Universidad de las Ciencias Informáticas

Facultad 3

Herramienta para la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe

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

Autores: William Amed Tamayo Guevara

Ricardo Enrique Suárez Riquenes

Tutores: Ing. René Bauta Camejo

Ing. Claudia Bravo Batista

Ing. Mileydi Sarduy Pérez

La Habana, Junio de 2014

"Año 56 de la Revolución"

Declaración de autoría

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

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

William Amed Tamayo Guevara

_____

Firma del Autor

Ricardo Enrique Suárez Riquenes

_____

Firma del Autor

Claudia Bravo Batista

_____

Firma del Tutor

Mileydis M. Sarduy Pérez

_____

Firma del Tutor

René R. Bauta Camejo

_____

Firma del Tutor

Datos de contacto

Tutor: Ing. Claudia Bravo Batista.

Graduado de Ingeniería en Ciencias Informáticas en Julio del 2012.
Dirección: Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km. 2 ½. Torrens, municipio de La Lisa. La Habana, Cuba.

E-mail: cbravo@uci.cu

Tutor: Ing. Mileidy M. Sarduy Pérez.

Graduado de Ingeniería en Ciencias Informáticas en Julio del 2009. Se ha desempeñado como planificadora, analista principal y actualmente como Jefa del proyecto Gestión Integral de Seguridad (Acaxia).
Dirección: Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km. 2 ½. Torrens, municipio de La Lisa. La Habana, Cuba.

E-mail: mmsarduy@uci.cu

Tutor: Ing René R. Bauta Camejo.

Graduado de Ingeniería en Ciencias Informáticas en Julio del 2009. Se desempeña actualmente como Jefe del Departamento de Tecnología del Centro de Informatización de la Gestión de Entidades (CEIGE).
Dirección: Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km. 2 ½. Torrens, municipio de La Lisa. La Habana, Cuba.

E-mail: rrbauta@uci.cu

Dedicatoria

A mis padres y a mis abuelos, que dieron todo porque llegara hasta aquí.

William

A Masielita, Javi y Lore, para que luchen por sus sueños y los puedan cumplir.

A mi mamá y mi papá, por hacerme sentir orgulloso de ser su hijo.

Ricardo

Agradecimientos

Quiero agradecer a mis familiares, que siempre me han apoyado y me han brindado confianza y cariño: los "yoyos", abuela y Lozy, tía Tania, Jorge, y mi prima Vicky, también a abuelo y Elena. También a los de Camagüey: tío Henry y Lari, tío Marcos, tío Orlando y Guille, a mi abuelo Guillermo y a mi primo Enriquito. A mi primo Alejandro por los buenos tiempos de la vida. Agradezco también a mi primo Ernesto, por los sabios consejos y las lecciones de vida que me ha transmitido. Por ser mi hermano mayor, el que no tuve por ser primerizo.

A mis amistades del pre: Rey y Carlos; y a los que conocí en la universidad: Isa, Leodanny, Any, Rosalina, Camilo, Yendrie y Raynel, gracias a ustedes por su tiempo para escucharme, por enseñarme no pocas cosas, por hacerme reflexionar y por compartir conmigo en buenas y malas. Gracias a Odelkis, Eylis y Hermito por estar siempre ahí. La gente de mi grupo, Osviel, las Mari, Jorge, Alberto, a Ornoldo, a Tania por aceptar cantar conmigo en el festival, a Abraham, y en especial a "el Yoan" gracias por la sinceridad aunque fuera cruda y por las enseñanzas.

A mis amigos Eddy y Geider gracias por los tiempos del café, los de amanecer estudiando para cualquier prueba, las recres y los de quedarnos en la UCI cuando estaba desierta. De verdad gracias por brindarme su amistad.

Agradezco a mi amigo Ramón, hermano de duras batallas y grandes festejos durante 14 años. También agradezco a mi amigo Willy, llegado más reciente, pero que también me ha acompañado en momentos difíciles y gratos de estos últimos 5 años. Le agradezco que haya sido el compañero de tesis que ha sido: dedicado, responsable e inteligente. Agradecezco a mis tutores Claudia, Mile y René, por la atención y la asesoría. Gracias a Pedro por el tiempo que nos dedicó y todas las sugerencias.

Agradecerle especialmente a mi chiquita, mi otra compañera de tesis y de la vida en estos últimos 4 años. Por los buenos momentos, por todo lo aprendido contigo, por la paciencia y dedicación, por todo tu amor. Gracias a toda tu familia por acogerme y en especial a Betty por ese cariño que me profesa.

Gracias a mi papá, el mejor de mis mejores amigos, el que me ha apoyado en todo momento con su consejo certero. Parte de lo que he llegado a ser hoy es gracias a ti. Gracias a Massi también, por la confianza y el amor que me brindó, por quererme como a su hijo. Gracias a mis hermanitos Javier y Massiel, por su amor hacia mí, quiero estar para ustedes siempre que me necesiten.

Por último y no menos importante quiero agradecer a mi mamá, mi mejor amiga, mi confidente, mi guía de la vida, por darme este tamaño y la educación que tengo. Gracias a ti también he llegado a ser lo que soy hoy en día. Gracias por estar en los momentos difíciles, por brindarme tu vida y por todos los momentos lindos que no regresarán, pero que tú y yo nunca olvidaremos. Gracias a mi hermanita Lore, mi cicloncito, por tu adoración, para ti también quiero estar siempre que me necesites.

Gracias a abuela Mirta y a Rey, aunque ya no estén siempre estarán en mi corazón.

Gracias a Fidel y a la Revolución por darme la oportunidad de ser ingeniero.

Al T3AM Dragon3s, por todas las locuras y revoluciones.

Gracias a todos los que incidieron de una forma u otra en mi formación como profesional.

Ricardo Enrique Suárez Riquenes

Agradecimientos

A mis padres, por hacerme la persona que soy hoy, a mi mamá por ser mi guía, eres la persona que más admiro en el mundo. A mi papá por estar siempre ahí para mí. Los quiero muchísimo.
A mi novia Lilian, por ser la persona que ha estado conmigo en los momentos más difíciles en la universidad, por el cariño y el amor que me das, por todos los momentos mágicos que hemos pasado juntos.
A mis tíos, a los que están y a los que no, a todos les debo mucho, gracias por la paciencia y el cariño, los quiero con el alma.
A mis primas lindas, las adoro, gracias por todo, han sido mis hermanas en las buenas y en las malas.
A mis abuelos lindos, gracias por el inmenso amor que me dieron y por ser su nieto del alma.
A mi suegra y mi otra abuela, por quererme como un hijo más.
A mis amigos, Richard, Aris y Ramón gracias por su lealtad y confianza.
A todos mis compañeros de aula, a todas las personas que me han ayudado en todo este tiempo, gracias por dejarme ser partícipe en sus vidas.
A Conde, gracias por toda la atención, eres de mi familia.
A Pedro, por toda la ayuda brindada en los momentos que necesitábamos.
A mis tutores, Claudette, Mile, René y a todos mis compañeros del proyecto.

William Amed Tamayo Guevara

Resumen

El auge y la evolución de las técnicas para ofrecer información a los usuarios, ha causado que cada vez sea más difícil mostrarla de acuerdo con sus necesidades, gustos y preferencias. Alrededor del mundo se realizan numerosos esfuerzos en función de obtener soluciones efectivas para los usuarios, proporcionándoles más opciones de configuración en cuanto a la forma en que reciben la información. La apariencia visual de las herramientas informáticas que la proveen es por tanto un factor decisivo que ha incrementado la exigencia en sus diseños. Para lograr que se adapten a las configuraciones preferidas por los usuarios es necesario modificar los elementos que conforman sus interfaces. Este ajuste de la configuración visual en el marco de trabajo Sauxe, se realiza en el presente de forma manual, modificando directamente los atributos de la capa de presentación en los archivos internos del sistema. Conformar un diseño visual de este modo, conlleva a un notable retraso en la entrega del producto final.

El presente trabajo está orientado a la creación de una herramienta que reduzca el tiempo de personalización de la capa de presentación de Sauxe. Para el desarrollo de la solución se realizó un análisis de los conceptos asociados al tema, así como de varias soluciones existentes que aportaron aspectos de importancia. En el trabajo se describe la solución propuesta y se realiza un análisis de los resultados obtenidos durante la validación.

Palabras clave: interfaz, tiempo de personalización, diseño.

Índice

Índice de figuras

Índice de tablas

Introducción

El avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha devenido en la informatización de los procesos más importantes que se llevan a cabo diariamente en el mundo, siendo una de las causas por las que numerosas empresas utilizan sistemas de gestión para controlar sus operaciones y servicios. En consecuencia, se requieren interfaces gráficas de usuarios que reflejen los objetivos de las instituciones, se ajusten a las necesidades y preferencias del personal que interactúa con la aplicación, mostrando al mismo tiempo diseños sencillos, fáciles de administrar y logrando una personalización[1] adecuada (Moreno, 2005)

"La personalización se ha convertido en un campo ampliamente investigado, dentro del cual se han implementado una gran variedad de herramientas para proporcionar información de forma personalizada, en base a los patrones de comportamiento y particularidades del usuario, con el fin de aumentar su constancia y conseguir que se sienta identificado" (Kim, 2002). De modo que la personalización, integrada a aplicaciones de cualquier índole, otorga un nivel de profesionalidad y mayor aceptación e interés por parte del cliente. Los usuarios le manifiestan fidelidad a aquel producto que, adaptado a su contexto, los satisface con mayor exactitud.

Cuba ha creado diferentes políticas que tributan al logro de la informatización de la sociedad, donde la Universidad de las Ciencias Informáticas (UCI) funge como cantera en la formación de profesionales integrales y comprometidos con la Revolución. Asimismo, contribuye al desarrollo de productos informáticos en centros que combinan la investigación y la docencia, a la producción de software[2]. El Centro de Informatización de la Gestión de Entidades (CEIGE), es uno de los centros de la UCI que se encarga del desarrollo, despliegue y soporte de software de gestión. Para la creación de sus productos, utiliza el marco de trabajo Sauxe, que brinda una base tecnológica en el desarrollo de aplicaciones web para la gestión de entidades, y con la ventaja de ser una herramienta creada por el Departamento de Tecnología del propio CEIGE.

Los productos que utilizan Sauxe como base tecnológica, necesitan ser personalizados en función de los atributos específicos -definidos en el diseño de identidad- que posee cada institución u organización, como

[1] Dígase personalizar, la acción de modificar o diseñar software para que se adapte a las preferencias de un usuario en específico. (Moreno)
[2] Esencia de la Misión de la Universidad de las Ciencias Informáticas publicada en su sitio web oficial.

cambiar los temas de las interfaces de usuarios, especificando colores e imágenes representativas de la entidad, tipografías, atributos de las fuentes, características de componentes visuales, edición de colores, registro y selección de juego de íconos, entre otras. Teniendo en cuenta que Sauxe posee un volumen elevado de archivos asociados a la capa de presentación y que la personalización se hace de forma manual, es posible que se introduzcan errores durante el proceso. Además la personalización debe ser ejecutada por un personal capacitado con conocimientos del sistema.

Durante la observación efectuada en el proyecto Marco de trabajo para el desarrollo de aplicaciones web de gestión, con cuatro desarrolladores dedicados a esta tarea y estaciones de trabajo con buenas prestaciones para la creación de una nueva propuesta visual, se pudo constatar que la ejecución de estos cambios implica tiempo de desarrollo, con una duración aproximada de 33 horas. Los especialistas expresaron la necesidad de reducir la duración de esta tarea, teniendo en cuenta que es posible recibir concurrentemente solicitudes de rediseño por cada dominio de entidad. Esto suele suceder en el despliegue inicial, aunque en sistemas previamente entregados puede suceder que más de una entidad realice una petición de rediseño simultáneamente.

Luego de haber realizado un análisis de la problemática existente surge como **problema a resolver:** ¿Cómo disminuir el tiempo de personalización de los temas de la capa de presentación en el marco de trabajo Sauxe?

La problemática planteada se enmarca en el siguiente **objeto de estudio:** el proceso de personalización de los temas de la capa de presentación en tecnologías web; y posee como **campo de acción:** el proceso de personalización de las interfaces de usuario en el marco de trabajo Sauxe.

El **objetivo general** que se persigue con esta investigación es desarrollar una herramienta que permita la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe, para disminuir el tiempo de personalización de las interfaces de usuario.

Para darle cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos:**

- Construir el marco teórico de la investigación sobre el proceso de personalización de los temas de la capa de presentación en aplicaciones web para definir una propuesta de solución.
- Realizar el análisis y diseño de la herramienta para la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe.

- Realizar la implementación de la herramienta para la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe.
- Validar la herramienta para la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe utilizando pruebas funcionales y estructurales.
- Validar la investigación utilizando el método pre-experimento.

Se espera alcanzar con el desarrollo de la investigación una herramienta para la personalización de los temas de interfaz de usuario en el marco de trabajo Sauxe y su documentación técnica.

Para el desarrollo del presente trabajo se parte de la **idea a defender:** Si se desarrolla una herramienta para la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe, se logrará disminuir el tiempo de personalización de los temas de interfaces de usuario.

Definición y operacionalización de las variables

- **Variable dependiente:** tiempo de personalización de los temas de interfaces de usuario.
- **Variable independiente:** herramienta para la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe.

Operacionalización

Tabla 1.Operacionalización de la variable dependiente

Nombre de la variable	U/M
Tiempo de personalización de interfaces de usuario	Horas

Muestreo

Población: La población se define como el *"conjunto de elementos que tengan una o más propiedades en común definidas por el investigador y que puede ser desde toda la realidad, hasta un grupo muy reducido de fenómenos"* (León, 2008). Conjunto de proyectos que utilizan como base tecnológica el marco de trabajo Sauxe.

Unidad de estudio: son los *"elementos, fenómenos sujetos o procesos que integran la población y pueden ser individuos, grupos de personas, hechos, procesos, talleres, turno de trabajo, empresas, documentos"* (León, 2008). Personalización de interfaz de usuario en los proyectos que utilicen como base tecnológica el marco de trabajo Sauxe.

Selección de la muestra y criterio de selección

Muestra: Proyecto Marco de trabajo para el desarrollo de aplicaciones web de gestión.

Criterio de selección: Para la selección de la muestra se utilizó la técnica No probabilística de muestreo, con el objetivo de garantizar que al menos uno de los elementos seleccionados de la población tenga la característica deseada (Grau, y otros, 2004), que para este caso significa que debe tener integrada la Herramienta para la personalización de temas de interfaz de usuario.

Diseño del experimento

Tipo de experimento: Para la validación de la propuesta se realizará un diseño pre experimental.

Instrumentos: Para medir la variable operacional se utilizarán las encuestas.

Métodos de investigación científica a aplicar:

Métodos teóricos:

- Histórico-lógico: utilizado para realizar un estudio acerca de la trayectoria, evolución y principales tendencias en la personalización de interfaces de usuario utilizando tecnologías web.
- Analítico-sintético: utilizado para procesar la documentación existente sobre las principales características de los sistemas para la personalización de interfaces de usuario, análisis determinante en la factibilidad de uso de alguna propiedad en la herramienta a desarrollar.
- Modelación: utilizado para representar de forma visual las características estructurales de la herramienta, así como el flujo de los distintos escenarios definidos en el diseño de la solución.

Métodos empíricos:

- Observación: utilizado para analizar el comportamiento del sistema en cuanto a los tipos de acciones que realizan en diferentes situaciones y medir los tiempos de respuesta.

- Experimentación: utilizado para determinar una forma confiable de personalización de interfaces de usuario de manera que el flujo sea lo más pequeño posible.

Para dar cumplimiento a los objetivos trazados, el presente documento estará estructurado en tres capítulos. A continuación se expone una breve descripción de cada uno de ellos.

Capítulo 1: Fundamentación Teórica

En este capítulo se describen los conceptos fundamentales que se aplicarán a lo largo de la investigación. Se realiza un estudio del estado del arte referente a la edición y diseño de interfaces en el mundo y en la universidad. Además se hace una descripción de las tecnologías que se emplearán, así como del Modelo de desarrollo de Software definido por el CEIGE, que guiará todo el proceso de la construcción de la herramienta.

Capítulo 2: Propuesta de solución

Se realiza una descripción de la solución y sus principales características. Se presenta el modelo conceptual, así como los requisitos funcionales y no funcionales identificados. Se exponen además: el modelo de diseño, prototipos de interfaces, el modelo de datos y los diagramas de clases. Se identifican los patrones arquitectónicos y de diseño a utilizar para el desarrollo de la herramienta.

Capítulo 3: Implementación y pruebas

En este capítulo se analizan los estándares de codificación utilizados para lograr un buen entendimiento y legibilidad del código. Se exponen los resultados de la aplicación de las métricas Tamaño Operacional de Clases y Relaciones entre Clases al diseño elaborado. Se muestran los resultados de las pruebas realizadas a la solución para demostrar su correcto funcionamiento y la validación de la investigación.

Capítulo 1: Fundamentación Teórica

Las tecnologías para el desarrollo de interfaces visuales hacen su aparición desde que se hace latente la creciente necesidad de mostrarle al usuario maneras diferentes y más sencillas de interactuar con la información en un computador. Existen disímiles maneras de generar elementos visuales acordes con el lenguaje de programación, pero su fundamento es igual para todos, crear una máxima abstracción en el desarrollo de las capas visuales. En el siguiente capítulo se realiza el marco conceptual sobre los principales temas relacionados con la investigación. Se analizan además varios editores de interfaces gráficas de usuario, teniendo en cuenta las principales características de estas herramientas que tributen a la solución de la problemática existente.

1.1. Conceptos fundamentales

1.1.1 La interfaz gráfica de usuario

Las Interfaces Gráficas de Usuarios surgen dada la necesidad de hacer más simple el uso de los ordenadores para todo tipo de usuarios y no solo restringir el uso de estos a usuarios avanzados. Por eso ha llegado a convertirse en un hábito usar Interfaces Gráficas de Usuario (GUI[3]), de manera que sea más fácil la interacción del usuario final con el ordenador.

Interfaz: Este concepto se ha definido ampliamente desde diferentes puntos de vista, según el ámbito de conocimientos en que se aplique. Por ejemplo, en la biología es la capa de un organismo que separa su interior del exterior; en la electrónica y las telecomunicaciones, se ha definido como puerto a través del que se envían o reciben señales desde un sistema de subsistemas hacia otros; y en química es la superficie existente entre dos fases distintas en una mezcla heterogénea. La traducción literal atendiendo a su etimología sería: “superficie vista, o lado mediador”. (Expósito, 2008)

Interfaz gráfica de usuario: En términos de informática ha sido definida como un tipo de entorno que representa en la pantalla programas como archivos y opciones por medio de íconos como menús y cuadros de diálogos. El usuario puede seleccionar y activar estas opciones apuntando y haciendo clic con el ratón o, frecuentemente, con el teclado. Un elemento particular (tal como una barra de desplazamiento) trabaja de igual forma para el usuario en todas las aplicaciones, pues la interfaz gráfica de usuario

[3] **GUI**: Por sus siglas en inglés *Graphics User Interface*

proporciona rutinas de software estándar; las aplicaciones invocan a estas rutinas con los parámetros específicos en lugar de intentar reproducirlas a partir de la nada. (Moreno, 2005)

Tema de GUI: Se ha definido de diversas formas por diferentes autores, por ejemplo la Empresa Microsoft lo describe como: *"combinación de imágenes de fondo de escritorio, colores de ventanas y sonidos*". (Windows, 2014) También conocido como skin[4], tema o tapiz, se define como "*una serie de elementos gráficos que, al aplicarse sobre un determinado software, modifican su apariencia externa"* (SkinStudio, 2010). Ambos conceptos reflejan equivalencia para el contexto del marco de trabajo Sauxe, sin embargo para el dominio de la problemática un *tema de interfaz gráfica de usuario* se ha definido genéricamente como: configuración de los archivos pertenecientes a una interfaz gráfica de usuario.

En relación al diseño gráfico, se consideran importantes dos conceptos que se deben tener en cuenta por estar estrechamente relacionados con los conceptos antes mencionados. "*Al observar un diseño lo primero que se define es el color, después el dibujo y luego cualquier símbolo formal, marca, logotipo, palabra o frase."* (ISDI, 2008)

El **color** es el elemento que más afecta la memoria emocional de las personas. En la medida en que la respuesta subjetiva a la imaginería visual responde a la correspondencia entre los símbolos visuales y la experiencia humana, esas experiencias implican transferencias entre un sentido y otro. Por ello en cualquier diseño es importante la proporción de colores utilizados así como la relación entre ellos, pues una vez que el color ha llamado la atención, la ha retenido y ha transmitido un mensaje, su tarea final es contribuir a lograr fidelidad de marca. (ISDI, 2008). Por lo antes expuesto, se relaciona directamente con otro concepto, ya que precisamente la **percepción** de un dibujo y/o una textura puede cambiar en dependencia de los colores empleados a su alrededor, al igual que la percepción de un color puede cambiar con la inclusión de tramas o gráficos. (ISDI, 2008)

1.2 Edición y diseño de interfaces

"Actualmente, en el mundo prima la cultura de la interfaz gráfica fácil de aprender y vistosa, en la que con un simple clic sobre algún componente gráfico que esté presente en pantalla, se sustituye la tediosa tarea de escribir código fuente para que el ordenador interprete que debe realizar alguna acción." (Gordo, 2007) Esta mejora, se ha extrapolado además al escenario de los creadores de software, que si bien han de

[4] **Skin**: piel en ingles.

tener conocimientos suficientes para realizar una interfaz gráfica utilizando líneas de código, también han decidido crear herramientas que agilicen el proceso.

