo I

Diagrama de clases persistentes



Anexo II: Criterio de los expertos.

Cálculo del coeficiente de concordancia

Tabla 1: Matriz de Competencias (**C**) expresada por los expertos (**E**).

COMPETENCIAS (C)	**Expertos (E)**				
	E_1	**E_2**	**E_3**	**E_4**	**E_5**
El componente propuesto aumenta la configurabilidad de las notificaciones del sistema Ventanilla Única Aduanera	X	X	X	X	X
Se aumente la adaptabilidad del sistema frente a nuevos escenarios que requieran el uso de notificaciones	X	X	X	X	-
la implantación del componente de gestión de notificaciones del sistema VUA es beneficiosa para los usuarios del sistema	X	X	X	X	X
El componente tiene posibilidades reales de aplicación	X	X	X	X	-
El componente presenta adaptabilidad ante nuevos escenarios.	X	X	X	-	X
Puede repercutir positivamente en el sistema Ventanilla Única Aduanera.	X	X	X	X	-

Tabla 2: Matriz de Competencias (**C**) Depuradas con Nivel de Concordancia.

COMPETENCIAS (C)	**Expertos (E)**					**Cc%**
	E_1	**E_2**	**E_3**	**E_4**	**E_5**	
El componente propuesto aumenta la configurabilidad de las notificaciones del sistema Ventanilla Única Aduanera.						100
Se aumente la adaptabilidad del sistema frente a nuevos escenarios que requieran el uso de notificaciones.					**N**	80
la implantación del componente de gestión de notificaciones del sistema VUA es beneficiosa para los usuarios del sistema.						100
El componente tiene posibilidades reales de aplicación.					**N**	80
El componente presenta adaptabilidad ante nuevos escenarios.				**N**		80
Puede repercutir positivamente en el sistema Ventanilla Única Aduanera.					**N**	80

Determinación del coeficiente w de Kendall

Competencias	**E_1**	**E_2**	**E_3**	**E_4**	**E_5**	**Rj**	**Rj-∑R/N**	**(Rj-∑R/N)²**
El componente propuesto aumenta la configurabilidad de las notificaciones del sistema Ventanilla Única Aduanera.	10	8	8	10	4	40	**12**	**144**
Se aumente la adaptabilidad del sistema frente a nuevos escenarios que requieran el uso de notificaciones.	10	4	10	10	4	38	**-8**	**64**
la implantación del componente de gestión de notificaciones del sistema VUA es beneficiosa para los usuarios del sistema.	10	10	10	10	10	50	**20**	**400**
El componente tiene posibilidades reales de aplicación.	8	10	8	8	2	36	**-10**	**100**
El componente presenta adaptabilidad ante nuevos escenarios.	8	10	8	10	6	42	**10**	**100**
Puede repercutir positivamente en el sistema Ventanilla Única Aduanera.	10	6	8	10	4	38	**-11**	**121**
Σ						**244**		**929**

$$W = \frac{S}{\frac{1}{12} K^2 (N^3 - N)}$$

$$W = \frac{929}{\frac{1}{12} 81(216 - 6)}$$

$$W = 66\%$$

Donde:

N: Número de competencias.

K: Número de expertos.

$$S = \sum \left(R_j - \frac{\sum R}{N} \right)^2$$

S = 929

Donde:

R: Los valores que asignaron los expertos a cada competencia.

Anexo III

Descripción de requisitos.

4. 3.1 Descripción textual del requisito

Precondiciones	1. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*
Flujo de eventos	
Flujo básico modificar notificación	
1.	*Se muestra un listado con las notificaciones existentes con los siguientes datos: Nombre, Titulo, Evento, Implicado y cuerpo. Brinda también las siguientes opciones:* • Adicionar • Editar • Eliminar
2.	El usuario marca la categoría que se desea editar y selecciona la opción “Editar”. En caso de seleccionar la opción “Adicionar” ver requisito funcional Adicionar Notificación. En caso de seleccionar la opción “Eliminar” ver requisito funcional Eliminar Notificación.
3.	Se debe mostrar una interfaz con los siguientes campos *Ver Ilustración2:* Nombre Titulo Tipo de notificación Evento Implicado Cuerpo Opción de idioma Permite además las opciones “Aceptar” y “Cancelar”
4.	El usuario modifica los valores que desee.
5.	El usuario selecciona la opción “Aceptar”. En caso de que los campos requeridos no estén llenos ver flujo alterno 5.*a Datos incompletos*. En caso de que exista la notificacíon insertada ver flujo alterno 5.*b Notificación repetida.*

	En caso de seleccionar la opción "Cancelar" ver flujo alternativo 5.c Cancelar.
6.	Se actualiza el listado de las categorías y muestra el mensaje "Notificación modificada satisfactoriamente".
7.	Concluye la ejecución del requisito.

