Universidad de las Ciencias Informáticas

Facultad 3

“Componente para la configuración y visualización de documentos primarios en el marco de trabajo Sauxe”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Roberto Ramírez Ferro

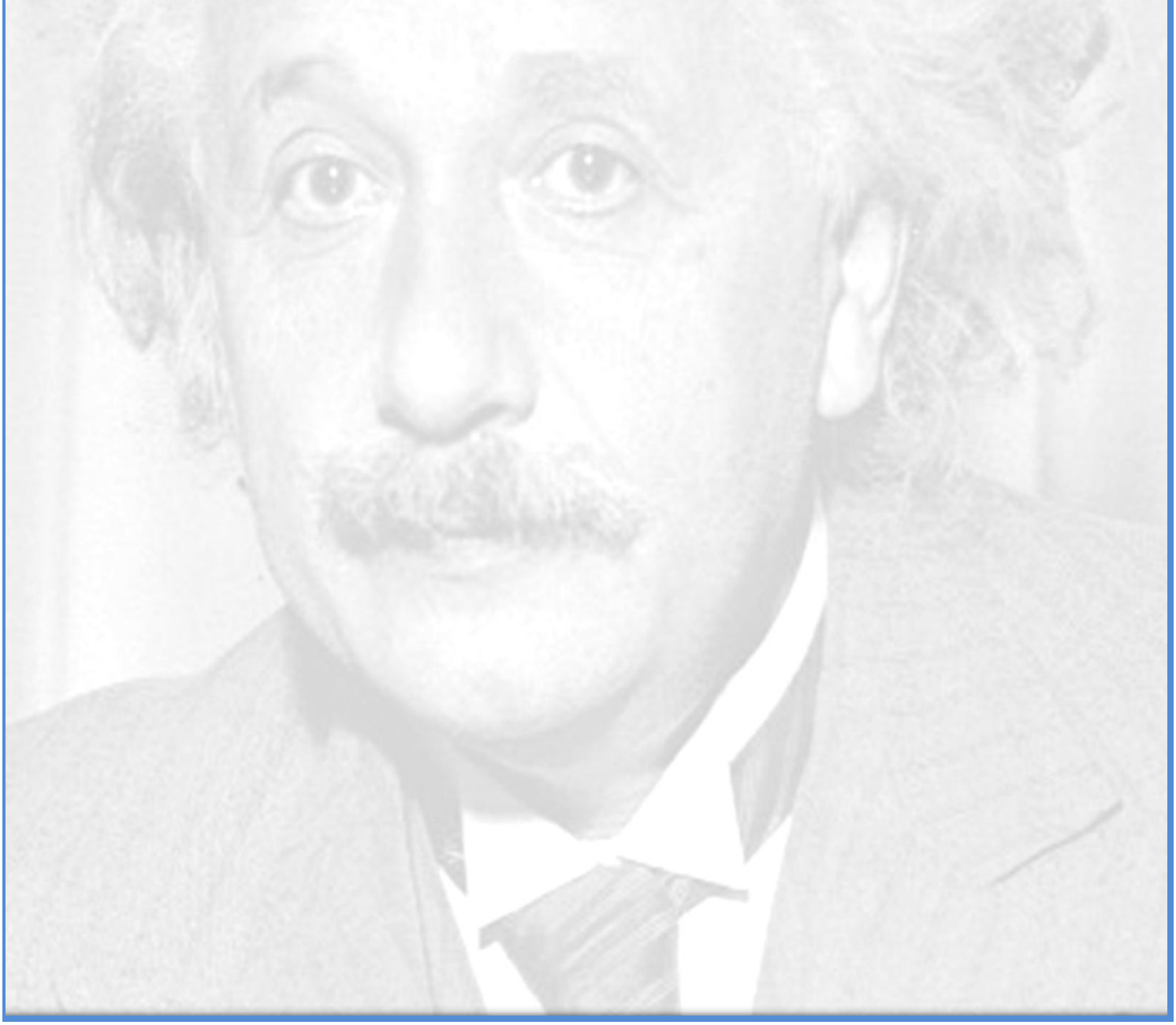
Tutores: Ing. Pedro Manuel Nogales Cobas

Ing. Javier Ruiz Duran

La Habana, Junio de 2011

“Los conceptos y principios fundamentales de la ciencia son invenciones libres del espíritu humano.”