Un editor de interfaces gráficas de usuario, también conocido como *constructor de GUI* (GUI builder) o *diseñador de GUI* (GUI designer), es una herramienta de software que simplifica la creación de GUI permitiéndole al diseñador usar un editor WYSIWYG[5] para arrastrar y soltar componentes hasta obtener lo deseado. Sin un Editor GUI, estas deben ser construidas manualmente especificando los parámetros de cada elemento mediante código, sin ninguna retroalimentación hasta que la aplicación no esté corriendo. (Vromans, 2010)

Diseñador Gráfico: es la herramienta informática que facilita el diseño visual colocando, arrastrando y cambiando elementos visuales como botones, paneles e imágenes, en una interfaz gráfica para su posterior uso en una aplicación. (Moreno, 2005) En muchos de los entornos de desarrollo IDE[6] se integran los módulos de diseño visual, mientras que en otras tecnologías no es objetivo fundamental el uso de uno. Sin embargo poseer un diseñador visual es de vital importancia.

Editor gráfico: programa o herramienta que permite realizar cambios de la configuración visual, tales como atributos de colores, fuentes e imágenes en una interfaz de usuario, mediante el acceso y la escritura de los archivos pertenecientes a la interfaz. (Moreno, 2005)

En general la visión de un diseñador gráfico suele ser más amplia que la de un editor, dado a que no solo se limitan a la modificación de atributos visuales, sino que incluyen el diseño, estructura y funcionamiento de una interfaz. Las características de la herramienta que se desea desarrollar se corresponden con las de un editor gráfico, que comprenderá solamente la modificación de la capa de presentación sin rediseño de su estructura.

En la investigación correspondiente a la elaboración de una herramienta para la personalización de temas de interfaces gráficas de usuario del marco de trabajo Sauxe, se realizó una búsqueda bibliográfica, mediante la cual se encontraron varias herramientas informatizadas para este proceso, aunque todas con sus particularidades. Se analizaron cinco características a tener en cuenta de cada herramienta: modificaciones en tiempo de ejecución; desarrollo sobre tecnologías libres; integración al marco de

[5] **WYSIWYG**: por sus siglas en inglés *What You See Is What You Get.* Traducción: lo que ves es lo que obtienes.
[6] **IDE**: por sus siglas en inglés *Intregrated Development Environment.* Traducción: Entorno de desarrollo integrado.

trabajo; usabilidad del sistema; y edición sin rediseño de la estructura. Estos aspectos se basan fundamentalmente en las funcionalidades que debe cumplir la aplicación a implementar, aunque el indicador con más valor es la integración con la plataforma existente, pues es primordial lograr con el menor costo el mayor acoplamiento posible al marco de trabajo. El análisis de estas herramientas tiene como objetivo llegar a conclusiones que sustenten la creación de una herramienta con las características necesarias para satisfacer a los usuarios de productos desarrollados utilizando Sauxe.

1.2.1 Artisteer 4.0

Es un programa para automatizar el diseño de plantillas destinadas a ser visualizadas en páginas Web, que crea al instante una red de gran apariencia, plantillas únicas y entradas de blog. Permite diseñar temas para insertarlos en gestores de contenidos como Drupal, Joomla, Wordpress o Blogger, consiguiendo tener una web completamente personalizada, en la que es posible configurar hasta el más mínimo aspecto. Sus principales características son: (Artisteer, 2008)

- *Edición de contenido*: se puede diseñar la plantilla y crear páginas con contenido para cualquier tipo de CMS.
- *HTML5 y soporte CSS3*: compatibilidad con la última evolución de los estándares de mejores páginas web estructuradas.
- *Soporte para dispositivos móviles*: genera plantillas escalando el diseño de sus páginas web para dispositivos móviles y portátiles.
- *Diseñador visual del encabezado*: un nuevo diseñador de encabezado que se puede utilizar para agregar funciones que antes no eran posibles como: añadir una presentación de diapositivas a su cabecera o crear collages de fotos, entre otras.
- *Soporte de varios formatos*: ARTX, HTML, JPG, PNG, GIF.
- *Interfaz simple e intuitiva:* es una aplicación fácil de usar.
- *Soporte en diferentes idiomas*.

La herramienta Artisteer, a pesar de ser una opción satisfactoria para la personalización de interfaces gráficas de usuario, tiene el inconveniente de centrarse en la creación de plantillas para páginas y sitios

web que posteriormente pueden ser usadas en un CMS. Además esta aplicación supone un problema en la integración al tratarse de una aplicación de escritorio.

1.2.2 Ext Designer

Ext Designer es un constructor de interfaz gráfica de usuario para las aplicaciones web con Ext JS. Esta aplicación de escritorio facilita la creación de las GUI de manera muy rápida y sencilla, permitiendo dedicar la mayor parte del tiempo a la programación y no tanto al diseño. Ext Designer, proporciona un fácil uso de su entorno *Drag and Drop*[7], con el que en pocos minutos y conocimientos mínimos, se pueden realizar el prototipado rápido de componentes de interfaz de la aplicación, la conexión de componentes de interfaz de datos y exportar de forma correcta el código orientado a objetos para cada componente. El propio entorno *Drag and drop* da la posibilidad de acomodar los componentes a gusto con el mouse. Además proporciona gran cantidad de información técnica en el sitio oficial del producto para evacuar las dudas, e incluso videos y tutoriales aclarativos con buenos ejemplos. Sus características más sobresalientes son: (Sencha, 2008)

- Cambio entre la vista diseño/código: no será necesario hacer todo el diseño mediante código, con la existencia de la vista previa.
- La lista de todos los componentes a la mano: una paleta de componentes definidos que permite seleccionarlos o arrastrarlos hacia el área de trabajo que se esté creando.
- Exportar el proyecto a los archivos fuentes correspondientes.
- Duplicación de Componentes: le proporciona la capacidad de duplicar conjuntos de componentes y luego modificar sus valores sin arrastrar y soltar los componentes para volver a crear configuraciones comunes una y otra vez.
- Transformación de Componentes: el diseñador realizará automáticamente las transformaciones necesarias para convertir un componente en otro según se necesite.
- Deshacer / Rehacer: Permite deshacer un error o rehacer el último cambio que se hizo a través de botones en la barra de herramientas.

[7] **Drag and Drop:** término en inglés que significa *arrastrar* y *soltar*.

Su principal desventaja radica en que esta aplicación no es gratuita, y para utilizarlo es necesario pagar una licencia de software. Además se especifica que es una aplicación de escritorio y por lo tanto no es integrable al marco de trabajo.

1.2.3 Qt Designer

Es una herramienta para el diseño y la creación de interfaces gráficas de usuario para los componentes de Qt. Esta herramienta pertenece al entorno de desarrollo integrado Qt Creator, que proporciona apoyo a la creación y funcionamiento de aplicaciones de Qt para entornos de escritorio (Windows, Linux, FreeBSD y Mac OS) y dispositivos móviles. (Qt-Project, 2013)

Con Qt Designer es posible componer y personalizar los widgets o cuadros de diálogo en un entorno WYSIWYG, además se pueden probar estos componentes con diferentes estilos y resoluciones directamente en el editor. (Qt-Project, 2013)

Los reproductores, formas y componentes creados con Qt Designer se integran perfectamente con el código programado, utilizando las señales y el mecanismo de las franjas horarias de Qt. Cuenta además con el Diseñador Qt Quick, que es una herramienta para el desarrollo de animaciones utilizando un lenguaje de programación declarativa QML. Todas las propiedades establecidas en Qt Designer se pueden cambiar de forma dinámica dentro del código. Además, características como la promoción widget y plugins personalizados permiten utilizar componentes propios en Qt Designer. (Qt-Project, 2013)

El inconveniente de esta solución es su tecnología, enfocado a componentes del lenguaje Qt y apoyo en la creación de aplicaciones de escritorio. En cambio la investigación está encaminada a la búsqueda de una solución para los componentes de Ext JS en aplicaciones sobre la web.

1.2.4 Lycan-Génesis

Lycan-Genesis es una herramienta para el diseño de componentes de Ext JS desarrollada por el Centro de Tecnologías de Gestión de Datos (DATEC) de la Facultad 6 de la UCI. Concebido como un IDE para el desarrollo de aplicaciones del propio centro, principalmente para la Plataforma de Apoyo a la Toma de Decisiones y Soluciones Integrales (PATDSI). Constituye un constructor de componentes reutilizables de Ext JS que permite diseñarlos de manera intuitiva, rápida y cómoda, de modo que promueva la usabilidad y eleve la productividad de los desarrolladores. Implementa buenas prácticas de arquitectura y diseño contribuyendo a la calidad del desarrollo. (Lobo, 2010)

La herramienta provee un entorno de diseño con las siguientes características: (Lobo, 2010)

- Diseñador integrado por los componentes que asisten el trabajo de diseño (Espacio de Diseño, Paleta de Herramientas, Editor de Propiedades, Navegador del Diseño, Editor de Código).
- Diseño vía Sujetar-Arrastrar-Soltar componentes.
- Diseño vía Seleccionar-Mover / Dimensionar.
- Asistentes de alineación de componentes (asistente de acercamiento superior, inferior, derecho izquierdo; asistentes de alineación vertical derecho, centro, izquierdo; asistentes de alineación horizontal superior, centro, inferior).
- Facilidad Deshacer / Rehacer.
- Exportación / Importación de componentes.
- Facilidad de especificación e integración de nuevos componentes para el diseñador.

Entre sus ventajas se puede destacar que: disminuye la cantidad de esfuerzo invertido en capacitación a los desarrolladores, así como eleva la productividad al disminuir los tiempos de desarrollo. Además desarrolla activos reutilizables en PATDSI para cubrir las funcionalidades de diseño de reportes y encuestas en los respectivos módulos. (Lobo, 2010)

Sin embargo el inconveniente de esta solución como opción factible para el problema planteado, radica en que Lycan ha sido concebido para el entorno de la plataforma PATDSI y para el desarrollo de aplicaciones específicamente del centro DATEC. También es necesario resaltar que no provee la creación de temas o modificación de aspectos visuales como las imágenes y colores de los componentes.

Resultados del análisis

Las soluciones informáticas para edición y diseño de interfaces gráficas de usuario descritas en la investigación, centran sus objetivos en determinadas características de los entornos para los que se crearon. Con respecto a los indicadores a tener en cuenta, todas estas herramientas proveen un entorno de diseño con facilidades de uso, mediante el cual se pueden obtener los resultados de forma sencilla e intuitiva. Esto supone un factor importante referido a la usabilidad, por lo que se tiene un buen punto de partida. Mediante este estudio se pudo constatar que excepto Qt Designer, estas soluciones son

orientadas de una forma u otra al desarrollo sobre la web, aunque solo Lycan-Génesis es de naturaleza web, mientras que Qt Designer, Ext Designer y Artisteer son aplicaciones de escritorio, imposibilitando así su integración al marco de trabajo. Sin embargo, Lycan es un IDE pensado para el desarrollo en PATDSI del centro DATEC, por lo que su integración tampoco es posible en el entorno de Sauxe.

Por otro lado, Lycan y Ext Designer no comprenden el tratamiento de la tematización (*theming*[8]) con respecto a modificaciones de imágenes y colores de los componentes, proceso necesario para definir los rasgos visuales de identidad requeridos por las entidades que utilicen productos desarrollados sobre la base de Sauxe. Solo Qt Designer incluye un tratamiento adecuado de la tematización, mientras que Artisteer permite hacer modificaciones de imágenes y colores, pero su objetivo es el de crear plantillas para CMS, o sea que no incluye la creación de componentes para aplicaciones de gestión.

Igualmente, todos tienen basamentos en tecnologías libres, excepto Ext Designer y Artisteer, cuya principal dificultad para su comunidad de desarrolladores es que para utilizarlo es necesario pagar una licencia de software. Otros puntos importantes son: la posibilidad de realizar modificaciones en tiempo de ejecución -funcionalidad que solo Lycan brinda-; y que la concepción de estas soluciones no ha sido definida para realizar edición de interfaces visuales sin hacer un rediseño de la estructura.

A pesar de no ser posible la utilización de las herramientas analizadas, para los propósitos de personalización en el marco de trabajo Sauxe por las razones antes expuestas, sí se considera que aportan el contenido teórico, así como los procedimientos a seguir en la edición de interfaces gráficas. Estos factores fomentan las bases para un mecanismo de personalización encauzado a las tendencias actuales referentes al tema. Se reafirma entonces la necesidad de crear una solución informática que se adecue a la problemática existente, defendiendo la idea de la soberanía tecnológica, integración con el menor costo posible, y que logre la personalización de las interfaces gráficas de usuario en el marco de trabajo Sauxe sin rediseñar la estructura.

1.3 Transformaciones visuales en tiempo de ejecución

Los sistemas basados en la Web han incrementado y extendido sus ámbitos y usos, generando fuertes dependencias al mundo cotidiano, y capturando al usuario con la incorporación de técnicas de personalización. Cualquier software orientado a la visualización de elementos al usuario, está sometido a

[8] **Theming:** término del inglés cuyo espectro aborda el trabajo separado en temas de GUI

una gran variabilidad como consecuencia de una actualización, o a los cambios de la imagen corporativa de la entidad; haciéndose necesaria una reestructuración de la configuración visual. (Martín, 2003)

Debido al avance de las tecnologías web se han creado nuevas formas y alternativas para realizar cambios en la interfaz visual en tiempo de ejecución. La utilización generalizada de la arquitectura cliente servidor para la creación de aplicaciones dinámicas, ha permitido que las modificaciones en los archivos mostrados por la página web, se puedan hacer tanto del lado del cliente como del lado del servidor.

1.3.1 Cambios en el lado del cliente

Una de las formas más comunes de hacer cambios en una interfaz visual web es mediante JavaScript. Los datos de la nueva configuración son cargados desde el servidor o recogidos dentro de la interfaz y son aplicados directamente en la configuración visual. Su principal desventaja es la carencia de persistencia de estos cambios para todos los usuarios, o sea se hace necesario salvar estos datos en el servidor, además el código del script debe descargarse completamente antes de poderse ejecutar y aplicar estos datos en tiempo de ejecución de JavaScript para cada usuario que haga una petición de una interfaz gráfica. (Valdelli, 2006) Los cambios en tiempo de ejecución de JavaScript suelen ser bastante costosos cuando el volumen de datos a aplicar es elevado, se pierde tiempo en procedimientos que son hechos en cada carga de interfaz.

1.3.2 Cambios en el lado del servidor

Es otro modo de hacer cambios en la interfaz de usuario a través de la modificación de archivos de configuración CSS [9]o en el código JavaScript. Dichos cambios se realizan directamente en los archivos contenidos por el servidor, que son reenviados al cliente para ser mostrados al usuario. No suele ser muy utilizado aunque es una de las vías de solución que mejores resultados arroja en cuanto a rendimiento. Su peligro radica en que un mal manejo puede provocar la corrupción total de los archivos fuentes ya sea CSS o JavaScript. (Valdelli, 2006)

Resultados del análisis

El marco de trabajo Sauxe, en estos momentos está siendo sometido a la aplicación de nuevas vías para la reducción de procedimientos de alto costo o relevancia baja. Específicamente la disminución de peticiones al servidor y eliminación de rutinas de procesamiento de baja importancia. Debido a estas

[9] **CSS**: por las siglas en inglés de ***C**ascade **S**tyle **S**heet*

directivas de funcionamiento, y teniendo en cuenta las formas y alternativas mencionadas previamente para hacer cambios en las configuraciones visuales de aplicaciones web dinámicas, se decide utilizar la variante de **cambios en el lado del servidor**. Esta elección está encaminada directamente al rendimiento y a la reducción de los tiempos de respuesta en la personalización de las interfaces visuales.

1.4 Modelo de desarrollo de software

El departamento de Tecnología utiliza el Modelo de desarrollo de software definido por el CEIGE versión 1.2, en el que se detalla el ciclo de vida de sus proyectos, con la incorporación de los distintos subprocesos dictados por el Nivel II de CMMI[10] obtenido por la UCI y reconocido por el SEI[11] como aval de la calidad de su proceso de desarrollo de software. Este modelo estandarizado establece las distintas fases por las que se debe transitar durante el desarrollo, y el conjunto de artefactos que se generan en cada una de ellas. Las fases definidas son: *Inicio* o *Estudio preliminar* y *Desarrollo.* (Obregón, 2013)

La solución que se desea implementar se enmarca en la fase de Desarrollo, en la cual se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el Desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. El objetivo de esta fase es: Obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales (Obregón, 2013).

1.4.1 Flujo de trabajo

En el marco de la presente investigación el flujo de trabajo comprende de las disciplinas pertenecientes a la fase de **Desarrollo**, solo las que se muestran en la siguiente figura:

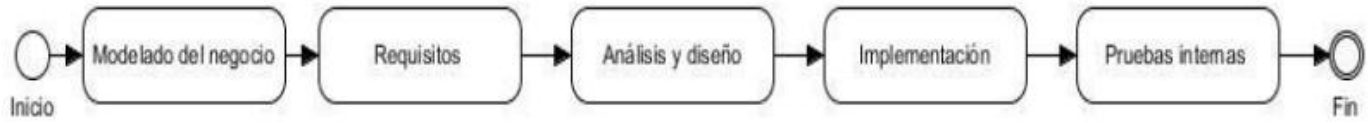


Figura 1. Flujo de trabajo
Fuente: Modelo de desarrollo de software CEIGE

Durante el flujo de trabajo en la disciplina Modelado del negocio se emplea modelo de dominio, no se utiliza modelo de procesos debido a que para el sistema que se desea desarrollar no existen procesos de negocio definidos. En esta disciplina, para el modelo de dominio se genera el modelo conceptual, el cual describe los aspectos del dominio, identificándose los principales elementos físicos o lógicos del negocio

[10] **CMMI:** por las siglas en inglés de ***C**apability **M**aturity **M**odel **I**ntegration*
[11] **SEI:** por las siglas en inglés de ***S**oftware **E**ngineering **I**nstitute*

que ayuden a entender el problema y que generalmente se presentan como clases. El modelo conceptual explica cuáles son y cómo se relacionan los conceptos relevantes en la descripción del dominio de un problema, identificando atributos y asociaciones existentes entre ellos.

En la disciplina Requisitos se identifican los requisitos funcionales, luego se describen y posteriormente se elaboran los prototipos de interfaces de usuario asociados a ellos. Se identifican también los requisitos no funcionales con los que contará la herramienta. Durante la disciplina de Análisis y diseño se define el estilo arquitectónico y el patrón arquitectónico que determinan la estructura fundamental de la herramienta. Se genera el modelo de diseño, artefacto conformado por los diagramas de clases del diseño con estereotipos web y los diagramas de secuencia, mientras que para finalizar esta disciplina se realiza el modelo de datos y se elaboran los diseños de casos de prueba. En la disciplina Implementación se elabora el diagrama de componentes, se definen los estándares de codificación a utilizar y se implementan los requisitos funcionales identificados con sus correspondientes interfaces. En la disciplina Pruebas internas se realizan pruebas de caja negra mediante la técnica de partición equivalente y las pruebas de caja blanca utilizando la técnica del camino básico, también se resuelven las no conformidades detectadas durante la ejecución de estas pruebas. (Obregón, 2013)

1.5 Ingeniería de Requisitos

La Ingeniería de Requisitos se define como *"el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo que satisfaga las necesidades del usuario."* (Pressman, 2005)

Según Pressman, este proceso ayuda a los ingenieros de software a entender mejor el problema para el cual se necesita la solución y su objetivo es darle a todas las partes una explicación escrita del problema. Además incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software.

1.5.1 Técnicas de captura de requisitos

Para realizar el proceso de captura de requisitos existen varias técnicas. La combinación de varias de estas técnicas es una opción factible para lograr un buen proceso de captura de requisitos, y su uso está estrechamente relacionado con las características del proyecto en el que vayan a ser aplicadas.

Entrevistas: es un medio tradicional de obtención de requisitos. La entrevista es un método muy efectivo que permite conocer los problemas de los clientes y encontrar requisitos generales. Para aplicar este método es necesario conocer la forma en que se debe realizar una entrevista para lograr una buena comunicación entre el entrevistador y el entrevistado. (Ganesh, 2008)

Prototipos: es la representación o visualización de parte del sistema. Es una herramienta valiosa para clarificar requisitos confusos. Proveen a los usuarios un contexto para entender mejor qué información necesitan proporcionar. (Ganesh, 2008)

Tormenta de ideas: esta técnica es usada para generar nuevas ideas y encontrar la solución a cuestiones específicas. Es muy común en los comienzos del proceso de ingeniería de requisitos. (Ganesh, 2008)

Observación: este método consiste en la identificación de requisitos cuando se observan a las personas haciendo su trabajo diario. Es muy usado para encontrar requisitos adicionales cuando el usuario es incapaz de explicar los requisitos que necesita para el nuevo sistema. (Ganesh, 2008)

1.6.2 Técnicas de validación de requisitos

El proceso de validación de requisitos comprende actividades que generalmente se realizan una vez que se ha obtenido una primera versión de la documentación de requisitos. Según (Sommerville, 2005) la validación de requisitos *"trata de mostrar que éstos realmente definen el sistema que el cliente desea. Su importancia radica en que los errores en el documento de requisitos pueden conducir a importantes costes al repetir el trabajo cuando son descubiertos durante el desarrollo o después que el sistema esté en uso."* A continuación se muestran varias de estas técnicas:

Revisión de requisitos: Los requisitos son analizados sistemáticamente por un grupo de revisores. Es un proceso manual en el que se verifica el documento de requisitos en cuanto a anomalías y omisiones y se puede gestionar de igual forma que las inspecciones de programas. (Sommerville, 2005)

Generación de casos de pruebas: Los requisitos deben poder probarse. Si las pruebas para éstos se conciben como parte del proceso de validación a menudo revela los problemas en requerimientos. Si una prueba es difícil o imposible de diseñar, normalmente significa que los requerimientos serán difíciles de implementar deberían ser considerados nuevamente. (Sommerville, 2005)

Construcción de prototipos: Consiste en mostrar un modelo ejecutable o semi-ejecutable a los usuarios finales y clientes para que experimenten con este modelo y valoren si satisface sus necesidades. (Sommerville, 2005)

1.6 Modelo de Diseño

El Modelo de diseño ha sido definido como *"una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño*". (Baryolo, 2010) El modelo de diseño puede contener: diagramas, clases, paquetes, interfaces, entre otros, que son utilizados como entrada esencial en las actividades relacionadas a la implementación.