Pos-condiciones

1.	Se modifica la notificación satisfactoriamente
1.	

Flujos alternativos

Flujo alternativo 5.a Datos incompletos

1.	Se muestra el mensaje de error "Este campo es requerido".
2.	Vuelve al paso 3 del flujo básico.

Pos-condiciones

1.	Se muestra un mensaje de error informando que es obligatorio el llenado de los campos marcados como requeridos.

Flujo alternativo 5.b Notificación repetida

1.	Se muestra el mensaje de error "La notificación ya existe en Se".
2.	Vuelve al paso 3 del flujo básico.

Pos-condiciones

1.	Se muestra un mensaje de error informando que ya existe la notificación.

Flujo alternativo 7.d Cancelar.

1.	Cancela la acción de Modificar Notificación.
2.	Vuelve al paso 1 del flujo básico

Pos-condiciones

1.	Cancela la acción de Modificar Notificación.

Validaciones

1.	Todos los campos son obligatorios.

Conceptos	**Notificación**	*Visibles en la interfaz:* *Nombre* *Titulo* *Tipo de notificación* *Evento*

		implicado *Cuerpo* *Idioma* *Utilizados internamente:* *N/A*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A.*	

3.2 Prototipo elemental de interfaz gráfica de usuario

5. 3.3 Formatos de entrada/salida

6. 3.4 Entradas

N/A

7. 3.5 Salidas

N/A

4 Especificación de Requisito Eliminar Notificación

8. 4.1 Descripción textual del requisito Eliminar Notificación.

Precondiciones	2. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*

Flujo de eventos	
Flujo básico modificar notificación	
1.	*Se muestra un listado con las notificaciones existentes con los siguientes datos: Nombre, Titulo, Evento, Implicado y cuerpo. Brinda también las siguientes opciones:* • Adicionar • Editar • Eliminar
2.	El usuario marca la notificación que se desea eliminar y selecciona la opción "Eliminar". *Ver Ilustración 3.* En caso de seleccionar la opción "Adicionar" ver requisito funcional Adicionar Notificación. En caso de seleccionar la opción "Editar" ver requisito funcional Modificar Notificación.
3.	Se muestra un mensaje de confirmación "¿Está seguro que desea eliminar la notificación seleccionada?". Además las opciones "Aceptar" y "Cancelar"
4.	El usuario selecciona el botón "Aceptar" del mensaje. En caso de seleccionar el botón "Cancelar" ver flujo alternativo 4.*a Cancelar*
5..	Se elimina la notificación, actualiza el listado de las mismas y muestra el mensaje "Notificación eliminada satisfactoriamente".
6..	Concluye la ejecución del requisito.
Pos-condiciones	
1.	Se elimina la notificación y actualiza el listado de las mismas.
Flujos alternativos	
Flujo alternativo 4.a Cancelar.	
1.	Cancela la acción de Eliminar Notificación.
2.	Vuelve al paso 1 del flujo básico
Pos-condiciones	
1.	Cancela la acción de Eliminar Notificación.
Validaciones	
1.	N/A

Conceptos	**Categoría**	Visibles en la interfaz: • Nombre. • Alias. • Autor

		• Fecha de creada • Hora de creada • Descripción.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

9. 4.2 Prototipo elemental de interfaz gráfica de usuario

Formatos de entrada/salida

10. 4.4 Entradas

N/A

11. 4.5 Salidas

N/A

5 Especificación de Requisito Adicionar Tipo Notificación

12. 5.1 Descripción textual del requisito

Precondiciones	3. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*
Flujo de eventos	
Flujo básico Tipo Notificación	
1.	*Se muestra una interfaz con los campos para los datos tipo e ícono. Brinda también las siguientes opciones:* • Adicionar • Modificar • Eliminar
2.	El usuario seleccionar la opción “Adicionar”. En caso de seleccionar la opción “Eliminar” ver requisito funcional Eliminar Tipo de Notificación En caso de seleccionar la opción “Modificar” ver requisito funcional Modificar Tipo de Notificación
3.	Se debe mostrar una interfaz con los siguientes campos *Ver Ilustración:* Tipo Ícono Permite además las opciones “Aceptar” y “Cancelar”
4.	El usuario introduce los datos.
5.	El usuario selecciona la opción “Aceptar”. En caso de que los campos requeridos no estén llenos ver flujo alterno 5.*a Datos incompletos*. En caso de seleccionar la opción “Cancelar” ver flujo alternativo 5.*c Cancelar.*
6.	Se muestra el mensaje ¨ Tipo Notificacion creada con exito¨
7.	Concluye el requisito

Pos-condiciones

2.	Se adiciona correctamente el tipo notificación

Flujos alternativos

Flujo alternativo 5.a Datos incompletos

1.	Se muestra el mensaje de error "Este campo es requerido".
2.	Vuelve al paso 3 del flujo básico.

Pos-condiciones

1.	Se muestra un mensaje de error informando que es obligatorio el llenado de los campos marcados como requeridos.

Flujo alternativo 8.c Cancelar.