Albert Einstein



DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro para la Informatización de Gestión de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2011.

Autor:

_____

Roberto Ramírez Ferro

Tutores:

_____

Ing. Javier Ruiz Durán.

_____

Ing. Pedro Manuel Nogales Cobas.

DATOS DE CONTACTO

Tutor: Ing. Pedro Manuel Nogales Cobas

CEIGE, Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: pmnogales@uci.cu

Tutor: Ing. Javier Ruiz Durán

CEIGE, Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jduran@uci.cu

AGRADECIMIENTOS

Quisiera agradecer a todas las personas que de una forma u otra han contribuido al desarrollo de este trabajo.

A mi familia, en especial a mi mamá y a mis abuelas por brindarme su apoyo en cualquier circunstancia.

A mi padrastro Rafael y mi tío Ernesto por estar ahí cuando los necesito.

A los mejores compañeros y amigos que he tenido durante los cinco años de la carrera, mi segunda familia: Tamara, Rogsany, Alejandro y Humberto, quienes han estado a mi lado incondicionalmente en este largo recorrido.

A mis compañeros del proyecto: Alexander, René, Inna, Oscar, Mayté, Gabriel, Yossel y Yaniris.

A mis amigos Roberto y Fernando por su paciencia y ayuda incondicional.

Al grupo al que pertenecí desde primer año y hasta tercero, sin dudas fue el mejor grupo del que he formado parte.

A mis tutores Javier y Pedro por todo lo que me enseñaron en este tiempo.

A mi oponente Carlos por su colaboración.

A todas las personas que conocí y que de alguna manera formaron parte de mi vida en esta escuela, gracias por ser parte de ella.

DEDICATORIA

A mi mamá y mis abuelas que son mi razón de ser, que han sabido soportarme durante 24 años.

Y a todos los que de una forma u otra forman o han formado parte de mi vida y de mi formación.

Para todos aquellos que no me han abandonado y han creído en mí.

RESUMEN

El desarrollo de la industria del software y la utilización de las aplicaciones en el sector empresarial garantizan que las labores diarias de los trabajadores disminuyan en complejidad y tiempo que se les dedica. Debido a que muchas empresas o negocios están en constante crecimiento, los productos de software también deben hacerlo, con el objetivo de brindar a los usuarios una aplicación que se ajuste a las necesidades reales, que facilite y mejore el trabajo diario.

Los documentos primarios son datos o registros de las operaciones de una empresa y su consulta puede ser de gran importancia para la toma de decisiones. Es fundamental que estos documentos brinden a los usuarios la información de una forma sencilla y estandarizada para que puedan ser de utilidad en la toma de decisiones.

En el Centro para la Informatización de la Gestión de Entidades (CEIGE) de la Universidad de la Ciencias Informáticas (UCI), se está desarrollando un marco de trabajo para desarrollar aplicaciones web de gestión llamado Sauxe, en el cual no se gestionan los documentos primarios, motivo por el cual el presente trabajo constituye un punto de partida para lograr el desarrollo de un componente dentro de Sauxe, que permita la configuración y visualización de los documentos primarios.

La presente investigación pretende brindar a Sauxe una herramienta eficiente para el trabajo con los documentos primarios. Esto permitirá su configuración y visualización, e indirectamente una reducción del tiempo empleado en las labores de consulta y toma de decisiones.

Palabras claves: documentos primarios, estandarización, configuración, visualización, marco de trabajo.

Índice de contenido

Índice de figuras

Índice de tablas

INTRODUCCIÓN

Hoy en día, con el desarrollo constante de las Tecnologías de la Información y las Comunicaciones (TICs), las empresas se encuentran dependientes y expectantes de mejores productos de software, que posibiliten y aumenten el aprovechamiento del tiempo, así como la optimización de los recursos y automatización de las tareas a realizar. Además esperan que estas aplicaciones sean eficientes y permitan resultados más específicos y de mayor utilidad en el momento y forma en que son solicitados y mostrados para su consulta.