1.6.1 Arquitectura de software

Según varios autores la arquitectura del software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación. Es importante concebir una arquitectura sólida, pues la estructura del sistema depende en gran medida de la arquitectura de software que se utilice para desarrollarlo. Esta estructura se constituye de componentes, que nacen de la noción de abstracción, cumpliendo funciones específicas e interactuando entre sí. (Camacho, 2004)

La definición oficial ofrecida por la IEEE[12] dicta que: *"La Arquitectura del Software es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución".* (IEEE, 2007)

En la arquitectura de software se determina el estilo arquitectónico y el patrón arquitectónico, cuyas definiciones se muestran a continuación:

Estilo arquitectónico: expresa la arquitectura en el sentido más formal y teórico, describiendo una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes. Una vez que se han identificado los estilos, es lógico y natural pensar en reutilizarlos en situaciones semejantes que se presenten en el futuro. (Reynoso, 2004)

[12] **IEEE**: por las siglas en inglés ***I**nstitute of **E**lectrical and **E**lectronics **E**ngineers*. Traducción: Insituto de Ingenieros Eléctricos y Electrónicos

Según (Buschmann, 2007) un patrón como concepto general es: *"una solución probada que se puede aplicar con éxito a un determinado tipo de problema que aparece con frecuencia".* De ahí que la definición propuesta por el propio Buschmann para el patrón arquitectónico sea la siguiente:

Patrón arquitectónico: expresa el esquema de organización estructural fundamental para sistemas de software. Este esquema provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. Pudiera representarse como una plantilla para arquitecturas de software concretas, que especifica las propiedades estructurales de una aplicación. (Buschmann, 2007)

1.6.2 Patrones de diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. (Buschmann, 1996) A continuación se describen las dos vertientes principales de estos patrones:

Los **Patrones GRASP**[13]: describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (Tedeschi, 2012)

- **Experto**: Asigna una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad. (Larman, 1999)
- **Creador**: Asigna a la clase B la responsabilidad de crear una instancia de clase A. (Larman, 1999)
- **Controlador**: Asigna la responsabilidad de administrar un mensaje de eventos del sistema a una clase. (Larman, 1999)
- **Bajo acoplamiento**: Asigna responsabilidades de modo que se mantenga bajo acoplamiento. (Larman, 1999)
- **Alta cohesión**: Asigna responsabilidad de modo que se mantenga alta cohesión al objetivo de las clases que lo posean. (Larman, 1999)

[13] **GRASP**: acrónimo de ***G****eneral* ***R****esponsibility* ***A****ssignment* ***S****oftware* ***P****atterns.* Traducción: Patrones Generales de Software para Asignación de Responsabilidades

Los **Patrones GoF**[14]: se dieron a conocer a principios de los años 90 con el libro "*Design Patterns. Elements of Reusable Object-Oriented Software*". Según su naturaleza se caracteriza en dicho libro a 23 patrones en los tres siguientes grupos:

- **Creacionales**: inicialización y configuración de objetos. (Prieto, 2009)
- **Estructurales**: separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes. (Prieto, 2009)
- **Comportamiento**: más que describir objetos o clases, describen la comunicación entre ellos. (Prieto, 2009)

1.7 Modelo de implementación

El modelo de implementación se inicia a partir de los resultados obtenidos del diseño y constituye la modelación de los aspectos físicos del sistema a desarrollar. En él se describe cómo se implementan en términos de componentes los elementos del modelo de diseño que constituyen los planos lógicos del sistema. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado. Las dependencias entre los componentes, así como los recursos necesarios para poder ejecutar la herramienta desarrollada son aspectos que también se puntualizan en este modelo. (Acuña, 2009)

1.7.1 Diagrama de componentes

En la modelación de los aspectos físicos de un sistema se realiza el diagrama de componentes, que muestra las dependencias entre un conjunto de componentes y se utiliza para modelar la vista de implementación estática. Gráficamente un diagrama de componentes es una colección de nodos y arcos, que normalmente puede contener: componentes e interfaces, así como relaciones de dependencia, generalización, asociación y realización. Además pueden contener paquetes o subsistemas, los cuales se utilizan para agrupar elementos del modelo en bloques mayores. (Jacobson, 1999)

1.7.2 Diagrama de despliegue

Es el otro diagrama que se realiza en la modelación de los aspectos físicos de un sistema, en conjunto con el diagrama de componentes. Este tipo de diagramas muestra la configuración de nodos que

[14] **GoF:** acrónimo de ***G**ang **o**f **F**our.* Traducción: Banda de los Cuatro.

participan en la ejecución y de los componentes que residen en ellos. Los diagramas de despliegue se utilizan para modelar la vista de despliegue estática de un sistema y son verdaderamente útiles para visualizar, especificar y documentar sistemas: empotrados, cliente/servidor y distribuidos. (Jacobson, 1999)

1.8 Métricas de software

Las métricas de software constituyen una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso de software. En esta evaluación se reúnen los datos básicos referentes a los factores de calidad que posteriormente serán analizados, comparados con promedios anteriores y evaluados, para determinar las mejorías en términos de éstos factores. (Pressman, 2005)

Además señala que las métricas serán utilizadas para la evaluación de determinados atributos de calidad. A continuación se muestran varios de estos atributos:

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta. (Baryolo, 2010)
- **Complejidad de implementación:** consiste en el grado de dificultad que tiene implementar un diseño de clases determinado. (Baryolo, 2010)
- **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software. (Baryolo, 2010)
- **Acoplamiento:** consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización. (Baryolo, 2010)
- **Complejidad del mantenimiento:** consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirectamente, pero fuertemente en los costes y la planificación del proyecto. (Baryolo, 2010)
- **Cantidad de pruebas:** consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, entre otras) diseñado. (Baryolo, 2010)

1.8.1 Tamaño operacional de clases (TOC)

Está dado por el número de métodos asignados a una clase. (Baryolo, 2010) La relación entre esta métrica y los atributos de calidad mencionados, se observa en la Tabla 2.

Tabla 2 Tamaño operacional de clase (TOC) (Baryolo, 2010)

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

1.8.2 Relación entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otra. (Baryolo, 2010) La relación entre esta métrica y los atributos de calidad mencionados, se observa en la Tabla 3.

Tabla 3 Relación entre clases (RC) (Baryolo, 2010)

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad de mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

1.9 Pruebas de Software

Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Dichas pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto. Los casos de prueba especifican una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, las condiciones bajo las que ha de probarse, así como los resultados esperados. (Baryolo, 2010) Por otra parte Pressman, asegura que además de detectar errores, las

pruebas tienen como objetivo medir el grado en que el software cumple con los requerimientos, además, expone que hay dos enfoques principales, las pruebas de *"caja blanca"* y las pruebas de *"caja negra"*.

1.9.1 Pruebas estructurales o de caja blanca

También conocidas como pruebas de *"caja de cristal"*, este es un método de diseño de casos de pruebas que utiliza la estructura de control del diseño procedimental para obtener los casos de pruebas que garanticen que: (Pressman, 2005)

- Se ejerciten al menos una vez todas las rutas independientes del módulo.
- Se ejecuten todas las decisiones lógicas en sus opciones verdadera y falsa.
- Se ejerciten todos los bucles en sus límites.
- Se usen las estructuras internas de datos para asegurar su validez.

Existen varias técnicas de pruebas de caja blanca, como se describen a continuación:

Camino básico: permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los casos de pruebas obtenidos garantizan que se ejecute al menos una vez cada sentencia del programa. (Pressman, 2005)

Prueba de condición: ejercita las condiciones lógicas contenidas en el módulo de un programa y su propósito es detectar no solo errores en las condiciones, sino también otro tipo de errores. (Pressman, 2005)

Pruebas de flujos de datos: selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variantes del programa. (Pressman, 2005)

Prueba de bucles: se centra exclusivamente en la validez de las construcciones de bucles. Se pueden definir cuatro clases diferentes de bucles: bucles simples, bucles concatenados, bucles anidados y bucles no estructurados. (Pressman, 2005)

1.9.2 Pruebas funcionales o de caja negra

Este tipo de pruebas, también conocidas como "*de comportamiento*", se concentran en los requisitos funcionales del software y descubren una clase diferente de errores de los que se descubrirán con los métodos de caja blanca. (Pressman, 2005) Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

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

Para realizar las pruebas de caja negra existen varias técnicas, algunas de ellas son:

Partición equivalente: es una técnica que divide el dominio de entrada de un programa en clases a partir de las cuales pueden derivarse casos de pruebas, permitiendo examinar los valores válidos e inválidos de estas entradas existentes en el software. Además descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Esto permite reducir el número de casos de prueba a elaborar. (Pressman, 2005)

Análisis de valores límites: los errores tienden a darse más en los límites del campo de entrada que en el centro. Esta es una técnica que complementa la partición equivalente y la mayorí de las veces se aplica de forma inconsciente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, lleva a elección de casos de prueba en los extremos de la clase. (Pressman, 2005)

Prueba de comparación: este tipo de pruebas se emplea cuando la fiabilidad del software es algo crítico, (por ejemplo cuando se desarrolla para aeronaves o plantas nucleares) varios equipos de ingeniería del software desarrollan versiones independientes de una misma aplicación, usando los mismos requisitos. Todas las versiones son probadas con datos iguales, para asegurar que todas proporcionan una salida idéntica. (Pressman, 2005)

Prueba de la tabla ortogonal: esta prueba puede aplicarse a problemas en que el dominio de entrada es relativamente pequeño pero demasiado grande para solicitar pruebas exhaustivas. El método de la tabla ortogonal es útil al encontrar errores asociados con fallos localizados. (Pressman, 2005)

1.10 Herramientas y tecnologías para el desarrollo

El desarrollo exitoso de los sistemas informáticos, depende en gran medida de una correcta elección de las herramientas y tecnologías a utilizar. Las definidas en la Vista Entorno de Desarrollo Tecnológico de Sauxe, se describen a continuación:

1.10.1 Lenguaje para el modelado

El **Lenguaje Unificado de Modelado** (UML[15]) es un leguaje de modelado estandarizado de propósito general en el campo de la ingeniería de software orientada a objetos. UML incluye un conjunto de técnicas de notación gráfica para crear modelos visuales de programación orientada a objetos. (Pressman, 2005)

Entre sus principales objetivos está: lograr que el modelado visual sea independiente del lenguaje de implementación, consiguiendo así que los diseños realizados usando UML puedan implementarse en cualquier lenguaje que soporte las posibilidades de este; así como establecer un lenguaje estándar de comunicación entre todas las personas involucradas en el proyecto.
UML representa para los desarrolladores de aplicaciones y sistemas una serie de ventajas, al igual que para las organizaciones, entre estos beneficios destacan: (ENTEL, 2013)

- Produce un aumento en la calidad del desarrollo.
- Mejora en un 50% o más los tiempos totales de desarrollo.
- Brinda la posibilidad de obtener un "plano" del sistema.
- Ofrece un mejor soporte a la planificación y control del proyecto.
- Constituye un lenguaje de modelado entendido tanto por humanos como por máquinas.
- Permite realizar una verificación y validación del modelo realizado.

[15] **UML:** por sus siglas en inglés ***U**nified **M**odeling **L**anguage*

- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y viceversa.

1.10.2 Lenguajes de programación

Los lenguajes de programación juegan un papel fundamental en la Informática: actúan como enlace entre los problemas a resolver (aplicaciones) y la máquina en que se resuelven (hardware) (Largo, 2009). Existe gran variedad de estos lenguajes, siendo utilizados convenientemente para el desarrollo de sistemas según sus principales características.

Programación del lado del servidor

PHP[16] 5.2.6 es un lenguaje interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser incrustado en páginas HTML[17]. La mayoría de su sintaxis es similar a los lenguajes C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los programadores páginas web más dinámicas de una manera rápida y fácil. (Gutmans, y otros, 2004) Este lenguaje de programación presenta muchas ventajas para los desarrolladores, dentro de las principales se pueden mencionar:

- Se caracteriza por ser un lenguaje muy rápido.
- Es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web.
- Compatibilidad con las bases de datos más comunes: PostgreSQL, MySQL, Oracle, Informix, entre otros.
- Presenta una gran potencia, un alto rendimiento, no consume muchos recursos y es seguro.

Programación del lado del cliente

Java Script 1.8 es un lenguaje de programación interpretado, utilizado principalmente en su forma del lado del cliente, permitiendo mejoras en la interfaz de usuario sobre todo en páginas web

[16] PHP: por sus siglas en inglés *Hypertext Preprocessor*
[17] HTML: por sus siglas en inglés HyperText Markup Language

dinámicas. Es un lenguaje interpretado que la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros lo soportan. (ECMA, 2010)

1.10.3 Herramientas utilizadas:

El **servidor web Apache 2.0** está desarrollado bajo el criterio Open Source. Funciona en múltiples sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable de diseño modular. Entre sus principales características se encuentran: (TheApache, 2013)

- Multiplataforma.
- Presenta una alta configuración en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.
- Se desarrolla de forma abierta.
- Extensible: gracias a que es modular se han desarrollado diversas extensiones entre las que destaca PHP.

El **gestor de base de datos PostgreSQL 9.0** es un sistema de gestión de bases de datos relacional con su código fuente disponible libremente. Utiliza un modelo cliente/servidor. Incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Otras características que posee: (Oracle, 2012)

- Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, entre otros.
- Soporte de protocolo de comunicación encriptado por SSL.
- PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. similar al de Oracle, PL/SQL.

El **PgAdminIII** es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados. Incluye una interfaz administrativa gráfica y una herramienta de consulta

SQL [18]. Es multiplataforma, es Software libre y se puede adquirir el código fuente desde la web oficial. Permite desde ejecución de consultas SQL simples, hasta la elaboración de bases de datos complejas, con el apoyo de las últimas características de PostgreSQL. No requiere ningún controlador adicional para comunicarse con el servidor de base de datos. La aplicación se encuentra desarrollada por una comunidad de especialistas en base de datos de todo el mundo y se encuentra disponible en más de 30 idiomas. (PgAdmin, 2011)

RapidSVN 1.6.6 es un cliente gráfico para Subversion. Es fácil de usar, tanto para los que conocen Subversion como para los que empiezan, pudiendo acceder a direcciones SVN, subir y descargar contenido y sincronizarlo con el servidor original, comprobar su estado, crear y fusionar direcciones. (López, 2010)

Subversion es una herramienta de código abierto, multiplataforma para el control de versiones de ficheros electrónicos, como son el software o la documentación. Se basa en un repositorio central que actúa como un servidor de ficheros, con la capacidad de recordar todos los cambios que se hacen tanto en sus directorios como en sus ficheros. El repositorio incrementa un número global de revisión con cada conjunto de cambios enviados (commit). Es posible copiar y renombrar ficheros; crear una rama del proyecto es tan fácil como copiar un directorio. También se puede pedir una salida con las diferencias entre dos revisiones arbitrarias, o que recupere algún sub-árbol de la revisión N. (Serradilla, 2008)

El **IDE NetBeans 7.0** es un entorno de desarrollo integrado de código abierto para desarrolladores de software. Cuenta con todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y aplicaciones móviles con la plataforma Java, así como con C / C + +, PHP, JavaScript y Groovy. NetBeans recibe el soporte integrado para lenguajes dinámicos, todo en una herramienta de gran alcance. El editor de PHP de NetBeans ofrece plantillas de código y generación (getters y setters), la refactorización, información sobre herramientas de parámetros, consejos y soluciones rápidas, y la finalización de código inteligente. El IDE también ofrece un completo editor de HTML, JavaScript y CSS, con la ventaja de proveer un resaltado de sintaxis completo, completamiento de código inteligente y la comprobación de errores para HTML, CSS y JavaScript, incluyendo HTML 5, JavaScript 1.7. (Oracle, 2012)

[18] **SQL**: por sus siglas en inglés Structured Query Language

1.10.4 Herramienta de Ingeniería de Software Asistida por Computación

Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computación (CASE[19]). Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Entre sus principales características se pueden mencionar: (Visual-Paradigm, 2005)

- Entorno de creación de diagramas para UML.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Disponibilidad de integrarse en los principales IDE.
- Generación de bases de datos.
- Importación y exportación de ficheros XML.

1.10.5 Frameworks y librerías para el desarrollo

Sauxe 2.2 es un marco de trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, reutiliza las siguientes tecnologías libres: (Baryolo, 2010)

Ext JS 4.2 es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas que además de flexibilizar el manejo de componentes de la página como el DOM[20], Peticiones AJAX[21], DHTML[22], tiene la gran funcionalidad de crear interfaces de usuario bastante funcionales, fáciles de usar y atractivas, similar a una aplicación de escritorio. Esto permite a los desarrolladores web concentrarse en la funcionalidad de las aplicaciones web en lugar de las advertencias técnicas y

[19] **CASE:** por sus siglas en inglés ***C**omputer **A**ssistant **S**oftware **E**ngineering*
[20] **DOM**: por sus siglas en inglés ***D**ocument **O**bject **M**odel*
[21] **AJAX**: por sus siglas en inglés ***A**synchronous **J**avaScript **A**nd **X**ML*
[22] **DHTML**: por sus siglas en inglés ***D**ynamic **HTML***

las incompatibilidades respecto a los navegadores. (Frederick, 2008)

Sus principales ventajas son: (Sencha, 2008)

- Permite crear aplicaciones complejas utilizando componentes predefinidos.
- Evita tener que validar el código para el correcto funcionamiento en cada uno de los navegadores (Firefox, Internet Explorer, Safari, Opera).
- El funcionamiento de las ventanas flotantes lo pone por encima de cualquier otro.
- Relación balanceada entre Cliente-Servidor: Se distribuye la carga de procesamiento, permitiendo que el servidor pueda atender más clientes al mismo tiempo.
- Eficiencia de la red: Disminuye el tráfico en la red pues las aplicaciones cuentan con la posibilidad de elegir qué datos desea trasmitir al servidor y viceversa.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información logrando una abstracción del cliente en el proceso.

Su utilización como eje principal en el desarrollo, se deriva de las numerosas ventajas y funcionalidades que aporta la librería, sobre todo del trabajo con peticiones asincrónicas al servidor. Sus características lo convierten en un potente framework, las cuales mejoran considerablemente la experiencia del usuario final de la aplicación, sin dejar de mencionar las herramientas que brinda para la creación de ventanas, formularios y validadores.

Zend Framework 1.11 se trata de un marco de trabajo para el desarrollo de aplicaciones web y servicios web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. (Zend-Technologies, 2006)

Algunas de las principales ventajas de este framework son: (Zend-Technologies, 2006)

- Ofrece un gran rendimiento.
- Posee una robusta implementación del patrón Modelo-Vista-Controlador.

- Cuenta con módulos para manejar archivos PDF[23], canales RSS, Web Services.
- Tiene una abstracción de base de datos fácil de usar, sin necesidad de escribir ninguna consulta SQL.
- Usa código 100% orientado a objetos aplicando modernas técnicas de diseño como el uso de patrones de diseño.

Doctrine 1.2.2 es un mapeador de objetos-relacional (ORM[24]) escrito en PHP, es una capa de abstracción que se sitúa justo encima de un SGBD (Sistema Gestor de Base Datos). Está completamente diseñado utilizando la técnica orientada a objetos, ofreciendo una capa de persistencia transparente a los objetos de PHP. Uno de sus puntos fuertes es que cuenta con su propio lenguaje de consultas a bases de datos llamado DQL[25], basado en el dialecto HQL[26] de Hibernate. (Doctrine, 2006)

Doctrine brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa. Una de las ventajas que ofrece el uso de Doctrine, es que le evita al desarrollador escribir consultas SQL con sintaxis específicas para un sistema de gestión de bases de datos concreto, para ello transforma de forma automática las llamadas a los objetos en consultas optimizadas para el tipo de sistema que se esté utilizando. (Doctrine, 2006)

[23] **PDF**: por sus siglas en inglés Professional Document File
[24] **ORM:** por sus siglas en inglés ***O**bject **R**elational **M**apper*
[25] **DQL**: por sus siglas en inglés ***D**octrine **Q**uery **L**anguage*
[26] **HQL**: por sus siglas en inglés ***H**ibernate **Q**uery **L**anguage*

1.11 Conclusiones del capítulo

La revisión bibliográfica permitió conocer y caracterizar los conceptos fundamentales relacionados con la situación problemática, como la interfaz gráfica de usuario, la personalización y los temas de interfaz, entre otros que contribuyen a un mejor entendimiento del contexto.