1	Cancela la acción de Adicionar Tipo de Notificación.
2	Vuelve al paso 1 del flujo básico

Validaciones

1.	*Todos los campos son obligatorios*

Conceptos	**Tipo Notificación**	*Visibles en la interfaz:* *Tipo* *Ícono*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

13. 5.2 Prototipo elemental de interfaz gráfica de usuario

14. 5.3 Formatos de entrada/salida

15. 5.4 Entradas

N/A

16. 5.5 Salidas

N/A

6 Especificación de Requisito Modificar Tipo Notificación

17. 6.1 Descripción textual del requisito

Precondiciones	18. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*
Flujo de eventos	
Flujo básico Tipo Notificación	
1.	*Se muestra una interfaz con los campos para los datos tipo e ícono. Brinda también las siguientes opciones:* • Adicionar • Modificar • Eliminar
2.	El usuario seleccionar la opción “Modificar”. En caso de seleccionar la opción “Eliminar” ver requisito funcional Eliminar Tipo de Notificación En caso de seleccionar la opción “Adicionar” ver requisito funcional Adicionar Tipo de Notificación
3.	Se debe mostrar una interfaz con los siguientes campos *Ver Ilustración:* Tipo Ícono Permite además las opciones “Aceptar” y “Cancelar”
4.	El usuario introduce los datos.
5.	El usuario selecciona la opción “Aceptar”.

	En caso de que los campos requeridos no estén llenos ver flujo alterno 5.*a Datos incompletos*. En caso de seleccionar la opción "Cancelar" ver flujo alternativo 5.*c Cancelar.*
6.	Se muestra el mensaje ¨ Tipo Notificacion modificada con exito¨
7.	Concluye el requisito

Pos-condiciones

1.	Se modifica correctamente el tipo notificación

Flujos alternativos

Flujo alternativo 5.a Datos incompletos

1.	Se muestra el mensaje de error "Este campo es requerido".
2.	Vuelve al paso 3 del flujo básico.

Pos-condiciones

2.	Se muestra un mensaje de error informando que es obligatorio el llenado de los campos marcados como requeridos.

Flujo alternativo 5.c Cancelar.

1	Cancela la acción de Modificar tipo de Notificación.
2	Vuelve al paso 1 del flujo básico

Validaciones

2.	*Todos los campos son obligatorios*

Conceptos	**Tipo Notificación**	*Visibles en la interfaz:* *Tipo* *Ícono*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

19. Prototipo elemental de interfaz gráfica de usuario

20. 6.3 Formatos de entrada/salida

21. 6.4 Entradas

N/A

22. 6.5 Salidas

N/A

7 Especificación de Requisito Eliminar Tipo Notificación

23. 7.1 Descripción textual del requisito

Precondiciones	5. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*
Flujo de eventos	
Flujo básico Tipo Notificación	
8.	*Se muestra una interfaz con los campos para los datos tipo e ícono. Brinda también las siguientes opciones:* • Adicionar • Modificar • Eliminar
9.	El usuario seleccionar la opción “Eliminar”. En caso de seleccionar la opción “Modificar” ver requisito funcional Modificar Tipo de Notificación En caso de seleccionar la opción “Adicionar” ver requisito funcional Adicionar Tipo de Notificación
10.	Se muestra un mensaje de confirmación “¿Está seguro que desea eliminar el tipo de notificación seleccionado?”. Además las opciones “Aceptar” y “Cancelar”
11.	El usuario selecciona el botón “Aceptar” del mensaje. En caso de seleccionar el botón “Cancelar” ver flujo alternativo 4.*a Cancelar*
12.	Concluye el requisito

Pos-condiciones

3.	Se modifica elimina correctamente el tipo de notificación
Flujos alternativos	
	Flujo alternativo 4.a Cancelar.
1	Cancela la acción de Modificar tipo de Notificación.
2	Vuelve al paso 1 del flujo básico
Validaciones	
	N/A