Sin lugar a dudas, en todo sistema o aplicación que maneje cualquier tipo de información es muy importante la forma en que se muestran los datos que le son solicitados, así como el formato, la exactitud y coherencia en relación a lo que se brinda y lo que se necesita, para que solo sean devueltos los datos suficientes y necesarios para llevar a cabo determinada acción.

Las aplicaciones web están entre los productos de software más usados en la actualidad, debido a las ventajas y facilidades que brindan. Específicamente, las aplicaciones web de gestión están en una situación similar, pero con más enfoque a los negocios y empresas que pretenden brindar un servicio a sus clientes con mayor calidad, seguridad y que esté disponible todo el tiempo para su utilización, sin que deba el cliente tener nada más instalado que un navegador en su ordenador. Muchas de las aplicaciones web son implementadas sobre marcos de trabajo para optimizar el tiempo y reutilizar soluciones.

Los Marcos de Trabajo o Frameworks (en Inglés) son estructuras tecnológicas que dan soporte para que otras aplicaciones puedan ser desarrolladas y específicamente los que son utilizados para desarrollar aplicaciones web, poseen gran demanda en la industria del software.

En Cuba, con el desarrollo de la informática y la puesta de este sector en función del desarrollo de la sociedad, la producción y optimización de los procesos en las empresas, se ha hecho necesario que el país realice sus propias aplicaciones informáticas, para de esta forma, contribuir con la economía y evitar gastos innecesarios de divisas en adquisición de software y licencias.

Las empresas y organismos cubanos para lograr mejores resultados y dar su aporte en el crecimiento económico, necesitan de aplicaciones que permitan la gestión de los procesos, tareas y personal vinculado a las mismas de una forma eficiente y segura. Asimismo estos productos deben ser capaces de adaptarse a las peculiaridades económicas que existan e incluso de estar en constante actualización e intercambio con sucursales o divisiones aún cuando ellas se encuentren en lugares distantes, pues de esta forma se logra un mejor funcionamiento. Es fundamental que estos sistemas permitan mostrar la información precisa en dependencia de la solicitud realizada, lo cual generalmente es la fuente principal para tomar una decisión, facilitar el trabajo, o bien optimizar el tiempo, gracias a que los datos se encuentran de manera más compacta, específica y están accesibles en el momento y lugar indicado sin tener que localizarlos o elaborarlos de forma manual.

Actualmente, equipos de desarrollo de software, compuestos por estudiantes y profesionales del Centro de Informatización para la Gestión de Entidades (CEIGE), de la Universidad de las Ciencias Informáticas (UCI), se encuentran inmersos en el desarrollo de un Marco de Trabajo para aplicaciones web llamado Sauxe.

El marco de trabajo Sauxe permite el desarrollo de aplicaciones web de gestión, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Los sistemas implementados sobre Sauxe pueden tener la necesidad de compartir documentos primarios, los cuales son el registro inicial de cualquier operación o transacción que sea realizada y recogen los datos de las mismas. En Sauxe, actualmente no es fácil el acceso a los documentos primarios y no están disponibles a tiempo completo. Si se desea realizar consultas a datos e informaciones es necesario enviar su solicitud por correo electrónico, personalmente o mediante otro sistema, si cuenta con los permisos para ello, lo que en el mejor de los casos obliga al usuario a salir del sistema donde se encuentra. Aunque también pueden solicitarse datos a través de servicios, su visualización no es de una manera estándar, esto ocasiona molestias y dificulta el trabajo.

De lo anteriormente planteado se deriva el siguiente **problema a resolver:** ¿Cómo gestionar la información visual de documentos primarios en los sistemas que utilizan el marco de trabajo Sauxe?

Para la solución del problema planteado se define como **objeto de estudio** los marcos de trabajo para aplicaciones web de gestión.