El análisis de varias soluciones informáticas para la edición y el diseño de interfaces, arrojó como resultado que debido a sus especificidades, ninguna de ellas puede ser integrada al marco de trabajo Sauxe, requisito fundamental a tener en cuenta para definir su viabilidad en el contexto de la solución. Sin embargo, teniendo en cuenta sus características específicas y analizando por separado los indicadores definidos: su estudio sí aporta una buena base para la elaboración de la solución en cuanto a las formas de realizar modificaciones en tiempo de ejecución, así como el tratamiento de la usabilidad en dichos sistemas.

El análisis del Modelo de desarrollo de software del CEIGE promovió la comprensión de sus principales características y sus fases en pos de guiar el posterior diseño de la solución. De igual forma, el estudio de las herramientas y tecnologías definidas por el departamento para el desarrollo de sus productos, garantiza un buen punto de partida en la construcción de la herramienta para la personalización de los temas de la capa de presentación del marco trabajo Sauxe.

Capítulo 2: Propuesta de solución

En el presente capítulo se realiza una propuesta de solución que describe la herramienta para la personalización de interfaces de usuario en el marco de trabajo Sauxe. Se realiza una descripción de los procedimientos que definen el funcionamiento de la herramienta. Se muestran además los diferentes artefactos generados durante las fases tanto de Requisitos como de Análisis y diseño.

2.1 Descripción de la solución

Para facilitar la personalización de los temas de la capa de presentación, se decidió implementar una herramienta que permita disminuir el tiempo de personalización de las interfaces. Esta herramienta formará parte integrante del marco de trabajo Sauxe, posibilitando que todas las interfaces del marco de trabajo se puedan modificar de forma sencilla, por medio del editor. Desde la interfaz de la herramienta se podrá acceder a cada elemento de la capa de presentación del marco de trabajo y modificar sus atributos. Posibilitando la personalización de uno o varios elementos de los que componen el tema de la capa de presentación: Ventana de autenticación; Entorno de escritorio comercial; Entorno de escritorio estilo Windows; y Entrada al sistema. En la propia interfaz de la herramienta se mostrará una vista previa en la cual se podrán observar los colores, íconos, imágenes y tipografías seleccionados durante la modificación de estos elementos.

Para las modificaciones visuales en tiempo de ejecución, se realizan los cambios en el lado del servidor, sin embargo, no existe un mecanismo definido para realizar dichas modificaciones, por lo que aplicado al contexto de Sauxe, se definió un procedimiento que se utilizará en la personalización. Este procedimiento se ajusta a las características de los lenguajes utilizados en el desarrollo y se nutre de los algoritmos creados para darle solución a la problemática.

2.1.1 Procedimiento para hacer los cambios en tiempo de ejecución

Los datos son recogidos en la interfaz gráfica mediante un editor avanzado. Este proporciona la vista previa de los cambios que se van realizando en el momento. Luego de su verificación son enviados al servidor para su procesamiento. El servidor recibe la nueva configuración del tema de interfaz actual y mediante el ASTS[27], cambia los respectivos elementos en el archivo y los reemplaza por los nuevos. Este

[27] **ASTS:** siglas de ***A****lgoritmo de* ***S****ustitución de* ***T****extos* ***S****ignificativos*

mecanismo permite que a cada usuario asociado a este tema de interfaz gráfica, le sean aplicados los cambios seleccionados. Su punto clave es que el cliente (navegador) realiza la petición de los archivos fuentes para el renderizado de la interfaz, y estos ficheros se retornan al cliente con los nuevos cambios incluidos dentro de ellos.

2.1.2 Algoritmo de Sustitución de Textos Significativos

La rutina de sustitución de textos en el fichero CSS representa el corazón de la solución. El lenguaje PHP al contrario de JavaScript no permite el manejo de la configuración CSS en forma de objetos, debido a que por su propia condición de ser un lenguaje del lado del servidor, no tiene acceso a los objetos de configuración CSS, creados por el intérprete del navegador. Tampoco define ningún mecanismo para la construcción de estos objetos del lado del servidor, sino que lee directamente los archivos fuentes CSS, debido a que el lenguaje no fue creado para este propósito.

De esta manera el acceso a la configuración visual contenida en estos archivos, queda limitada a la lectura en modo texto mediante los recursos que provee el lenguaje, para la apertura y manejo de ficheros. La modificación de una propiedad o regla, dentro de un archivo CSS, se lleva a cabo haciendo la detección de las reglas en el fichero, y luego ser remplazadas por las reglas con las nuevas propiedades.
El ASTS construye objetos de cada una de las reglas CSS definidas, parseando todo el archivo. Estos objetos contienen funcionalidades que permitirán el acceso y modificación de cada una de las propiedades definidas en las reglas CSS. La sintaxis definida para las acciones de acceso y modificación están basadas en las sentencias DQL, por lo que la hace de fácil entendimiento. Posee (integrado además a la sintaxis) el tratamiento de condicionales, basado también en DQL, que permitirá la ejecución de la modificación definida en la sintaxis, si el grupo de condiciones se cumplen. Las ventajas que aporta al desarrollo se evidencian en las siguientes características:

- No importa cuán compleja o extensa sea la condicional para el ASTS, la declaración de la regla siempre se realiza de forma secuencial, por lo que se eliminan los bloques condicionales típicos del lenguaje PHP y se facilita el entendiendo del código.
- El ASTS es genérico, garantizando la reutilización para futuras extensiones del componente, o simplemente para su utilización en otro tipo de sistema.
- Su implementación está basada en las funciones básicas definidas por PHP, lo que permite su uso en cualquier entorno que utilice este lenguaje.

Las funciones utilizadas en la implementación de este algoritmo son las siguientes:

- **Función de reemplazo en PHP**

La función *str_replace* de PHP, reemplaza todas las apariciones de la cadena buscada con la cadena de reemplazo. (Olson, 2009)

> *Descripción*
>
> **str_replace** (`$search` , `$replace` , `$subject`): Esta función devuelve un string o un array con todas las apariciones de `search` en `subject` reemplazadas con el valor dado de `replace`. (Olson, 2009)

Es utilizada para el reemplazo de una regla en su conjunto, ya que cada objeto de configuración CSS construido, guarda en modo texto la declaración original en el archivo CSS, lo que permitirá el uso de esta función.

- **Función de división de textos en PHP**

La función *split* de PHP, divide una cadena en un arreglo mediante una expresión regular. (Olson, 2009)

> *Descripción*
>
> **split** (`$pattern` , `$string` , `$limit`): Esta función devuelve un array de una cantidad de elementos igual a `limit`, donde cada uno de ellos es una sub-cadena de `string` formada al dividir éste con los límites establecidos mediante la expresión regular contenida en `pattern`. (Olson, 2009)

Es utilizada para la detección de las reglas que se someterán al ASTS, y para la formación/creación de las propiedades del objeto de configuración CSS.

2.2 Modelo conceptual

El modelo conceptual podría considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio. Mediante un lenguaje visual se pueden representar las cosas del mundo real que son de interés para el dominio de forma más sencilla y entendible. (Larman, 1999)

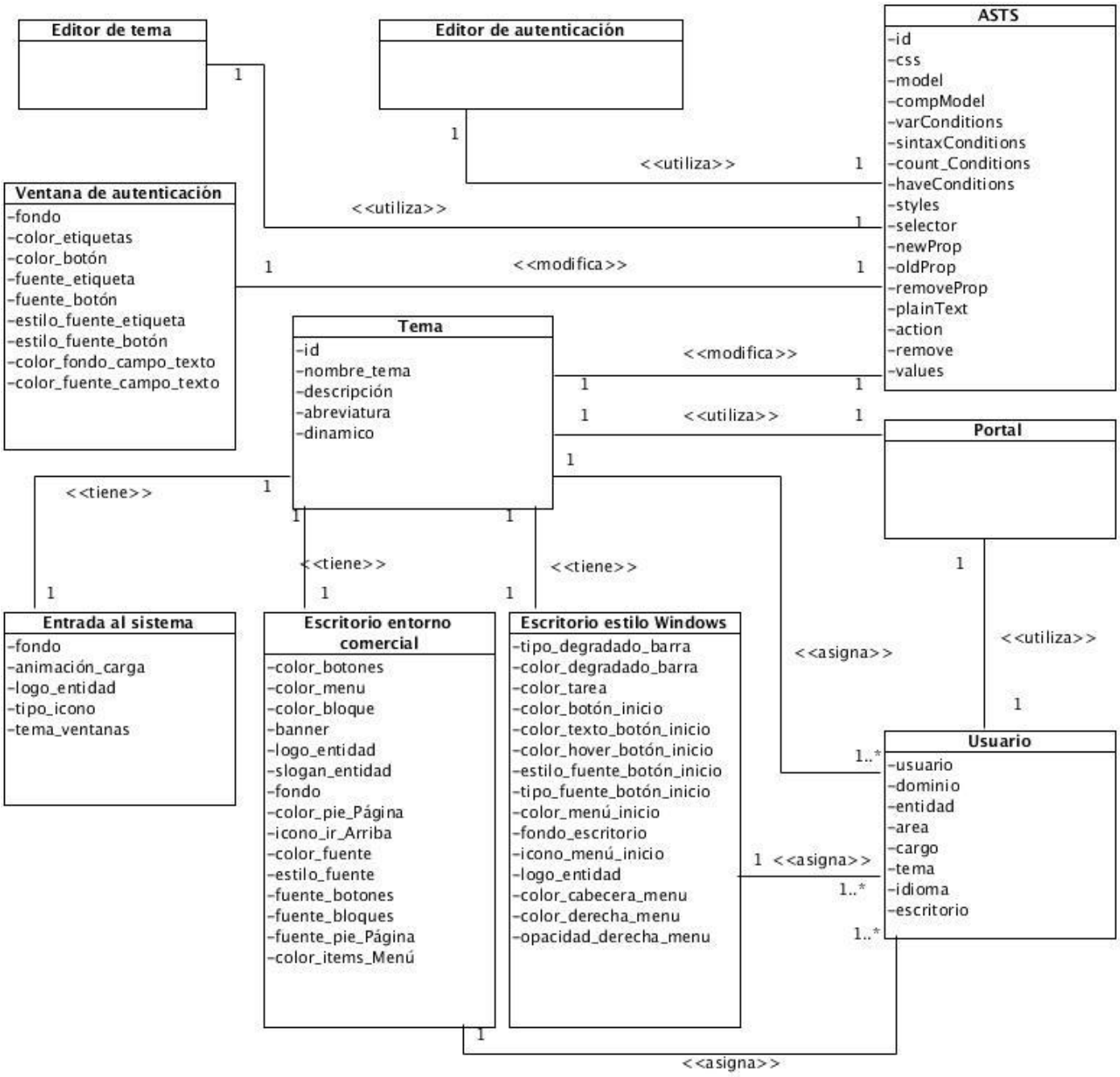


Figura 2 Modelo Conceptual

A continuación se describen los principales conceptos representados en el modelo conceptual perteneciente a la solución:

Tema: Agrupa el conjunto de configuraciones visuales de la capa de presentación del marco de trabajo Sauxe.

Escritorio Estilo Windows: configuración visual perteneciente al entorno de escritorio que semeja la apariencia visual del sistema operativo Windows.

Escritorio Entorno Comercial: configuración visual perteneciente al entorno de escritorio que semeja la apariencia visual de un sitio web.

Entrada al sistema: configuración visual perteneciente a la ventana de entrada al sistema.

Ventana de autenticación: configuración visual perteneciente a la ventana de autenticación.

ASTS: Algoritmo que modifica los atributos visuales del tema y de la ventana de autenticación.

Usuario: persona que interactúa con el sistema y desempeña un rol determinado en este.

Portal: componente que define el comportamiento del diseño de la capa de presentación de Sauxe.

Editor de tema: define los atributos que se van a modificar en el tema mediante el uso del ASTS.

Editor de autenticación: define los atributos que se van a modificar en la ventana de autenticación mediante el uso del ASTS.

2.3 Requisitos de Software

2.3.1 Captura de requisitos

El equipo de desarrollo debe determinar las técnicas que se deberán usar según el producto a desarrollar, una opción factible para realizar este proceso es combinar varias de estas técnicas. Las utilizadas para el desarrollo de la solución son: Entrevista, Tormenta de ideas y Observación.

2.3.2 Requisitos funcionales

Los requisitos funcionales identificados en el proceso suman un total de 30 requisitos, incluidos en las 6 agrupaciones de requisitos como se muestra en la siguiente tabla.

Tabla 4 Listado de requisitos funcionales

Agrupación de requisitos	Nombre del requisito
Gestionar tema	RF 1.1 Adicionar tema dinámico

	RF 1.2 Eliminar tema dinámico
	RF 1.3 Listar temas
Construir entorno de escritorio comercial	RF 2.1 Renderizar entorno de escritorio comercial
	RF 2.2 Listar funcionalidades del sistema en el menú de navegación
	RF 2.3 Cargar interfaces de las funcionalidades del sistema
	RF 2.4 Listar datos del usuario conectado
	RF 2.5 Cambiar contraseña del usuario conectado
	RF 2.6 Cerrar sesión del usuario conectado
Configurar tema de interfaz de usuario para el entorno de escritorio Estilo Windows	RF 3.1 Modificar tema dinámico para el entorno de escritorio Estilo Windows
	RF 3.2 Cargar logo de la entidad del entorno Estilo Windows
	RF 3.3 Cargar imagen de fondo del entorno Estilo Windows
	RF 3.4 Cargar ícono del menú de inicio del entorno Estilo Windows
	RF 3.5 Crear vista previa del entorno Estilo Windows
Configurar tema de interfaz de usuario para el Entorno comercial	RF 4.1 Modificar tema dinámico para el Entorno comercial
	RF 4.2 Cargar banner del Entorno comercial
	RF 4.3 Cargar imagen de fondo del Entorno comercial
	RF 4.4 Cargar slogan de la entidad del Entorno comercial
	RF 4.5 Crear vista previa del Entorno comercial
Configurar tema de interfaz de usuario para la Entrada al sistema y opciones generales	RF 5.1 Modificar tema dinámico para la Entrada al sistema
	RF 5.2 Listar temas de Ext JS
	RF 5.3 Listar juego de íconos del sistema
	RF 5.4 Importar íconos del sistema
	RF 5.5 Cargar logo de la entidad para la Entrada al sistema
	RF 5.6 Cargar imagen de fondo para la Entrada al sistema
	RF 5.7 Cargar animación para la Entrada al sistema
	RF 5.8 Crear vista previa para la Entrada al sistema
Configurar ventana de autenticación	RF 6.1 Editar ventana de autenticación
	RF 6.2 Cargar imagen de fondo de la ventana de autenticación
	RF 6.3 Crear vista previa de la ventana de autenticación

Las descripciones de requisitos de la agrupación **Configurar tema de interfaz de usuario para el entorno de escritorio Estilo Windows** se pueden observar en los Anexos (1-5) de la versión digital del documento. En el expediente del proyecto Sauxe se recogen todas las descripciones de requisitos del sistema.

2.3.3 Requisitos no funcionales

El desarrollo de la herramienta para la personalización forma parte del marco de trabajo Sauxe, desarrollado en el CEIGE. Por tanto, y de acuerdo con los requisitos que fueron establecidos por el centro al iniciar el proceso de desarrollo, los requisitos no funcionales a los que se debe acoger la aplicación a desarrollar son los que se describen a continuación:

Usabilidad

El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

Rendimiento

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Seguridad

Autenticación (Contraseña de acceso.)

Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

La atención al sistema incluyendo, el mantenimiento de las bases de datos así como la salva de la información se realizarán de forma centralizada por el administrador.

Verificación sobre las acciones irreversibles (eliminaciones).

Portabilidad

El sistema debe ser multiplataforma.

Soporte

La aplicación contará antes de su puesta en marcha con un período de pruebas para detectar posibles errores, se brindará el servicio de instalación, se garantizará la configuración inicial y se le dará mantenimiento una vez instalada.

2.3.4 Validación de requisitos

Para lograr una correcta visualización de lo que se desea implementar en cada requisito y permitir que los clientes valoren si a través de éstos se satisfacen sus necesidades se utiliza la técnica de validación de los requisitos: Construcción de prototipos. Dentro de cada descripción de requisitos se podrá observar el prototipo de interfaz correspondiente al requisito que se esté especificando.

2.4 Modelo de Diseño

El Modelo de diseño ha sido definido como *"una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño*". (Baryolo, 2010) El modelo de diseño puede contener: diagramas, clases, paquetes, interfaces, entre otros, que son utilizados como entrada esencial en las actividades relacionadas a la implementación. En el presente epígrafe se muestran los elementos que se tuvieron en cuenta para el diseño de la solución, así como los resultados y artefactos generados durante esta parte del proceso de desarrollo.

2.4.1 Arquitectura de Software

Para el desarrollo del marco de trabajo Sauxe, en el Departamento de Tecnologías del CEIGE se definió el **Estilo N capas**, con cuatro capas fundamentales que se describen a continuación: (Baryolo, 2010)

- **Capa de presentación**: esta es la capa encargada de elaborar y mostrar las interfaces de los usuarios. La capa de presentación está compuesta principalmente por el marco de trabajo ExtJS y centra su desarrollo en tres componentes fundamentales archivos JS, archivos CSS, páginas clientes y páginas servidoras.
- **Capa de control o negocio:** en esta capa se encuentra ubicado el marco de trabajo Zend, el cual implementa el patrón Modelo-Vista-Controlador.
- **Capa de acceso a datos:** está ubicado el marco de trabajo Doctrine, como ORM para la comunicación con el servidor de datos mediante el protocolo PDO[28].
- **Capa datos:** en esta capa se encuentran ubicado los servidores de base de datos y los archivos XML donde se almacena la información de la aplicación.

En la capa de control o negocio de la arquitectura descrita anteriormente para Sauxe, se implementa el patrón arquitectónico MVC[29], el cual se encarga de separar los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos: (Baryolo, 2010)

- **Modelo:** está compuesto por datos, reglas de negocio y las funcionalidades correspondientes para la comunicación con el encargado de persistir los datos Doctrine.
- **Controlador:** gestiona las entradas y las respuestas del sistema al usuario.
- **Vista:** muestra la información del modelo al usuario.

[28] **PDO:** Protocolo de conexión a base de datos basado en objetos
[29] **MVC**: patrón arquitectónico ***M**odelo **V**ista **C**ontrolador*

2.4.2 Patrones de diseño

En la implementación del sistema se utilizan patrones de diseño que pertenecen a las dos vertientes GRASP y GoF como se describe a continuación:

Patrones GRASP:

- **Experto**: Se evidencia el uso de este patrón en todas las clases a utilizar en el componente pues cada clase conoce su información y es la encargada de implementar las funcionalidades que les corresponde como por ejemplo la clase GestnomtemaController.
- **Creador**: Su uso se evidencia en la clase controladora pues es la que se encarga de la creación de objetos de varias clases, como son NomPropDesktopModel, NomPropComercial, entre otras.
- **Controlador**: En el sistema la clase GestnomtemaController se encarga de llevar el control de todos los eventos relacionados con el negocio. Implementa las funcionalidades que dan respuesta a las peticiones del usuario.
- **Bajo acoplamiento**: El uso de este patrón se evidencia en la poca relación existente entre las clases que conforman la herramienta, por ejemplo en la clase NomTema.
- **Alta cohesión**: El uso de éste patrón indica que la información almacenada en las clases debe ser coherente y relacionada a lo que se maneja en esas clases. Se evidencia su uso en todas las clases como por ejemplo en GestnomtemaController.

Patrones GoF:

Estrategia/Strategy: Este patrón pertenece al grupo de *Comportamiento* y su propósito es definir una familia de algoritmos encapsulando por separado cada uno de ellos y haciéndolos, por tanto, intercambiables. Esto permite a los algoritmos variar con independencia de los clientes que los usan. Suele ser usado cuando un algoritmo usa datos que el cliente no tiene por qué conocer o cuando muchas clases relacionadas sólo se diferencian en su comportamiento. (UDIS, 2004) El uso de este patrón se evidencia entre las clases CSSHandler y ComercialCSSBase.

Apoderado/Proxy: Este patrón es de tipo *Estructural* y su intención es proporcionar un representante o delegado que se encargue de controlar el acceso a un objeto, generalmente por motivos de eficiencia. Su uso es aconsejado cuando se necesita una referencia a un objeto más sofisticada y versátil que un simple puntero. (UDIS, 2004) Este patrón se pone de manifiesto entre las clases CSSHandler y AstsRule.

2.4.3 Diagrama de clases del diseño con estereotipos web

Los diagramas de clases especifican las diferentes clases que serán utilizadas en el sistema, así como las relaciones que existen entre ellas. A continuación se muestra el diagrama de clases del diseño para el RF 3.1 Modificar tema dinámico para el escritorio estilo Windows. El diagrama de clases general y el del RF 2.1 Construir entorno de escritorio comercial, se pueden observar en los Anexos 6 y 7.