Conceptos	**Tipo Notificación**	*Visibles en la interfaz:*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

Prototipo elemental de interfaz gráfica de usuario

24. 7.3 Formatos de entrada/salida

25. 7.4 Entradas

N/A

26. 7.5 Salidas

N/A

8 Especificación de Requisito Adicionar Transporte

27. 8.1 Descripción textual del requisito

Precondiciones	6. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*
Flujo de eventos	
Flujo básico Transporte	
1.	*Se muestra una interfaz con los campos para los datos tipo e ícono. Brinda también las siguientes opciones:* • Adicionar • Modificar • Eliminar
2.	El usuario seleccionar la opción “Adicionar”. En caso de seleccionar la opción “Eliminar” ver requisito funcional Eliminar Transporte En caso de seleccionar la opción “Modificar” ver requisito funcional Modificar Transporte
3.	Se debe mostrar una interfaz con los siguientes campos *Ver Ilustración:* Tipo Ícono Permite además las opciones “Aceptar” y “Cancelar”
4.	El usuario introduce los datos.
5.	El usuario selecciona la opción “Aceptar”. En caso de que los campos requeridos no estén llenos ver flujo alterno 5.*a Datos incompletos*. En caso de seleccionar la opción “Cancelar” ver flujo alternativo 5.*c Cancelar.*
6.	Se muestra el mensaje ¨ Tipo Notificacion creada con exito¨
7.	Concluye el requisito

Pos-condiciones	
1.	Se adiciona correctamente el tipo notificación
Flujos alternativos	
Flujo alternativo 5.a Datos incompletos	
1.	Se muestra el mensaje de error “Este campo es requerido”.
2.	Vuelve al paso 3 del flujo básico.
Pos-condiciones	
1.	Se muestra un mensaje de error informando que es obligatorio el llenado de los campos marcados como requeridos.
Flujo alternativo 8.c Cancelar.	
1	Cancela la acción de Adicionar Transporte.
2	Vuelve al paso 1 del flujo básico
Validaciones	
1.	*Todos los campos son obligatorios*

Conceptos	**Transporte**	*Visibles en la interfaz:* *Tipo* *Ícono*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

28. 8.2 Prototipo elemental de interfaz gráfica de usuario

29. 8.3 Formatos de entrada/salida

30. 8.4 Entradas

N/A

31. 8.5 Salidas

N/A

9 Especificación de Requisito Modificar Transporte

32. 9.1 Descripción textual del requisito

Precondiciones	7. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*
Flujo de eventos	
Flujo básico Transporte	
1.	*Se muestra una interfaz con los campos para los datos tipo e ícono. Brinda también las siguientes opciones:* • Adicionar • Modificar • Eliminar
2.	El usuario seleccionar la opción “Modificar”. En caso de seleccionar la opción “Eliminar” ver requisito funcional Eliminar Transporte En caso de seleccionar la opción “Adicionar” ver requisito funcional Adicionar Transporte
3.	Se debe mostrar una interfaz con los siguientes campos *Ver Ilustración:* Tipo Ícono Permite además las opciones “Aceptar” y “Cancelar”
4.	El usuario introduce los datos.
5.	El usuario selecciona la opción “Aceptar”. En caso de que los campos requeridos no estén llenos ver flujo alterno 5.*a Datos incompletos*. En caso de seleccionar la opción “Cancelar” ver flujo alternativo 5.*c Cancelar.*
6.	Se muestra el mensaje ¨ Tipo Notificacion modificada con exito¨
7.	Concluye el requisito

Pos-condiciones	
1.	Se modifica correctamente el tipo notificación
Flujos alternativos	
Flujo alternativo 5.a Datos incompletos	
1.	Se muestra el mensaje de error “Este campo es requerido”.

2.	Vuelve al paso 3 del flujo básico.
Pos-condiciones	
3.	Se muestra un mensaje de error informando que es obligatorio el llenado de los campos marcados como requeridos.
	Flujo alternativo 5.b Cancelar.
1	Cancela la acción de Modificar Transporte.
2	Vuelve al paso 1 del flujo básico
Validaciones	
3.	*Todos los campos son obligatorios*

Conceptos	**Transporte**	*Visibles en la interfaz:* *Tipo* *Ícono*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