A modo de dar resolución al problema anterior se traza como **objetivo general** desarrollar un componente que permita la visualización y configuración de documentos primarios en los sistemas del marco de trabajo Sauxe**.**

Fueron trazados para dar cumplimiento al objetivo general los siguientes **objetivos específicos:**

Construir el marco teórico de la investigación.
Realizar análisis y diseño de la solución.
Realizar implementación y prueba de la solución.

Como **campo de acción** se definió el marco de trabajo Sauxe y como **idea a defender:** si se desarrolla un componente que permita la configuración y visualización de documentos primarios en el marco de trabajo Sauxe, se podrá gestionar la información visual de los mismos en los sistemas que utilicen dicho marco de trabajo, lo cual permitirá que las labores sean menos engorrosas, sea menor el tiempo empleado en ellas y la toma de decisiones esté basada en una fuente de datos con mayor calidad.

Para dar cumplimiento a los objetivos expuestos se definieron las siguientes **Tareas de la investigación.**

1. Construir el marco teórico de la investigación.
2. Describir los procesos de negocio a automatizar.
3. Hacer un estudio de las herramientas similares existentes.
4. Capturar los requisitos del componente.
5. Definir los escenarios arquitectónicos del componente.
6. Realizar el diseño de la solución.
7. Implementar la solución.
8. Realizar las pruebas unitarias a la solución.
9. Documentar la solución obtenida.

El presente trabajo de diploma quedará estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica.

Serán descritos los principales conceptos relacionados con el tema del trabajo. Se realizará un estudio del estado del arte que incluye la disertación de las aplicaciones que serán utilizadas, las tecnologías y metodologías, haciendo un análisis crítico y valorativo de sus características, ventajas y desventajas, con el propósito de obtener base y conocimientos para proponer la mejor solución al problema que se plantea.

Capítulo 2: Propuesta de solución.

El capítulo se centra en definir los temas de análisis y diseño de la solución, donde se plantean las características del sistema para su posterior implementación. Se examinarán cada uno de los procesos de negocio, requisitos funcionales y no funcionales, prototipos de interfaces, modelos, y diagramas que serán la guía para la siguiente etapa de desarrollo.

Capítulo 3: Implementación y prueba.

En este capítulo se realizará una valoración de la solución. Se expondrá una explicación de cómo se lleva a cabo el proceso para la implementación de la solución y se validará el sistema desarrollado mediante la realización de métricas para el diseño y pruebas de caja blanca y negra para la implementación, las cuales garantizarán la calidad de la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo constituye el marco teórico de la investigación a realizar. Define los principales conceptos para facilitar un mejor entendimiento del objetivo fundamental del trabajo, el cual profundiza en temas como las aplicaciones web de gestión y los marcos de trabajo, enfatizando en los más usados a nivel global. Además se describen las tecnologías y herramientas que serán utilizadas durante la implementación de la solución.

1.1 Aplicaciones web

Son aquellos sistemas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Las interfaces web tienen ciertas limitaciones en las funcionalidades que se ofrecen al usuario. Hay funcionalidades comunes en las aplicaciones de escritorio, como dibujar en la pantalla o arrastrar-y-soltar que no están soportadas por las tecnologías web estándar. Los desarrolladores web generalmente utilizan lenguajes interpretados o script en el lado del cliente para añadir más funcionalidades, especialmente para ofrecer una experiencia interactiva que no requiera recargar la página cada vez (lo que suele resultar molesto a los usuarios). El desarrollo de aplicaciones web incorpora métodos de proceso especializados, métodos de ingeniería de software adaptados a características de desarrollo de las aplicaciones web y un conjunto de importantes tecnologías que permitan un correcto desarrollo de las mismas. Su evolución ha sido tanta que desde sus inicios a la actualidad realmente quedan pocas diferencias con las aplicaciones de escritorio, aunque la estandarización es uno de los problemas que presentan, pues la variedad de navegadores debe tenerse en cuenta para garantizar que todos los usuarios puedan obtener los beneficios completos de la aplicación. (1)