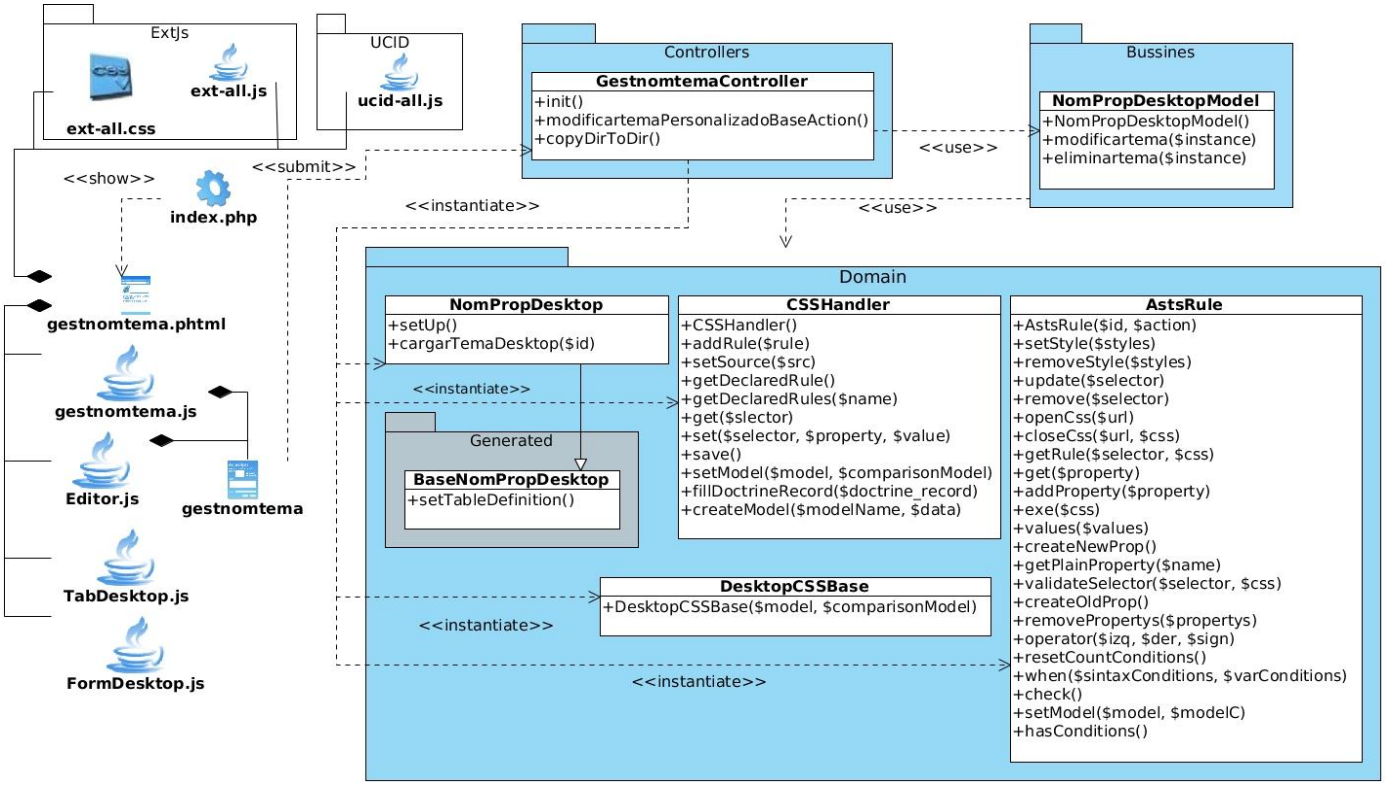


Figura 3. Diagrama de clases del diseño con estereotipos web para el requisito 3.1

2.4.4 Diagrama de secuencia

Los diagramas de secuencia describen la interacción entre los objetos de una aplicación y los mensajes recibidos y enviados por los objetos. (ALTOVA, 2014) A continuación se muestra el diagrama de secuencia correspondiente al requisito funcional **RF3.3 Cargar imagen de fondo para el entorno de escritorio estilo Windows.** El resto de los diagramas de secuencia pueden ser consultados en los modelos de diseño del expediente de proyecto.

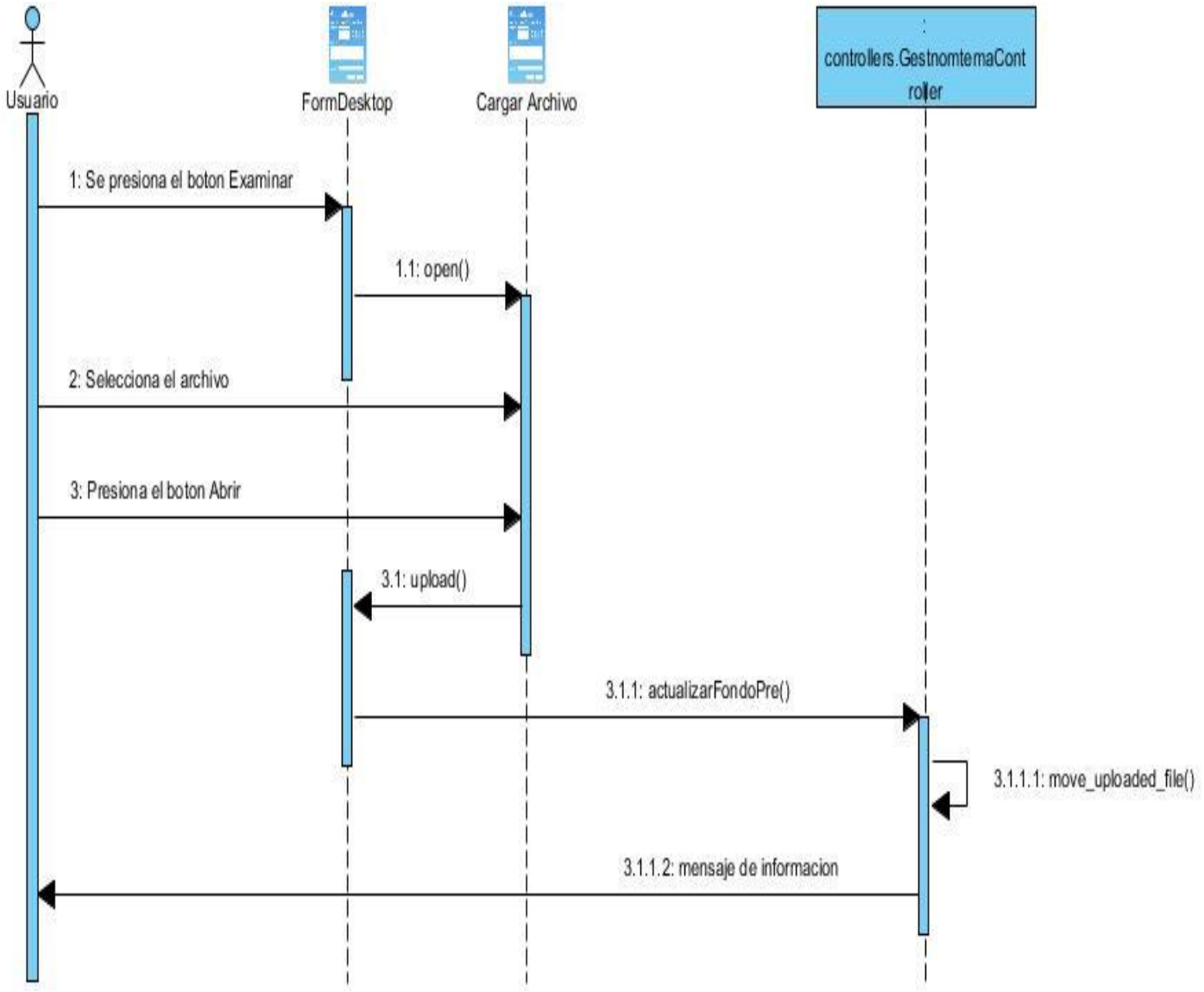


Figura 4. Diagrama de secuencia RF 3.3 Cargar imagen de fondo para el entorno de escritorio estilo Windows

2.4.5 Modelo de datos

El modelo de datos es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos permiten modelar la estructura de los datos y los operadores permiten modelar su comportamiento. (Date, 2001) El modelo de datos correspondiente a la herramienta contiene cinco tablas que se muestran a continuación.

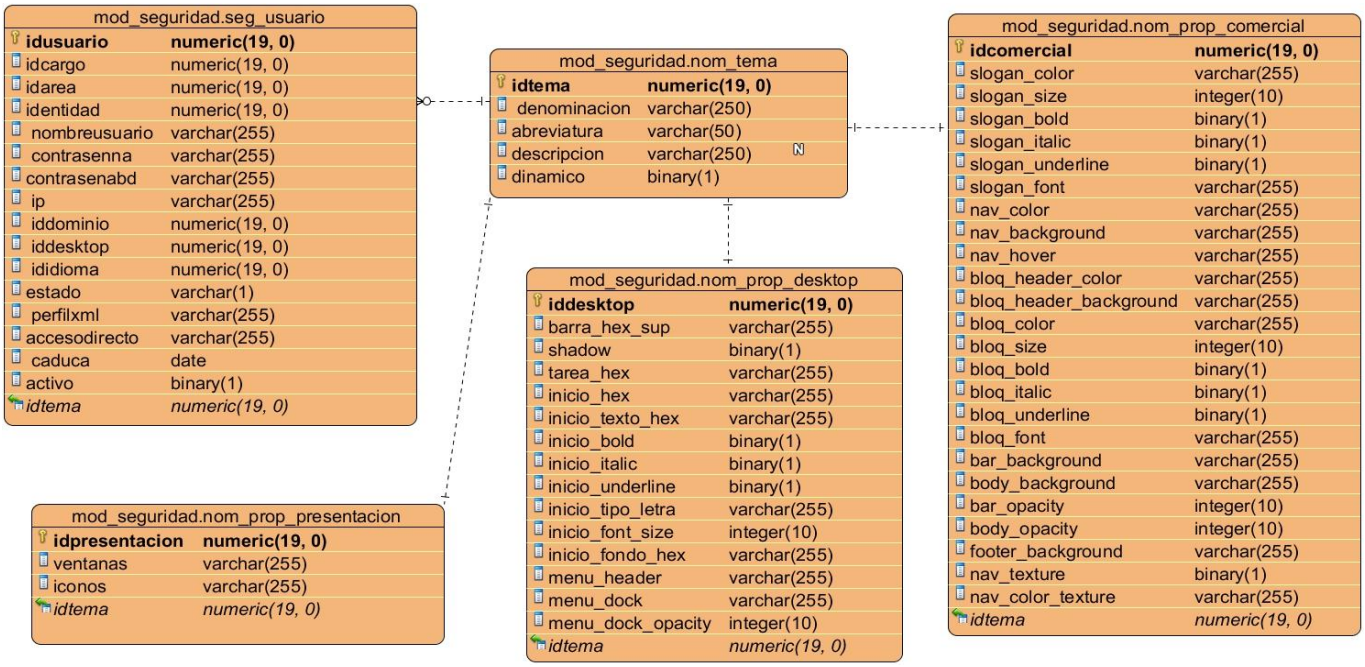


Figura 5. Modelo de entidad-relación

A continuación se describen los atributos de la tabla **mod_seguridad.nom_prop_desktop:**

iddesktop: llave primaria de la tabla.

barra_hex_sup: color de la barra superior en formato hexagecimal

shadow: booleano que determina si la barra de tareas es sombreada.

tarea_hex: color de las pestañas en la barra de tareas en formato hexagecimal.

inicio_hex: color del botón de inicio de la barra de tareas en formato hexagecimal.

inicio_texto_hex: color del texto del botón de inicio en formato hexagecimal.

inicio_bold: booleano que determina si el texto del botón de inicio tiene estilo de fuente negrita.

inicio_italic: booleano que determina si el texto del botón de inicio tiene estilo de fuente cursiva.

inicio_underline: booleano que determina si el texto del botón de inicio tiene estilo de fuente subrayado.

inicio_tipo_letra: tipo de letra del botón de inicio.

inicio_font_size: tamaño de la fuente del botón de inicio.

inicio_fondo_hex: color del evento hover del botón de inicio en formato hexagecimal.

menu_header: color de fondo de la cabecera del menú de inicio en formato hexagecimal.

menu_dock: color de fondo del panel derecho del menú de inicio en formato hexagecimal.

menu_dock_opacity: opacidad del panel derecho del menú de inicio.

idtema: llave foránea que determina a qué tema pertenece.

2.5 Conclusiones del capítulo

Para realizar los cambios visuales en tiempo de ejecución, se utiliza un Algoritmo de Sustitución de Textos Significativos elaborado por los desarrolladores, que proporciona un procedimiento estructurado para la variante de cambios en el lado del servidor, cuya representación establece la base de una reingeniería adecuada.

El modelo conceptual mostrado en el capítulo puntualiza los conceptos relacionados con el dominio del problema y su representación conforma una guía visual para la implementación de la solución tributando directamente a la fase de requisitos.

La captura, especificación y validación de los 30 requisitos funcionales mostrados en seis agrupaciones de requisitos generales, aportó una explicación escrita del problema y una mayor comprensión del comportamiento que se necesita implementar en la solución. Los requisitos no funcionales que debe cumplir la herramienta, garantizan que su desempeño es el adecuado para su integración en Sauxe.

La utilización de estilos y patrones arquitectónicos brinda solidez a la arquitectura, garantizando que la estructura propuesta rija de forma correcta el posterior diseño, a la vez que los patrones de diseño especificados, permiten ganar en reutilización y brindan robustez a la solución.

La realización del diagrama de clases del diseño con estereotipos web, los diagramas de secuencia y el modelo de datos, permitió representar: las clases y relaciones que componen el sistema; el flujo del comportamiento e interacción entre los objetos de la aplicación; y la estructura que tendrán los datos de la herramienta respectivamente, de forma tal que a través de estos diagramas se facilitó el entendimiento previo al proceso de implementación.

Capítulo 3: Implementación y pruebas

El siguiente capítulo aborda los elementos relacionados con la implementación de la Herramienta para la personalización de temas de la capa de presentación del marco de trabajo Sauxe, así como lo referente a las pruebas realizadas al sistema para validar su correcto funcionamiento. En el desarrollo se describen los estándares de codificación utilizados en función de garantizar el entendimiento del código fuente. Se muestran los casos de pruebas diseñados para realizar las pruebas funcionales, así como los resultados obtenidos, validando así que los requisitos identificados fueron implementados correctamente.

3.1 Modelo de implementación

3.1.1 Diagrama de componentes

Para modelar los aspectos físicos de la Herramienta para la personalización de los temas de la capa de presentación del marco de trabajo Sauxe se utiliza el siguiente diagrama de componentes:

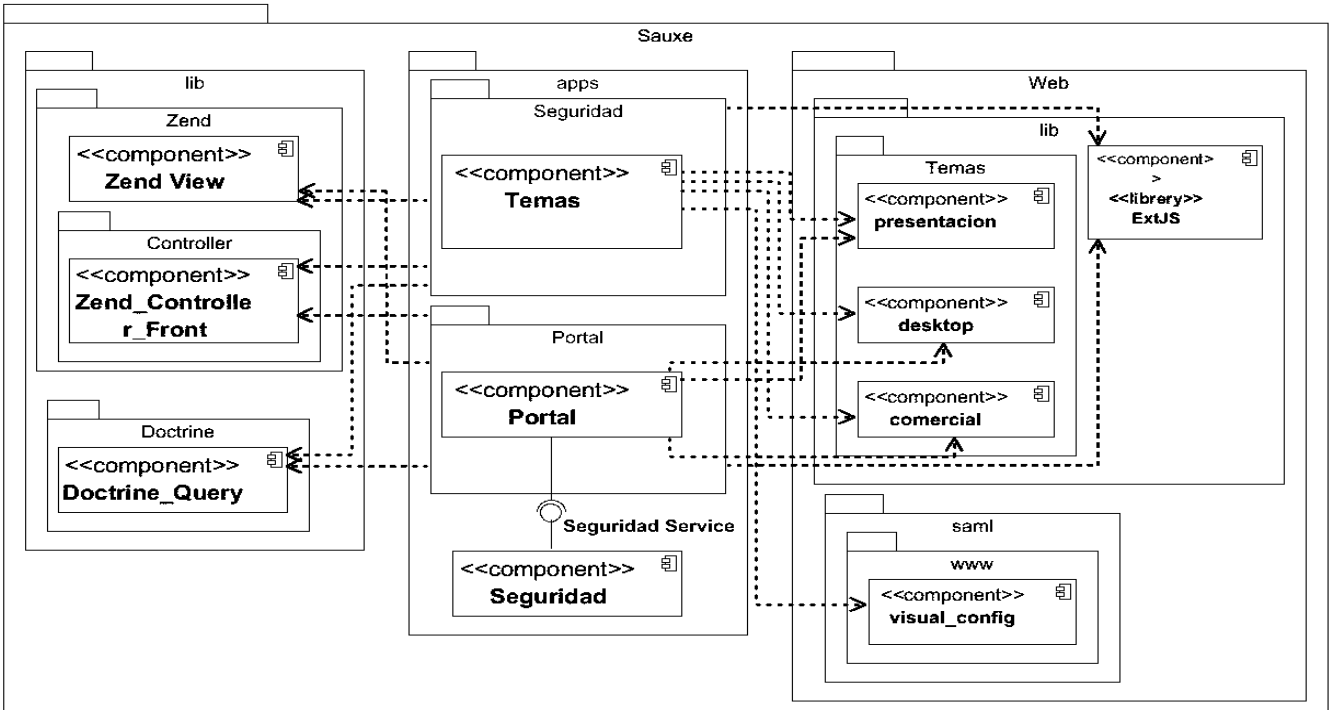


Figura 6 Diagrama de componentes

3.1.2 Diagrama de despliegue

Para modelar la vista de despliegue estática del marco de trabajo Sauxe (sistema al cual se integra la solución) se utiliza el diagrama de despliegue siguiente:

Figura 7 Diagrama de despliegue del marco de trabajo Sauxe

3.2 Estándares de codificación

Un factor importante para la implementación lo constituye la forma en que se construye el código fuente, esencialmente su organización y el estilo de codificación utilizado en el desarrollo. Una buena estructuración del código, mejora la lectura del software, permitiendo entender código nuevo rápidamente y más a fondo. (Lobo, 2012)

El uso de estándares de codificación permite lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo. (Pérez Alfonso, 2012) Para el desarrollo de la solución, se utilizarán algunos de los estándares y normas de codificación propuestos como parte de la Línea de Arquitectura determinada para el proyecto ERP Cuba, los cuales se muestran a continuación:

- **Notación Húngara:** definir prefijos para cada tipo de datos y según el ámbito de las variables. La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que indique su tipo de dato o ámbito. (Sperberg, 2012)

- **Notación PascalCasing:** es como la notación húngara pero sin prefijos. En este caso los identificadores y nombres de las variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula. (Svensk, 2012)

- **Notación CamelCasing:** es parecido al PascalCasing con la excepción que la letra inicial del identificador no debe estar en mayúscula. (Svensk, 2012)

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing. Con sólo leerlo se reconoce su propósito. Ejemplo: NomTema.

1. **Nomenclatura según el tipo de clases:**
   - **Clase controladora**: después del nombre llevan la palabra "Controller". Ejemplo: GestnomtemaController.

2. **Clases del modelo:**
   - **Business (Negocio):** las clases que se encuentran dentro de business después del nombre llevan la palabra "Model". Ejemplo: NomTemaModel.
   - **Domain (Dominio):** el nombre que reciben las clases que se encuentran dentro de domain es el de la tabla en la base de datos. Ejemplo: DatTema.
   - **Generated: e**l nombre de las clases que se encuentran dentro de generated comienza con el prefijo "Base" y continúa con el nombre de la tabla en la base de datos. Ejemplo: BaseNomTema.

Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing, y en caso de ser una acción de la clase controladora se debe especificar el nombre de dicha acción en minúscula y seguido el sufijo "Action". Ejemplo: modificartemaAction.

Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing. Ejemplo: $tema.

Normas de comentariado

Se debe comentar todo lo que se haga dentro del desarrollo, establecer las pautas que conlleven a lograr un código más legible y reutilizable y así se pueda aumentar su mantenimiento a lo largo del tiempo.

Estilo del código

En la implementación, al escribir las sentencias en php, se utilizarán los tabs del lenguaje como se muestra en la Figura 8:

```php
<?php
//código
?>
```

Figura 8 Estilo del código

- **Sangría o indexado:** La política de sangría a utilizar en la implementación es por **tab**. Ejemplo:

```php
<?php
/**
 * Indentation
 */
class Example {
    var $theInt = 1;
    function foo($a, $b) {
    switch ( $a) {
    case 0 :
    $Other->doFoo ();
    break;
    default :
    $Other->doBaz ();
    }
    }
    function bar($v) {
    for($i = 0; $i < 10; $i ++) {
    $v->add ( $i );
    }
    }
}
?>
```

Figura 9 Estilo del código: Sangría o indexado

- **Brazas o llaves:** En la declaración de clases o interfaces, métodos, bloques y switch, la apertura de llaves se hace en la misma línea. Ejemplo:

```php
<?php
/**
 * Braces
 */
interface EmptyInterface {
}

class Example {
    function bar($p) {
        for($i = 0; $i < 10; $i ++) {
        }
        switch ( $p) {
            case 0 :
                $fField->set ( 0 );
            break;
            case 1 :
                {
                    break;
                }
            default :
                $fField->reset ();
        }
    }
}
?>
```

Figura 10 Estilo del código: Brazas o llaves

3.3 Métricas de software

Según (Pressman, 2005), la medición permite tener una visión del proceso de software, pues proporciona un mecanismo para lograr una evaluación objetiva. La calidad de un sistema, depende directamente de los requisitos que detallan el problema, del diseño que modela la solución, del código ejecutable del

programa y de las pruebas que se ejecutan al software para detectar errores. Por tanto, se considera de vital importancia evaluar con medidas técnicas la calidad del diseño en la presente investigación y se utilizan las métricas TOC y RC.

3.3.1 Resultados de la aplicación de la métrica TOC al diseño

La Figura 11 muestra los resultados obtenidos de la aplicación de la métrica TOC al diseño. Estos resultados se agrupan en los intervalos representados en la gráfica. El gráfico refleja que la mayoría de las clases tienen de uno a cinco procedimientos.