9.2 Prototipo elemental de interfaz gráfica de usuario

33. 9.3 Formatos de entrada/salida

34. 9.4 Entradas

N/A

35. 9.5 Salidas

N/A

10 Especificación de Requisito Eliminar Transporte

36. 10.1 Descripción textual del requisito

Precondiciones	8. *El usuario encargado debe tener los permisos necesarios y estar autenticado en Se.*
Flujo de eventos	
Flujo básico Transporte	
13.	*Se muestra una interfaz con los campos para los datos tipo e ícono. Brinda también las siguientes opciones:* • Adicionar • Modificar • Eliminar
14.	El usuario seleccionar la opción “Eliminar”. En caso de seleccionar la opción “Modificar” ver requisito funcional Modificar Transporte En caso de seleccionar la opción “Adicionar” ver requisito funcional Adicionar Transporte
15.	Se muestra un mensaje de confirmación “¿Está seguro que desea eliminar el Transporte seleccionado?”. Además las opciones “Aceptar” y “Cancelar”
16.	El usuario selecciona el botón “Aceptar” del mensaje. En caso de seleccionar el botón “Cancelar” ver flujo alternativo 4.*a Cancelar*
17.	Concluye el requisito
Pos-condiciones	

4.	Se modifica elimina correctamente el Transporte

Flujos alternativos

Flujo alternativo 4.a Cancelar.

1	Cancela la acción de Modificar Transporte.
2	Vuelve al paso 1 del flujo básico

Validaciones

N/A

Conceptos	**Transporte**	*Visibles en la interfaz:*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

37. 10.4 Entradas

N/A

38. 10.5 Salidas

N/A

11 Especificación de Requisito Lanzar Notificación

39. 11.1 Descripción textual del requisito

Precondiciones	*1. Se debe lanzar un evento* *2. Se ejecutó el requisito Capturar evento* *3. Se ejecutó el requisito Obtener notificaciones vinculadas a evento*
Flujo de eventos	
Flujo básico Lanzar Notificación	
1.	Para cada notificación obtenida en el requisito Obtener notificaciones vinculadas a evento Se obtiene su implicado
2.	Se obtiene su transporte.
3.	Se conforma la notificación con la plantilla obtenida
4.	Se envía la notificación al implicado utilizando el transporte definido
5.	Concluye el requisito

Pos-condiciones	
1.	Se enviá la notificación
Validaciones	
	N/A

Conceptos	**Notificación**	*Usadas por Se* *Transporte* *implicado* *notificación*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

40. 11.2 Prototipo elemental de interfaz gráfica de usuario

N/A

41. 11.3 Formatos de entrada/salida

42. 11.4 Entradas

N/A

43. 11.5 Salidas

N/A

12 Especificación de Requisito Capturar evento

44. 12.1 Descripción textual del requisito

Precondiciones	*1. Se debe lanzar un evento*
Flujo de eventos	
Flujo básico Capturar evento	
1.	Se captura el evento a través de un listener
2.	Busca que notificaciones tiene asociadas el evento ver requisito Obtener notificaciones vinculadas a eventos
3.	Concluye el requisito

Pos-condiciones

1.	Se captura el evento
Validaciones	
	N/A

Conceptos	**Notificación**	*Usadas por Se* *Evento asociado*
Requisitos especiales	*N/A*	
Asuntos	*N/A*	

pendientes	

45. 12.2 Prototipo elemental de interfaz gráfica de usuario

N/A

46. 12.3 Formatos de entrada/salida

47. 12.4 Entradas

N/A

48. 12.5 Salidas

N/A

13 Especificación de Requisito Obtener notificaciones vinculadas a evento

49. 13.1 Descripción textual del requisito

Precondiciones	*1. Se debe lanzar un evento* *2.Se ejecutó el requisito capturar evento*	uiu
Flujo de eventos		
Flujo básico Obtener Notificación vinculada a evento		
1.	Cuando se captura el evento ver requisito capturar evento Se obtiene su identificador	
2.	Se obtiene su entidad asociada	
3.	Comprueba en las notificaciones si existe alguna notificación que se ejecute con ese evnto vinculada a la entidad obtenida Si no existe ninguna notificación se va al flujo alternativo 3.a	
4.	Se guarda una lista con cada notificación encontrada	
5..	Concluye el requisito	

Pos-condiciones

1.	Se devuelve una lista con las notificaciones asociadas al evento

Flujo alternativo 5.a Datos incompletos

1.	Concluye el requisito

Pos-condiciones

1.	

Validaciones

	N/A

Conceptos	**Notificación**	*Usadas por Se* *Evento asociado* *Entidad Asociada*
Requisitos especiales	*N/A*	
Asuntos pendientes	*N/A*	

50. 13.2 Prototipo elemental de interfaz gráfica de usuario

N/A

51. 13.3 Formatos de entrada/salida

52. 13.4 Entradas

N/A

53. 13.5 Salidas

N/A