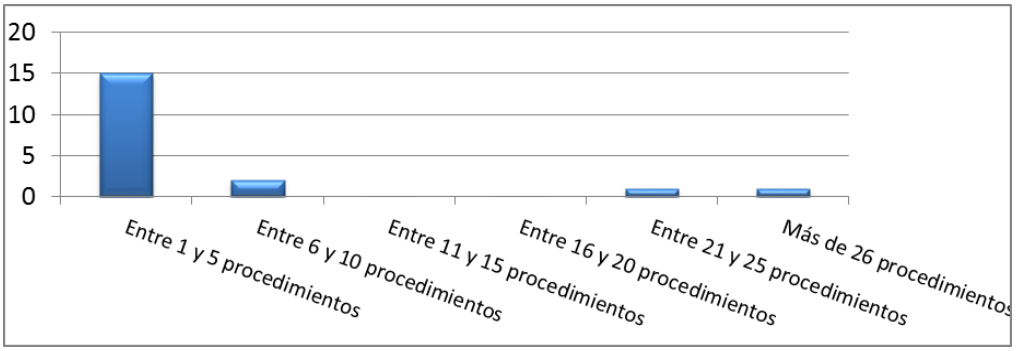


Figura 11 Representación de la evaluación de la métrica TOC

Este resultado demuestra que el funcionamiento general del sistema está distribuido equitativamente entre la mayoría de las clases, aunque es necesario analizar las que contienen la mayor cantidad de procedimientos para tratar de restarle funcionalidades y distribuirlas entre las demás clases para evitar que alguna sea demasiado crítica y que en caso de fallo sea menor el número de funcionalidades que queden fuera de servicios.

En la Figura 12 se observa la representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad. La gráfica muestra un resultado satisfactorio ya que el 69 % de las clases poseen una baja responsabilidad.

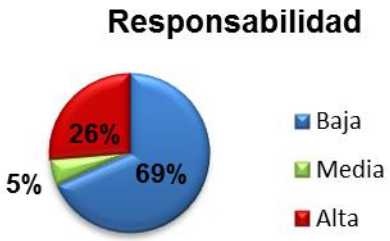


Figura 12 Representación incidencia de la métrica TOC en el atributo Responsabilidad

La Figura 13 muestra el resultado de la incidencia de la métrica TOC en el atributo Complejidad de Implementación (izquierda). Este gráfico muestra un resultado satisfactorio, pues el 69 % de las clases poseen una baja complejidad de implementación, característica que permite mejorar el mantenimiento y soporte de estas clases. Además la figura muestra en su parte derecha el resultado de la incidencia de la métrica TOC en el atributo Reutilización. Este gráfico muestra que el diseño de la herramienta tiene un grado aceptable pues solo el 26 % del total de las clases posee una baja reutilización.

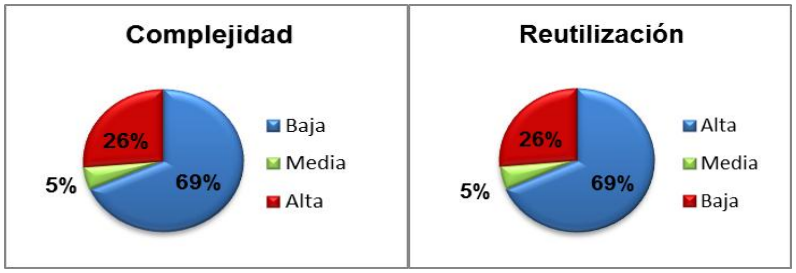


Figura 13 Resultados evaluación de la métrica TOC en: Complejidad de implementación; Reutilización.

Teniendo en cuenta los resultados de la aplicación de la métrica TOC al diseño, se puede señalar que en general tiene una calidad aceptable, pues el mayor por ciento del total de las clases posee baja responsabilidad, baja complejidad de implementación y una alta reutilización. Los instrumentos y las tablas de resultados del instrumento de medición de la métrica TOC se pueden observar en el Anexo 8 de la versión digital del documento.

3.3.2 Resultados de la aplicación de la métrica RC

La Figura 14 muestra los resultados obtenidos en el instrumento de evaluación de la métrica RC en porcientos agrupados en los intervalos definidos. El gráfico refleja que el 95 % de las clases tienen entre 1 y ninguna dependencia con otra clase, demostrando que casi el total de las clases presentan niveles aceptables de calidad.

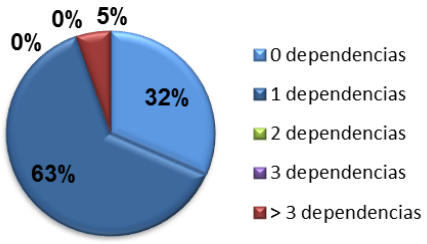


Figura 14 Representación de los resultados obtenidos en los intervalos definidos según la métrica RC

La Figura 15 muestra la representación de la incidencia de los resultados de la evaluación de la métrica RC en los atributos: Acoplamiento (izquierda) y Complejidad de mantenimiento (derecha). En el gráfico de la izquierda se evidencia un diseño eficiente al quedar reflejado que el 63 % de las clases presenta bajo acoplamiento, mientras que el 32 % no presenta dependencias con otras clases, lo que aumenta el grado de reutilización del componente. En el gráfico de la derecha el resultado es favorable ya que el 89 % de las clases presenta baja complejidad de mantenimiento, dándole respaldo a una buena calidad de la arquitectura.

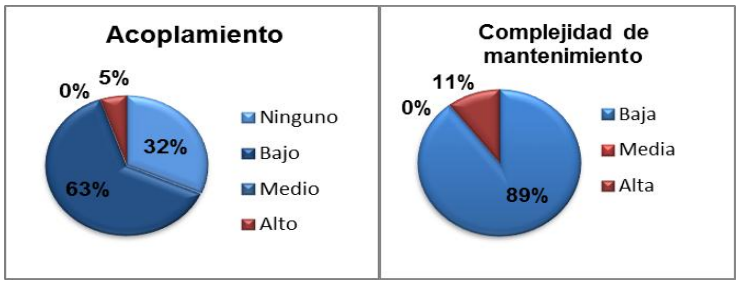


Figura 15 Resultados evaluación de la métrica RC para: Acoplamiento; Complejidad de mantenimiento

La Figura 16 muestra la representación de la incidencia de la métrica RC en los atributos: Cantidad de pruebas (izquierda) y Reutilización (derecha). El gráfico de la izquierda demuestra que la mayor cantidad de las clases no necesita una cantidad elevada de pruebas según las dependencias entre las clases. Por su parte el gráfico de la derecha muestra que el diseño tiene un 95 % de reutilización de sus clases, constituyendo un factor importante en el desarrollo del software.

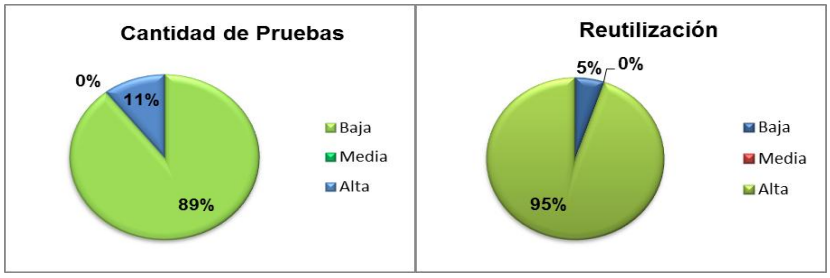


Figura 16 Resultados evaluación de la métrica RC para: Cantidad de pruebas; Reutilización

Analizando los resultados de la evaluación del instrumento de medición de la métrica relación entre clases (RC), se puede concluir que el diseño posee una calidad aceptable, ya que el 95 % de las clases de la solución tienen menos de 2 dependencias con otras clases. También se refleja en los resultados que en el 95 % el acoplamiento entre clases es mínimo; que la complejidad de mantenimiento y la cantidad de

pruebas son bajas en un 89 %; y que la reutilización se comporta en un valor elevado del 95 %. Este resultado demuestra que los atributos de calidad se encuentran en niveles satisfactorios. Los instrumentos y las tablas de resultados del instrumento de medición de la métrica RC se pueden observar en el Anexo 9 de la versión digital del documento.

3.4 Pruebas de Software

3.4.1 Resultados de la aplicación de la prueba de caja blanca

La técnica utilizada para realizar las pruebas de caja blanca a la Herramienta para la personalización de temas es la del **camino básico**. A continuación se muestra el grafo generado durante la aplicación de esta técnica al código de la funcionalidad **exe()** perteneciente al ASTS. El fragmento del código utilizado puede observarse en el Anexo 11 de la versión digital del documento.

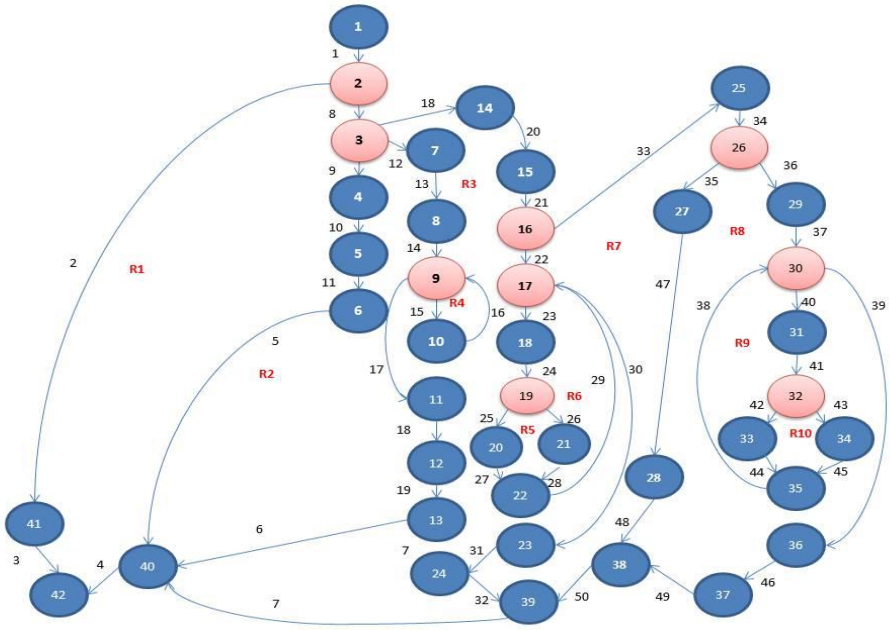


Figura 17 Grafo del camino básico asociado a la funcionalidad exe()

La complejidad ciclomática es la métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Esta métrica calcula la cantidad de caminos independientes de cada una de las funcionalidades del programa. También provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. El cálculo de la complejidad ciclomática se realiza después de construir el grafo y para ello se utilizan tres fórmulas que aseguran que esta métrica se ha calculado correctamente si coinciden sus resultados. (Pressman, 2005)

1. **V(G)=R** donde **R** representa la cantidad total de regiones.
   V(G)=10
2. **V(G)=A–N+2** donde **A** es el número de aristas del grafo de flujo y **N** es el número de nodos.
   V(G)=50–42+2
   V(G)=10
3. **V(G)=P+1** donde **P** es el número de nodos predicado contenidos en el grafo de flujo (se denomina nodo predicado a los nodos de los cuales parten dos o más aristas).
   V(G)=9+1
   V(G)=10

Posterior al cálculo de las fórmulas mencionadas, se pudo constatar que las tres dan el mismo resultado. Por tanto se puede afirmar que la complejidad ciclomática del método **exe()** tiene un valor de 10. Este valor significa que existen 10 caminos posibles para el flujo del método:

Camino básico # 1: 1–2–41–42
Camino básico # 2: 1–2–3–4–5–6–40–42
Camino básico # 3: 1–2–3–7–8–9–10–9–11–12–13–40–42
Camino básico # 4: 1–2–3–7–8–9–10–11–12–13–40–42
Camino básico # 5: 1–2–3–14–15–16–17–18–19–20–22–17–23–24–39–40–42
Camino básico # 6: 1–2–3–14–15–16–17–23–24–39–40–42
Camino básico # 7: 1–2–3–14–15–16–25–26–27–28–38–39–40–42
Camino básico # 8: 1–2–3–14–15–16–25–26–29–30–36–37–38–39–40–42
Camino básico # 9: 1–2–3–14–15–16–25–26–29–30–31–32–33–35–30–36–37–46–49–38–39–40–42
Camino básico # 10: 1–2–3–14–15–16–25–26–29–30–31–32–34–35–30–36–37–46–49–38–39–40–42

Además, este valor representa la mínima cantidad de casos de prueba para el procedimiento, teniendo en cuenta que se realiza un caso de prueba por cada camino básico:

Caso de prueba para el camino básico #1: Si $this->check()==false

Caso de prueba para el camino básico #2: Si $this->check()==true, Si $this->action()==Asts::REMOVE

Caso de prueba para el camino básico #3: Si $this->check()==true, Si $this->action()==Asts::CREATE, Si $this->newProp>0

Caso de prueba para el camino básico #4: Si $this->check()==true, Si $this->action()==Asts::CREATE, Si $this->newProp==0

Caso de prueba para el camino básico #5: Si $this->check()==true, Si $this->action()==Asts::UPDATE, Si $this->newProp>0, Si $this->remove==false, Si $plainProp==’ASTS-NO-EXIST-PROPERTY‘

Caso de prueba para el camino básico #6: Si $this->check()==true, Si $this->action()==Asts::UPDATE, Si $this->newProp==0, Si $this->remove==false

Caso de prueba para el camino básico #7: Si $this->check()==true, Si $this->action()==Asts::UPDATE, Si $this->remove==true, Si $this->removeProp==0

Caso de prueba para el camino básico #8: Si $this->check()==true, Si $this->action()==Asts::UPDATE, Si $this->remove==true, Si $this->removeProp!=0, S $this->newProp==0

Caso de prueba para el camino básico #9: Si $this->check()==true, Si $this->action()==Asts::UPDATE, Si $this->remove==true, Si $this->removeProp!=0, S $this->newProp>0, Si count($aux)>1

Caso de prueba para el camino básico #10: Si $this->check()==true, Si $this->action()==Asts::UPDATE, Si $this->remove==true, Si $this->removeProp!=0, S $this->newProp>0, Si count($aux)<=1

Con la realización de esta prueba se garantizó que todas las sentencias del programa se han ejecutado al menos una vez.

3.4.2 Resultados de la aplicación de las pruebas funcionales o de caja negra

Para realizar este tipo de pruebas, se utilizó la técnica de Partición equivalente para la cual se confeccionaron los diseños de casos de pruebas por cada funcionalidad del sistema, los cuales pueden ser observados en el expediente del proyecto Sauxe. Para comprobar la calidad de la Herramienta para la personalización de temas, se realizaron dos iteraciones de revisiones por el grupo de calidad del CEIGE. Las no conformidades encontradas se clasificaron en:

- No conformidades detectadas en la aplicación.
- No conformidades detectadas en la documentación.

En la Tabla 5 se muestra la cantidad de no conformidades detectadas en la aplicación y la documentación por cada iteración. De igual forma en la Figura 20 se muestran estos resultados de forma gráfica.

Tabla 5 Resultados de las pruebas funcionales por cada iteración

Tipo de no conformidades	Pruebas exploratorias	Primera iteración	Segunda iteración
Aplicación	8	1	0
Documentación	27	0	0
Total	35	1	0

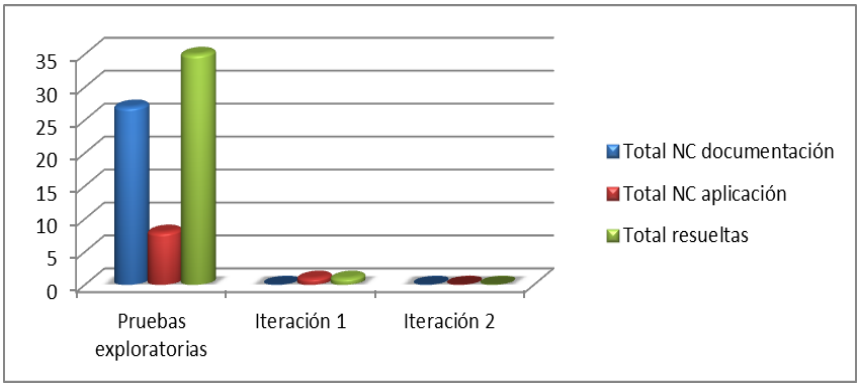


Figura 18 Representación de los resultados de las pruebas funcionales

3.5 Validación de la investigación

Con el objetivo de validar la propuesta se escogieron cuatro miembros del proyecto Marco de trabajo para el desarrollo de aplicaciones web de gestión, para realizar una personalización en dos maneras diferentes: de forma manual y con la herramienta. Para lograr una estimación del tiempo de personalización se realizó un cuestionario con dos preguntas abiertas, el cual se muestra en el Anexo 10. Las respuestas a estas preguntas indicaron los tiempos de realización de cada una de las tareas pertenecientes a la personalización, que está conformada por la edición de cualquiera de los siguientes elementos visuales, modificando directamente los archivos fuentes de sus respectivas configuraciones:

- Ventana de autenticación. Tiempo de edición correspondiente = **Ah**.
- Entorno de escritorio comercial. Tiempo de edición correspondiente = **Ch**.
- Entorno de escritorio estilo Windows. Tiempo de edición correspondiente = **Wh**.
- Entrada al sistema. Tiempo de edición correspondiente = **Ph**.

Para dar cumplimiento a la tarea se realizó la edición de estos cuatro elementos, asignados de forma individual a cada uno de los implicados, quienes ejecutaron la personalización en forma manual y con la herramienta, en aras de cumplimentar en conjunto la personalización del sistema. Los resultados obtenidos en el cuestionario se muestran a continuación, en ellos el tiempo de duración de estas tareas estará representado en horas (8h cada jornada laboral).

Tabla 6 Resultados del cuestionario aplicado

	Ah	Ch	Wh	Ph	Total
De forma manual	5	12	8	8	33
Con la herramienta	0,15	0,4	0,32	0,1	1,37

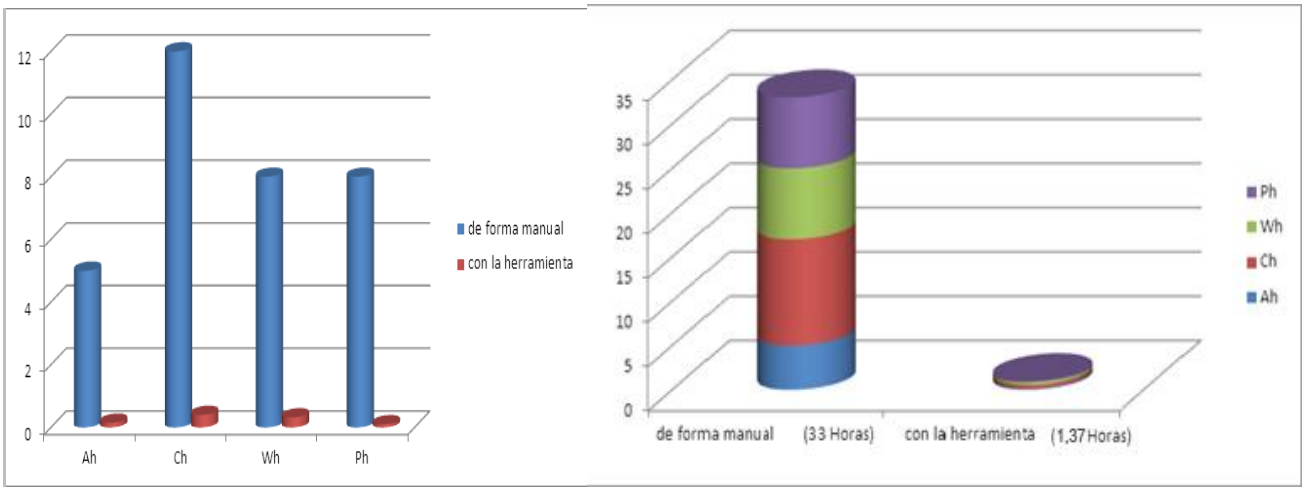


Figura 19 Representación gráfica de los resultados del cuestionario

Antes de aplicada la propuesta, se puede observar que la duración de cada tarea varía en dependencia de su complejidad, siendo la edición del entorno comercial la de mayor duración con 12 horas. Luego de aplicada la propuesta se observa una disminución del tiempo de personalización de cada tarea.

Los resultados del cuestionario revelan además que de forma manual, la personalización tiene una duración aproximada de 33 horas (cuatro jornadas laborales), para una sola persona. Mientras que haciendo uso de la herramienta, solo una duración de 1,37 horas, para una reducción del **95,85%** del tiempo de personalización que se dedicaba antes de aplicada la propuesta.

La personalización de forma manual fue cumplimentada en 12 horas (una jornada laboral y media), debido a que las cuatro tareas se realizaron simultáneamente. Sin embargo, una sola persona puede realizar esta tarea en solo 1,37 horas, con lo que se disminuye el tiempo de personalización y además se reduce la cantidad de recursos humanos a utilizar.

3.6 Validación de la Herramienta para la personalización de temas

Para la validación de la herramienta se tuvieron en cuenta tres indicadores: 1) factores de calidad, 2) el alcance de los requisitos y 3) la aceptación del cliente.

Factores de calidad

Funcionalidad: La herramienta mantiene un conjunto apropiado de funciones para las tareas y los objetivos del usuario especificados, permitiendo el manejo de archivos e imágenes. Además permite proteger la información y los datos, utilizando claves de autenticación para cada usuario y concediéndole acceso solo a las funciones que le estén permitidas.

Fiabilidad: La herramienta es capaz de detectar los tipos de error, recuperarse y notificar al usuario el tipo de error ocurrido, manteniendo así todas sus funcionalidades.

Usabilidad: La herramienta le permite al usuario entender si el software es idóneo, y cómo puede usarse para las tareas, especificando en las etiquetas de cada funcionalidad y los campos de cada interfaz, títulos asociados a su función de negocio. El uso de iconografía apropiada y de nombres de cada interfaz de no más de 30 caracteres le posibilitan al usuario aprender a utilizar el software. No se utilizan más de tres interfaces para lograr una funcionalidad completa, y se diferencian los mensajes de información de los mensajes de error utilizando íconos para cada tipo, brindándole a la herramienta una interfaz atractiva.

Mantenibilidad: La utilización de los estilos y patrones arquitectónicos, garantizó la flexibilidad, permitiendo agregar nuevas funcionalidades o modificar alguna existente sin romper la estructura y consistencia de los componentes.

Portabilidad: La solución puede ser utilizada en los navegadores más utilizados y ejecutado bajo cualquier sistema operativo.

Alcance de los requisitos

Del total de 30 requisitos de software identificados, se logró implementar el 100 %, dando cumplimiento así a todas las especificaciones planteadas por los clientes.

Aceptación del cliente

Con el objetivo de validar que la herramienta cumple las características planteadas por los clientes o especialistas, se creó una comisión en el proyecto Marco de trabajo para el desarrollo de aplicaciones web de gestión, encargada de someter la solución a una revisión técnica. Esta determinó que la herramienta está estable y lista para su uso. La comisión emitió el acta de liberación correspondiente, reflejada en el Anexo 12, evidenciando que la herramienta cumple con todas las expectativas planteadas.

3.7 Conclusiones del capítulo

El diagrama de componentes permitió representar mediante una colección de arcos y nodos, los aspectos físicos del sistema a desarrollar.

Las normas y estándares de codificación utilizados permitieron brindarle legibilidad y escalabilidad al código fuente.

La evaluación aplicada al diseño mediante las métricas TOC y RC, evidencia niveles satisfactorios como el 69 % de baja responsabilidad y el 95 % de reutilización de las clases del sistema, así como los niveles de los restantes indicadores evaluados.

Las pruebas estructurales realizadas al código a partir de la técnica del camino básico, evidencian una correcta implementación, que garantizó que todas las sentencias del programa se ejecutaron al menos una vez.

Los resultados de las pruebas funcionales, en las que con dos iteraciones totales se logró dar respuesta a las no conformidades detectadas, acreditan la correcta puesta en práctica de los requisitos funcionales.

El pre experimento realizado evidencia la disminución en un 95,85 % del tiempo de personalización de los temas de la capa de presentación de Sauxe, dándole cumplimiento a los objetivos específicos y al objetivo general propuesto para la investigación.

Conclusiones generales

Para lograr un correcto ajuste a la identidad de las distintas instituciones que utilicen el marco de trabajo Sauxe en el desarrollo de sus productos, se hace necesario el uso de temas personalizados de interfaz de usuario. El análisis de varias soluciones informáticas para la edición y el diseño de interfaces, arrojó como resultado que debido a sus especificidades, ninguna de ellas puede ser integrada al marco de trabajo Sauxe, aunque su estudio sí aporta una buena base para la elaboración de la solución necesaria.

El modelo conceptual permitió puntualizar los conceptos relacionados con el dominio del problema y su representación conforma una guía visual que tributa directamente a la fase de requisitos. La captura, especificación y validación requisitos funcionales, aportó una explicación escrita del problema y una mayor comprensión del comportamiento que se necesita implementar en la solución. Los requisitos no funcionales que debe cumplir la herramienta, garantizan que su desempeño es el adecuado para su integración en Sauxe.

La utilización de estilos y patrones arquitectónicos brinda solidez a la arquitectura, garantizando que la estructura propuesta rija de forma correcta el posterior diseño, a la vez que los patrones de diseño especificados, permiten ganar en reutilización y brindan robustez a la solución.

La evaluación aplicada al diseño mediante las métricas TOC y RC, evidencia niveles satisfactorios como el 69 % de baja responsabilidad y el 95 % de reutilización de las clases del sistema, así como los niveles de los restantes indicadores evaluados.

Las pruebas de software realizadas a la solución evidencian una correcta implementación y puesta en práctica de los requisitos funcionales.

El pre experimento realizado evidencia una disminución en un 95,85 % del tiempo de personalización de los temas de la capa de presentación de Sauxe. El tiempo de duración de la tarea se redujo en 31,63 h dándole cumplimiento a los objetivos específicos y al objetivo general propuesto para la investigación.

Recomendaciones

En función de mejorar la solución se recomienda:

- Extender la herramienta con funcionalidades que permitan el diseño de interfaces en el marco de trabajo.

Referencias bibliográficas

Acuña, Karenny Brito. 2009. Selección de metodologías de desarrollo para aplicaciones web. [En línea] 2009. http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm.

ALTOVA. 2014. Diagramas de secuencia UML. *ALTOVA.* [En línea] 2014. [Citado el: 5 de Enero de 2014.] http://www.altova.com/es/umodel/sequence-diagrams.html.

Artisteer. 2008. Artisteer. *Sitio Web oficial de Artisteer.* [En línea] 2008. [Citado el: 10 de Febrero de 2014.] http://www.artisteer.com.

Baryolo, Oiner Gómez. 2010. *Solución informática de autorización en entornos multientidad y multisistema.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2010. Tesis de Maestría.

Buschmann, F, Kevlinn Henney, Douglas C. Schmidt. 2007. *Pattern-Oriented Software Architecture: A pattern language for distribuited computing. Tennessee: J ohn Wiley & Sons.* 2007. 978-0-470-05902-9.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. 1996. *Pattern–Oriented Software Architecture. A System of Patterns. John Wiley & Sons.* Inglaterra : s.n., 1996.

Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. 2004. *Arquitecturas de software. Guía de estudio.* 2004.

Date, C. J. 2001. *Introducción a los sistemas de bases de datos.* México : Pearson Educación, 2001. 968-444-419-2.

Doctrine. 2006. Doctrine. *Doctrine-Project.* [En línea] 2006. [Citado el: 15 de Enero de 2014.] http://docs.doctrine-project.org/projects/doctrine1/en/latest/en/manual/introduction.html#what-is-doctrine.

ECMA. 2010. JavaScript Introduction. *W3Schools.com.* [En línea] 2010. [Citado el: 11 de Diciembre de 2013.] http://www.w3schools.com/js/js_intro.asp..

ENTEL. 2013. UML: Un Lenguaje Modelo. *CIENTEC. Grupo ENTEL.* [En línea] 2013. [Citado el: 12 de Enero de 2014.] http://www.cientec.com/analisis/ana-uml.html.

Expósito, Carlos Marrero. 2008. Interfaz gráfica de usuario: Aproximación semiótica y cognitiva. *Mentexpansiva.* [En línea] 2008. [Citado el: 12 de Enero de 2014.] http://www.chr5.com/investigacion/investiga_igu/index_igu.html..

Frederick, Shea and Ramsay, Colin. 2008. *Learning Ext JS. Birmingham.* s.l. : Packt Publishing Ltd, 2008. ISBN 978-1-847195-14-2.

Ganesh, Gunda Sai. 2008. *Requirements Engineering: Elicitation Techniques.* Suecia : Universidad del Este, Departamento de Tecnología, Matemática y Ciencia de la Computación, 2008. PR003..

Gordo, Marc Serra. 2007. Usabilidad de una biblioteca online con diseño centrado en el usuario. [En línea] 2007. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/31061/6/mserragoTFC0707memoria.pdf.

Grau, R, Correa, C y Rojas, M. 2004. *Metodología de la investigación.* s.l. : Segunda edición. Universidad de Ibagüé., 2004.

Gutmans, Andi, Sæther, Bakken Stig y Derick., Rethans. 2004. PHP 5 Power Programing. Professional Technical Reference. [En línea] 2004. [Citado el: 26 de Enero de 2014.] http://eva.uci.cu/file.php/106/Bibliografia_Basica/PHP_5_Power_Programming.rar.

IEEE. 2007. *Introducción a la Ingeniería de Requisitos. Ingeniería de software. .* Estados Unidos : s.n., 2007.

ISDI. 2008. *El diseño de signos identificadores. Bibliografía básica. Departamento de diseño de comunicación visual.* La Habana, Cuba : Instituto Superior de Diseño, 2008.

Jacobson, Ivar, Grady Booch, James Rumbaugh. 1999. *El Lenguaje Unificado de Modelado.* s.l. : Addison-Wesley Professiona, 1999.

Kim, W. 2002. *Personalization: Definition, status, and challenges ahead.* s.l. : Journal, 2002.

Largo, Faraón Llorens, y otros. 2009. *Capítulo 1: Lenguajes de Programación. Programación: Formalización, Análisis y Reutilización de Algoritmos Matemáticos.* s.l. : Computers & Applied Science, 2009.

Larman, Craig. 1999. 1999. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : Alhambra S. A, 1999. ISBN: 970-17-0261-1./8420534382.

León, R. A. H. y González, Z. C. 2008. *El paradigma Cuantitativo de la Investigación Científica.* La Habana, Cuba : Editorial Universitaria, 2008. 978-959-16-9343-2.

Lobo, Armando Robert. 2012. 2012. *Lycan-Génesis,Component Builder. Manual del Desarrollador. Centro DATEC.* La Habana. Cuba : Facultad 6, Universidad de las Ciencias Informáticas., 2012.

Lobo, Armando Robert. 2010. *Lycan IDE. Documento Arquitectura del Software Centro DATEC.* La Habana. Cuba : Facultad 6, Universidad de las Ciencias Informáticas, 2010.

López, José María. 2010. Softonic. Cliente para Subversion para principiantes y expertos. *Softonic.* [En línea] 2010. [Citado el: 2 de Febrero de 2014.] http://rapidsvn.softonic.com.

Martín, Adriana Elba. 2003. *Personalización de aplicaciones web. Un enfoque de reingeniería.* Argentina. : Universidad Nacional de la Plata, 2003.

Martínez, Rafael. 2010. Sobre PostgreSQL. Introducción. *PostgreSQL-es.* [En línea] 2010. [Citado el: 8 de Diciembre de 2013.] http://www.postgresql.org.es/sobre_postgresql.

Moreno, Luciano. 2005. Componentes de una interfaz web. *Desarrolloweb.com.* [En línea] 2005. [Citado el: 10 de Diciembre de 2013.] http:// www.desarrolloweb.com/articulos/2171.php..

Obregón, William González. 2013. *Modelo de desarrollo de software v1.2.* Centro de la Informatización de la Gestión de Entidades, Universidad de las Ciencias Informáticas. 2013.

Olson, Philip. 2009. Manual PHP. Función split. *PHP.* [En línea] 2009. [Citado el: 3 de Febrero de 2014.] http://www.php.net/manual/es/function.split.php.

Olson, Philip. 2009. Manual PHP. Función str_replace. *PHP.* [En línea] 2009. [Citado el: 5 de Febrero de 2014.] http://www.php.net/manual/es/function.str-replace.php.

Oracle, Corporation. 2012. FOSS License Exception. *MySql.* [En línea] 2012. [Citado el: 5 de Diciembre de 2013.] http://www.mysql.com/about/legal/licensing/foss-exception/.

Pérez Alfonso, Damián. 2012. *Normas y estándares de codificación de Sauxe.* La Habana,Cuba : Universidad de las Ciencias Informáticas, 2012.

PgAdmin. 2011. Excelente gestor PostGreSQL. *Puro Software.* [En línea] 2011. [Citado el: 8 de Diciembre de 2013.] http://www.purosoftware.com/programacion-bases-de-datos/11-pg-admin-3.htm.

Pressman, Roger S. 2005. *Ingenería del Software. Un enfoque práctico.* s.l. : Quinta Edición.McGraw Hil, 2005.

Prieto, Félix. 2009. *Patrones de diseño. .* España : Departamento de Informática. Universidad de Valladolid, 2009.

Qt-Project. 2013. Manuales. *Sitio Web oficial de Qt.* [En línea] 2013. [Citado el: 20 de Febrero de 2014.] http://qt-project.org/doc/qt-4.8/designer-manual.html.

Reynoso, B. 2004. *Architect Academy: Seminario de Arquitectura de Software.* 2004.

Sencha. 2008. Sencha. Designer. *Sitio Web oficial de Sencha Ext Js.* [En línea] 2008. [Citado el: 15 de Diciembre de 2013.] http://www.docs.sencha.com/designer/.

Sencha. 2008. 2008. What is Sencha Ext JS? *sitio web de Sencha.* [En línea] 2008. [Citado el: 1 de Diciembre de 2013.] http://www.sencha.com/products/extjs/.

Serradilla, Juan Luis. 2008. Control de Versiones con Subversion y TortoiseSVN. *ATICA. Unicversidad de Murcia.* [En línea] 2008. [Citado el: 3 de Diciembre de 2013.] http://www.um.es/atica/documentos/PREsubversion.pdf..

SkinStudio. 2010. An Introduction to SkinStudio. *Stardock.* [En línea] 2010. [Citado el: 17 de Febrero de 2014.] http://www.stardock.com/products/skinstudio/help/intro.html.

Sommerville, Ian. 2005. *Ingeniería de Software.7ma.* 2005.

Sperberg, Camilo. 2012. Sobre convenciones y notaciones (húngara, CamelCase, etc). *unreal4u´s Personal Network.* [En línea] 2012. [Citado el: 16 de Marzo de 2014.] http://blog.unreal4u.com/2011/03/sobre-convenciones-y-notaciones-hungara-camelcase-etc/.

Svensk, Magnus. 2012. DiVA. [En línea] 2012. [Citado el: 16 de Marzo de 2014.] http://urn.kb.se/resolve?urn=urn:nbn:se:hig:diva-12010..

Tedeschi, Nicolás. 2012. MSDN. ¿Qué es un Patrón de Diseño? [En línea] 2012. [Citado el: 27 de Febrero de 2014.] http://msdn.microsoft.com/es-es/library/bb972240.aspx..

TheApache. 2013. Apache httpd 2.0.65 Released. *The Apache HTTP server Project.* [En línea] 09 de 07 de 2013. [Citado el: 10 de Diciembre de 2013.] http://httpd.apache.org/.

UDIS. 2004. Patrones del Gang of Four. Facultad de informática- Universidad Politécnica de Madrid. *Sitio web de la UPM.* [En línea] 2004. [Citado el: 2 de Marzo de 2014.] http://is.ls.fi.upm.es/docencia/proyecto/docs/patrones_gof.pdf.

Valdelli, Ilario. 2006. Javascript. Aspectos y características generales. *HTMLPOINT.com.* [En línea] 2006. [Citado el: 28 de Enero de 2014.] http://www.htmlpoint.com/javascript/corso/js_02.html.

Visual-Paradigm. 2005. UML, BPMN and Database Tool for Software Development. *Visual Paradigm.* [En línea] 2005. [Citado el: 3 de Diciembre de 2013.] http://www.visual-paradigm.com.

Vromans, Johan. 2010. Vromans.ORG. GUI development with wxGlade. *Johan Vromans.* [En línea] 2010. [Citado el: 27 de Noviembre de 2013.] http://www.vromans.org/johan/articles/wxglade.pdf..

Windows, Microsoft. 2014. Temas de Windows. [En línea] 2014. [Citado el: 6 de Febrero de 2014.] http://windows.microsoft.com/es-419/windows/themes.

Zend-Technologies. 2006. Zend Framework. [En línea] 2006. [Citado el: 10 de Diciembre de 2013.] http://framework.zend.com/about/overview.

Anexos

Anexo 1 Descripción del requisito RF 3.1 Modificar tema dinámico para el entorno de escritorio Estilo Windows

RF 3.1 Modificar tema dinámico para el entorno de escritorio Estilo Windows

Precondiciones	Se ha registrado al menos un tema dinámico de interfaz gráfica de usuario en el sistema.
Flujo de eventos	
Flujo básico Configurar temas de Interfaz Gráfica de Usuario para el entorno de escritorio estilo Windows.	
1	Se selecciona el tema de Interfaz Gráfica de Usuario a modificar.
2	El sistema muestra y permite editar los atributos del tema.(ver especificación de requisito Crear vista previa)
3	Se seleccionan los atributos del tema que se desean modificar: Tipo degradado de la barra de tareas. Color superior de la barra de tareas. Color del botón de la barra de tareas. Color del botón de inicio. Color del evento hover del botón de inicio Color de texto del botón de inicio Tipo de fuente del botón de inicio Estilo de fuente del botón de inicio. Tamaño de fuente del botón de inicio Tipo de degradado del menú de inicio. Color del menú de inicio. Cargar ícono del botón de inicio (ver especificación de requisito Cargar ícono del botón de inicio) Cargar logo de la entidad (ver especificación de requisito Cargar logo de la entidad) Cargar imagen de fondo (ver especificación de requisito Cargar imagen de fondo)
4	Se presiona el botón Aceptar.
5	El sistema modifica los atributos del tema.
6	El sistema confirma la modificación de los atributos.
7	Concluye el requisito
Pos-condiciones	
1	Se establece una nueva configuración del tema de interfaz gráfica de usuario.
Flujo alternativo	
1	N/A
Pos-condiciones	
1	N/A
Validaciones	
1	N/A

Conceptos	**Tema**	Visibles en la interfaz: Entorno de escritorio comercial Entorno de escritorio estilo Windows Recursos visuales Internos: Nombre Identificador
	Entorno de escritorio	Visibles en la interfaz: Internos: Nombre Identificador
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo de interfaz RF3 Configurar temas de interfaz de usuario para el entorno de escritorio estilo Windows

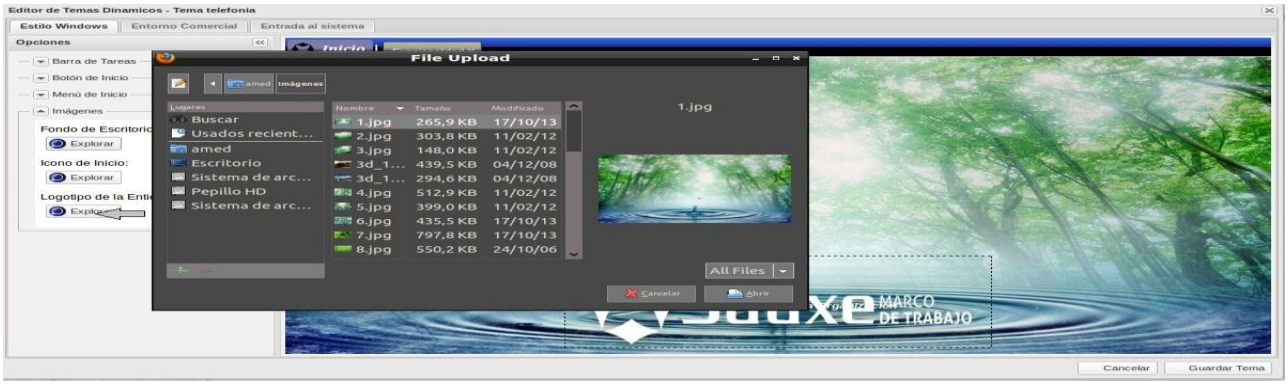
Anexo 2. Descripción del requisito RF 3.2 Cargar logo de la entidad para el entorno de escritorio Estilo Windows

- **RF 3.1 Cargar logo de la entidad para el entorno de escritorio Estilo Windows**

Precondiciones	
Flujo de eventos	
Flujo básico Cargar logo de la entidad.	
1	Se oprime el botón “Explorar”.

2	El sistema muestra una ventana para la búsqueda de la ubicación física del archivo en el disco duro.
3	Se selecciona la imagen que se desea cargar.
4	Volver al paso 4 de la especificación del requisito Configurar tema de interfaz de usuario para el entorno de escritorio estilo Windows
Pos-condiciones	
1	Se carga el nuevo logo de la entidad para el entorno de escritorio estilo Windows
Flujos alternativos	
Flujo alternativo	
1	N/A
Pos-condiciones	
1	N/A
Validaciones	
1	N/A
Conceptos	
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo de interfaz del requisito RF 3.2 Cargar logo de la entidad para el entorno de escritorio Estilo Windows

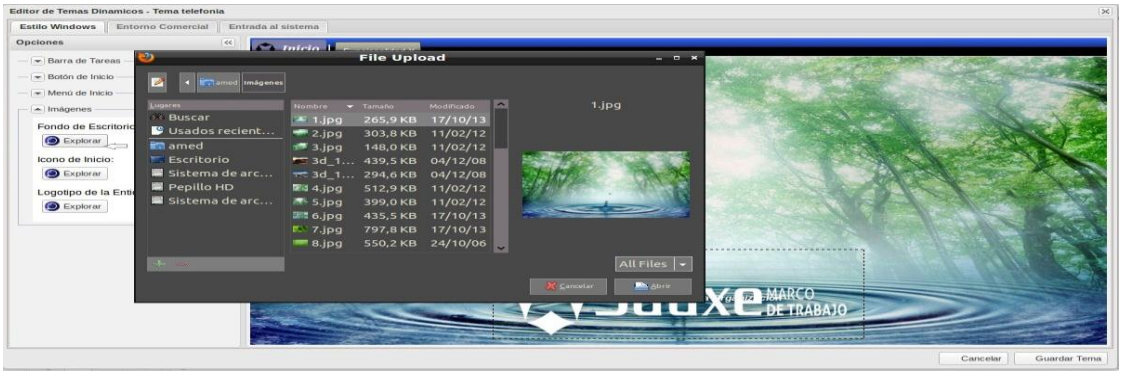


Anexo 3. Descripción del requisito RF 3.3 Cargar imagen de fondo para el entorno de escritorio Estilo Windows

- **RF 3.3 Cargar imagen de fondo para el entorno estilo Windows**

Precondiciones	
Flujo de eventos	
Flujo básico Cargar logo de la entidad.	
1	Se oprime el botón “Explorar”.
2	El sistema muestra una ventana para la búsqueda de la ubicación física del archivo en el disco duro.
3	Se selecciona el logo que se desea cargar.
4	Volver al paso 4 de la especificación del requisito Configurar tema de interfaz de usuario para el entorno de escritorio estilo Windows
Pos-condiciones	
1	Se carga el nuevo logo de la entidad para el entorno de escritorio estilo Windows
Flujos alternativos	
Flujo alternativo	
2	N/A
Pos-condiciones	
2	N/A
Validaciones	
2	N/A
Conceptos	
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo de interfaz del requisito RF 3.2 Cargar imagen de fondo

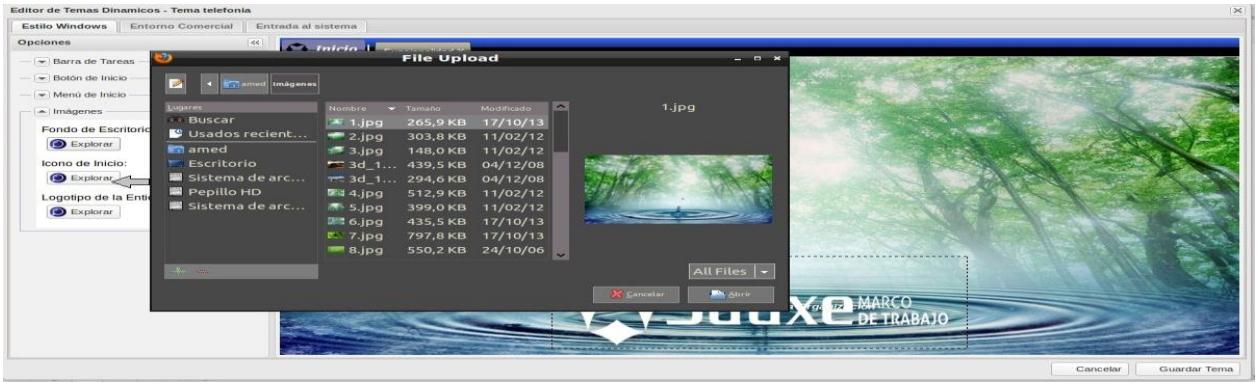


Anexo 4. Descripción del requisito RF 3.4 Cargar ícono del menú de inicio para el entorno de escritorio Estilo Windows

- **RF 3.4 Cargar ícono del menú de inicio para el entorno de escritorio Estilo Windows**

Precondiciones	
Flujo de eventos	
Flujo básico Cargar ícono del botón de inicio.	
1	Se oprime el botón “Explorar”.
2	El sistema muestra una ventana para la búsqueda de la ubicación física del archivo en el disco duro.
3	Se selecciona el ícono que se desea cargar.
4	Volver al paso 4 de la especificación del requisito Configurar tema de interfaz de usuario para el entorno de escritorio estilo Windows
Pos-condiciones	
1	Se carga el nuevo ícono del botón de inicio para el entorno de escritorio estilo Windows
Flujos alternativos	
Flujo alternativo	
1	N/A
Pos-condiciones	
1	N/A
Validaciones	
1	N/A
Conceptos	
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo de interfaz RF3.4 Cargar ícono del menú inicio para el entorno Estilo Windows



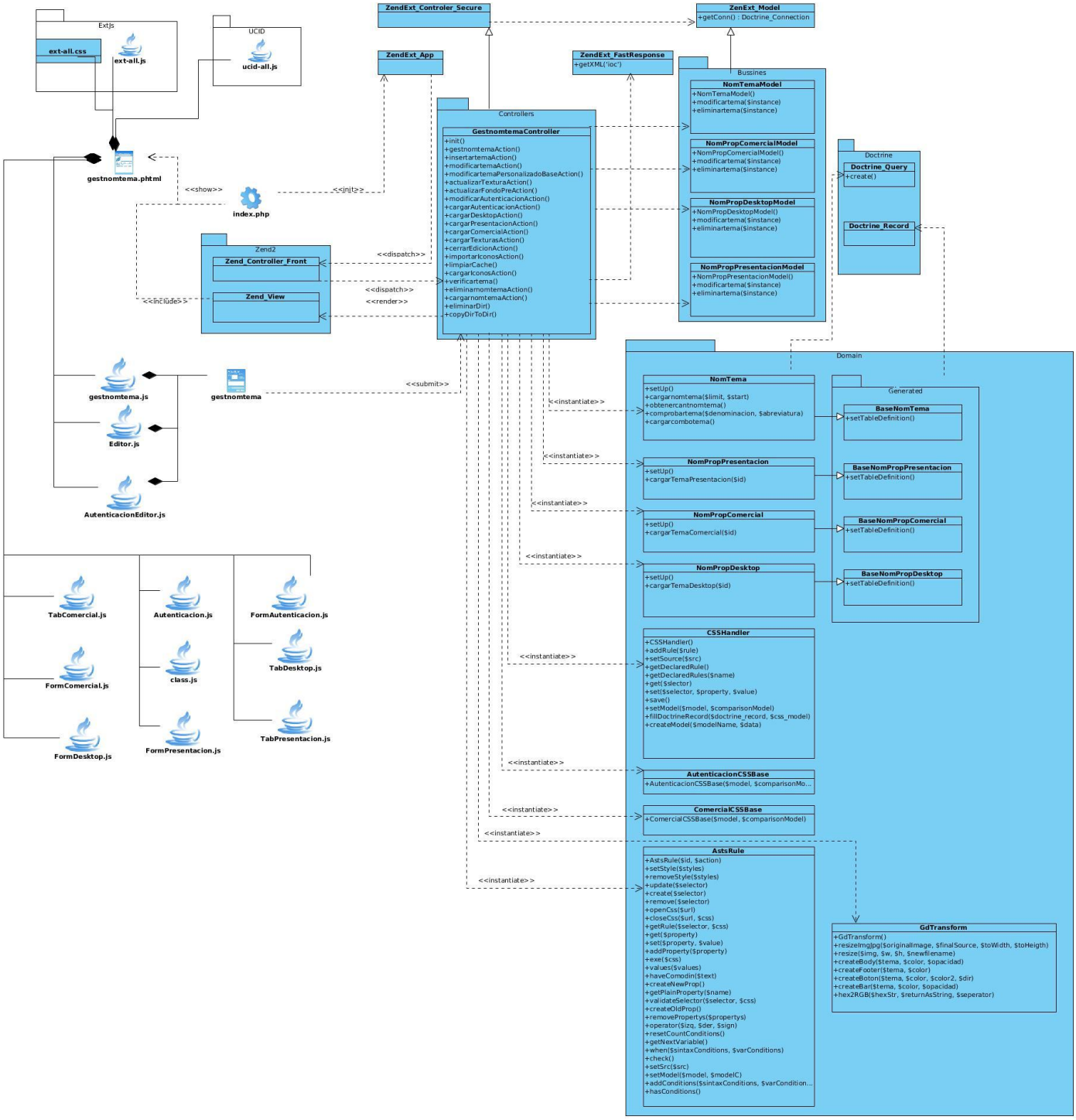
Anexo 5. Descripción del requisito RF 3.5 Crear vista previa para el entorno de escritorio Estilo Windows

- **RF 3.5 Crear vista previa para el entorno estilo Windows**

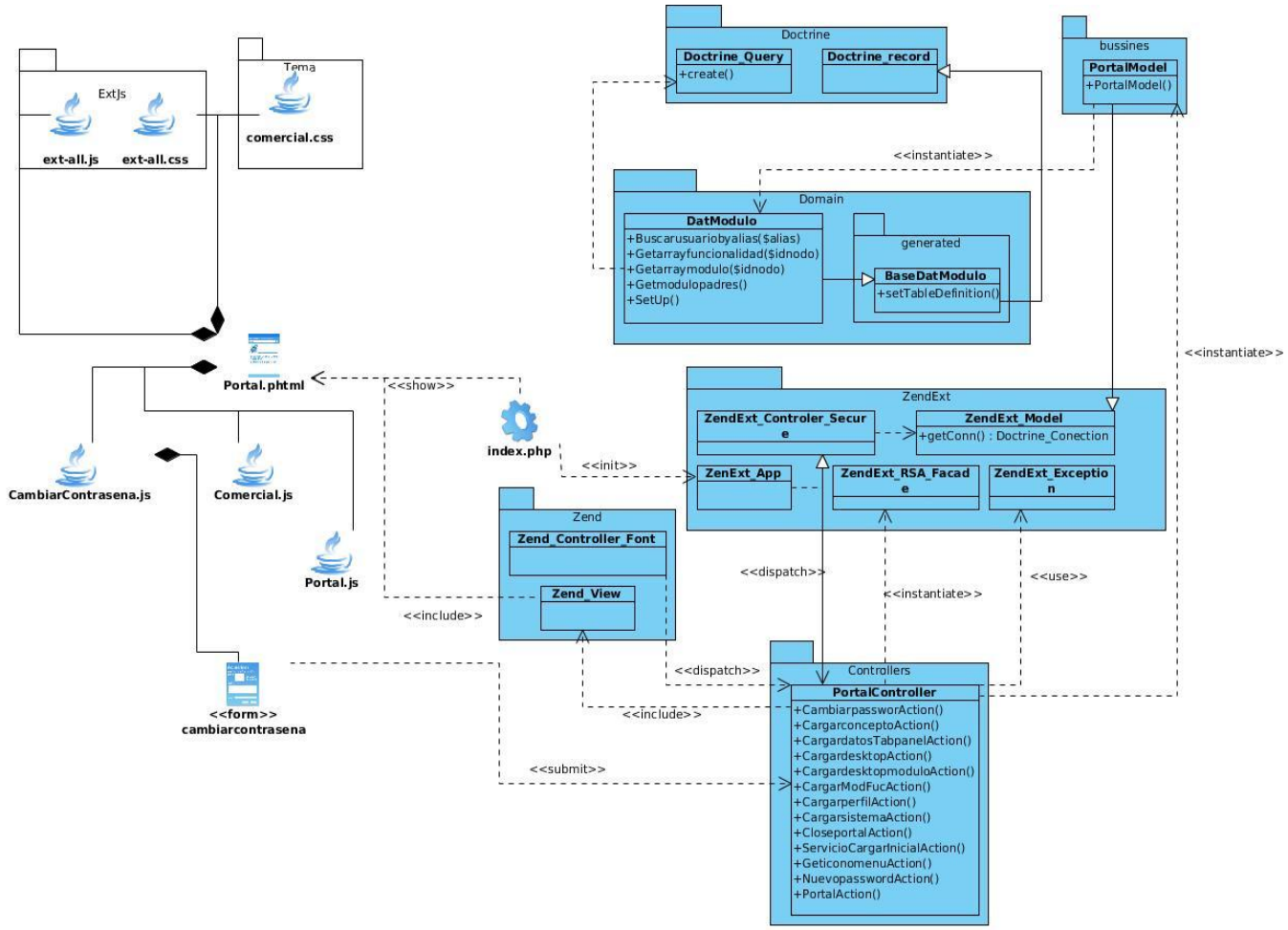
Precondiciones	Se ha seleccionado la opción “Modificar” y se posiciona en la pestaña del editor llamada “Estilo Windows”		
Flujo de eventos			
Flujo básico Crear Vista previa.			
1	El sistema muestra la paleta de componentes a modificar y una pantalla en la que se pueden observar los cambios que se estén realizando		
2	Concluye el requisito		
Pos-condiciones			
1	N/A		
Flujos alternativos			
Flujo alternativo			
1	N/A		
Pos-condiciones			
1	N/A		
Validaciones			
1	N/A		
Conceptos		**Tema**	Visibles en la interfaz: Entorno de escritorio estilo Windows Recursos visuales Internos: Nombre Identificador
		Entorno de escritorio	Visibles en la interfaz: Internos: Nombre Identificador
Requisitos especiales	N/A		
Asuntos pendientes	N/A		

Prototipo de interfaz RF3.5 Crear vista previa.

Anexo 6 Diagrama de clases Herramienta para la personalización de temas



Anexo 7 Diagrama de clases entorno de escritorio comercial



Anexo 8. Instrumento de evaluación de la métrica TOC

	Subsistema	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	Herramienta para la personalización de temas	GestnomtemaController	22	Alta	Alta	Baja
2	Herramienta para la personalización de temas	NomTemaModel	2	Media	Media	Media
3	Herramienta para la personalización de temas	NomPropComercialModel	1	Baja	Baja	Alta
4	Herramienta para	NomPropDesktopMo	1	Baja	Baja	Alta

	la personalización de temas	del				
5	Herramienta para la personalización de temas	NomPropPresentacionModel	1	Baja	Baja	Alta
6	Herramienta para la personalización de temas	NomTema	5	Alta	Alta	Baja
7	Herramienta para la personalización de temas	NomPropPresentacion	1	Baja	Baja	Alta
8	Herramienta para la personalización de temas	NomPropComercial	1	Baja	Baja	Alta
9	Herramienta para la personalización de temas	NomPropDesktop	1	Baja	Baja	Alta
10	Herramienta para la personalización de temas	BaseNomTema	1	Baja	Baja	Alta
11	Herramienta para la personalización de temas	BaseNomPropPresentacion	1	Baja	Baja	Alta
12	Herramienta para la personalización de temas	BaseNomPropComercial	1	Baja	Baja	Alta
13	Herramienta para la personalización de temas	BaseNomPropDesktop	1	Baja	Baja	Alta
14	Herramienta para la personalización de temas	CSSHandler	10	Alta	Alta	Baja
15	Herramienta para la personalización de temas	AutenticacionCSSBase	1	Baja	Baja	Alta
16	Herramienta para la personalización de temas	ComercialCSSBase	1	Baja	Baja	Alta
17	Herramienta para la personalización de temas	AstsRule	29	Alta	Alta	Baja
18	Herramienta para la personalización de temas	GDTransform	7	Alta	Alta	Baja
19	Herramienta para la personalización de temas	DesktopCSSBase	1	Baja	Baja	Alta

Anexo 9. Instrumento de evaluación de la métrica RC

No	Subsistema	Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
1	Herramienta para la personalización de temas	GestnomtemaController	14	Alto	Alta	Baja	Alta
2	Herramienta para la personalización de temas	NomTemaModel	1	Bajo	Baja	Alta	Baja
3	Herramienta para la personalización de temas	NomPropComercialModel	1	Bajo	Alta	Alta	Baja
4	Herramienta para la personalización de temas	NomPropDesktopModel	1	Bajo	Baja	Alta	Baja
5	Herramienta para la personalización de temas	NomPropPresentacionModel	1	Bajo	Baja	Alta	Baja
6	Herramienta para la personalización de temas	NomTema	1	Bajo	Baja	Alta	Baja
7	Herramienta para la personalización de temas	NomPropPresentacion	1	Bajo	Baja	Alta	Baja
8	Herramienta para la personalización de temas	NomPropComercial	1	Bajo	Baja	Alta	Baja
9	Herramienta para la personalización de temas	NomPropDesktop	1	Bajo	Baja	Alta	Baja
10	Herramienta para la personalización de temas	BaseNomTema	0	Ninguno	Baja	Alta	Baja
11	Herramienta para la personalización de temas	BaseNomPropPresentacion	0	Ninguno	Baja	Alta	Baja
12	Herramienta para la personalización de temas	BaseNomPropComercial	0	Ninguno	Baja	Alta	Baja
13	Herramienta para la personalización de temas	BaseNomPropDesktop	0	Ninguno	Baja	Alta	Baja
14	Herramienta para la personalización de temas	CSSHandler	1	Bajo	Baja	Alta	Baja

15	Herramienta para la personalización de temas	AutenticacionCSSBase	1	Bajo	Baja	Alta	Baja
16	Herramienta para la personalización de temas	ComercialCSSBase	1	Bajo	Baja	Alta	Baja
17	Herramienta para la personalización de temas	AstsRule	0	Ninguno	Baja	Alta	Baja
18	Herramienta para la personalización de temas	GDTransform	0	Ninguno	Baja	Alta	Baja
19	Herramienta para la personalización de temas	DesktopCSSBase	1	Bajo	Baja	Alta	Baja

Anexo 10. Cuestionario aplicado para realizar el pre-experimento

Pregunta 1

¿Qué tiempo demoró en cumplimentar la tarea de forma manual? Indique la cantidad de horas dedicadas solo en la tarea que realizó.

Ej: 0,5 para 50 minutos, 5 para 5 horas.

_____ Edición de la ventana de autenticación.

_____ Edición del entorno de escritorio comercial.

_____ Edición del entorno de escritorio estilo comercial.

_____ Edición de la ventana de presentación.

Pregunta 2

¿Qué tiempo demoró en cumplimentar la tarea con la herramienta? Indique la cantidad de horas dedicadas solo en la tarea que realizó.

Ej: 0,5 para 50 minutos, 5 para 5 horas.

_____ Edición de la ventana de autenticación.

_____ Edición del entorno de escritorio comercial.

_____ Edición del entorno de escritorio estilo comercial.

_____ Edición de la ventana de presentación.

Anexo 11. Código fuente utilizado para realizar las pruebas de caja blanca

```
public function exe($css){1
    if($this->check()){2
     switch($this->action) {3
        case AstsRule::REMOVE :{4
            $this->css=$css;5
            $one=split($this->selector,$css);5
            $two=split('}',$one[0]);5
            $r=$two[count($two)-1].$this->selector;5
            $this->selector=$r;5
            $this->createOldProp();5
            $this->css=str_replace($this->plainText,'',$this->css);5
            return $this->css;5
        }6
        case AstsRule::CREATE :{7
            $this->css=$css; 8
            $this->plainText=$this->selector.' {'; 8
            $this->createNewProp(); 8
            foreach($this->newProp as $key=>$value) { 9
                    $pro=$key.':'.$value.';'; 10
                    $this->plainText.=$pro; 10
                } 11
                $this->plainText.='}'; 12
                $this->css.=$this->plainText; 12
                return $this->css; 12
        } 13
        case AstsRule::UPDATE :{ 14
            $this->css=$css; 15
            $this->createNewProp(); 15
            $this->createOldProp(); 15
            $oldPlain=$this->plainText; 15
            if($this->remove==false){ 16
```

```
            foreach($this->newProp as $key=>$value) { 17
                $pro=$key.':'.$value; 18
                $plainProp=$this->getPlainProperty($key); 18
                if($plainProp=="ASTS-NO-EXIST-PROPERTY")  19
                    $this->plainText=$this->addProperty($pro); 20
                else
                    $this->plainText=str_replace($plainProp,$pro,$this->plainText);21
            } 22
            $this->css=str_replace($oldPlain,$this->plainText,$this->css);23
            return $this->css; 23
        }24
        else{ 25
            if(count($this->removeProp)==0){ 26
                $this->plainText=$this->selector.'{ }'; 27
                $this->css=str_replace($oldPlain,$this->plainText,$this->css); 27
                return $this->css; 27
            }28
            else{ 29
                foreach($this->removeProp as $key=>$value) { 30
                $tok=$this->getPlainProperty($value); 31
                $aux=split($tok.';',$this->plainText); 31
                    if(count($aux)>1) 32
                        $this->plainText=str_replace($tok.';','',$this->plainText); 33
                    else
                        $this->plainText=str_replace($tok,'',$this->plainText); 34
                } 35
                $this->css=str_replace($oldPlain,$this->plainText,$this->css); 36
                return $this->css;          36
            }37
        } 38
    }39
}40
else
    return false;  41
} 42
```

Anexo 12. Acta de liberación del proyecto Marco de trabajo para el desarrollo de aplicaciones web de gestión.

UCI Universidad de las Ciencias Informáticas

CEIGE CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES

DEPARTAMENTO DE TECNOLOGÍA.
22/05/2014

A quien pueda interesar:

Por este medio se hace constar que la solución Herramienta para la personalización de los temas de la capa de presentación en el marco de trabajo Sauxe de los autores William A. Tamayo Guevara y Ricardo E. Suárez Riquenes fue sometida a una revisión técnica en la cual se detectaron 8 no conformidades que fueron resueltas quedando esta solución estable y lista para su posterior uso.

Como parte del desarrollo de la solución se elaboraron y entregaron los siguientes artefactos:

1. Modelo conceptual.
2. Descripción de requisitos (6).
3. Modelo de diseño (6).
4. Diseños de casos de prueba (31).
5. Código fuente (ubicado en el repositorio de desarrollo del departamento.)

Para que así conste firman a continuación los miembros del equipo que realizaron la revisión, el autor y los tutores del trabajo.

Dado a los 3 días del mes de junio de 2014.

Nombre y apellidos	Firma
Revisores: René R. Bauta Camejo Claudia Bravo Batista Mileidy M. Sarduy Pérez Inoelkiz Velázquez Osorio	
Autor (es): William A. Tamayo Guevara Ricardo E. Suárez Riquenes	
Tutores: René R. Bauta Camejo Claudia Bravo Batista Mileidy M. Sarduy Pérez	

Jefe de Departamento de Tecnología
René R. Bauta Camejo