1.1.1 Ventajas

Aunque existen muchos problemas al tratar de ofrecer funcionalidades a través de la Web y con la facilidad de uso que estas ofrecen, las aplicaciones web han adquirido una extraordinaria popularidad, ya que ofrecen una serie de importantes ventajas tecnológicas sobre las aplicaciones de escritorio. (1)
Ventajas:

- ✓ Son fáciles de hacer y de bajo costo. Los usuarios solo necesitan una computadora, un navegador y una conexión a internet o intranet.
- ✓ Son fáciles y baratas a la hora de mejorarlas. Solamente tiene que ser mejorada la aplicación en el servidor y todos los clientes la tienen actualizada.
- ✓ Sus requisitos son flexibles para los usuarios finales. Son independientes de la plataforma y si está bien implementada la aplicación debe verse correctamente desde cualquier navegador.
- ✓ Hacen más fácil el uso de un almacén de datos central. Diferentes lugares tratando de acceder a una misma base de datos pueden causar problemas de seguridad y sincronización.

1.1.2 Aplicaciones web de gestión

Las aplicaciones web de gestión no son más que un tipo de aplicaciones web que permiten trabajar en un entorno unificado y centralizado, con acceso seguro e inmediato a los datos y las tareas de interés para cada organización. Y además, al tener la información en un servidor web, se puede acceder a ella desde cualquier dispositivo con acceso a Internet o Intranet según las peculiaridades del lugar donde este implantada.

Con todo ello, se gana en eficacia tanto a nivel interno como externo, ya que las ventajas pueden trasladarse a los clientes en forma de **servicios online** que sustituyan progresivamente a las tareas administrativas tradicionales.

Con este tipo de aplicaciones se pueden controlar procesos tan importantes como:

- ✓ La facturación.
- ✓ Los presupuestos.
- ✓ La actividad comercial.
- ✓ El personal.
- ✓ La contabilidad.
- ✓ Los productos en existencia e inventario.

Una aplicación que controle tantos procesos sin dudas debe permitir visualizar y manejar cierto tipo de documentación que es generada por las empresas al realizar determinados procesos, transacciones u operaciones; estos documentos son llamados documentos primarios.

1.2 Documentos primarios

Los documentos primarios son aquellos que contienen información nueva y original que no ha sido sometida a ningún tipo de tratamiento documental posterior (resumen, selección o interpretación). Son documentos primarios los libros, revistas, periódicos, programas de radio o televisión, películas, páginas web, archivos gráficos o textuales de ordenador. Son parte también de los denominados documentos primarios la literatura gris, documentos que no siguen los canales habituales de publicación y difusión, actas de congresos científicos, cuadernos de laboratorio, informes científicos, informes internos, ediciones técnicas como normas, patente o marcas, tesis doctorales y otros trabajos de investigación universitaria y, en cierto modo, las publicaciones oficiales. (2)

Se puede encontrar en la bibliografía que los documentos primarios: son aquellos donde se registran los hechos y fenómenos económicos ocurridos con sus características cualitativas y cuantitativas en el lugar y fecha en que ocurren y se originan.

Los documentos primarios son muy importantes en la contabilidad. Del cuidado, veracidad, exactitud, oportunidad y técnica con que se elaboren; depende que todo el proceso y registro de los hechos económicos ofrezca una información que suministren los elementos de juicio para la aceptada planificación y dirección de las tareas económicas. (3)

Entre los documentos primarios de uso más frecuente podemos encontrar:

- ✓ Vale para pagos menores.
- ✓ Informe de Recepción.
- ✓ Tarjeta de Estiba.
- ✓ Factura.
- ✓ Conduce.
- ✓ Anticipos y Liquidación de Gastos de Viaje.

- ✓ Solicitud de Materiales.
- ✓ Registro de Asistencia.
- ✓ Orden de Compra.
- ✓ Orden de Servicio.
- ✓ Reembolso para pagos menores.
- ✓ Ajuste de Inventario.
- ✓ Transferencias entre almacenes.

En sentido general y a modo de resumen puede decirse que los documentos primarios son el registro inicial de cualquier operación o transacción que sea realizada en las empresas, donde además son recogidos y registrados todos los datos referentes a las mismas.

1.3 Marcos de Trabajo

Los desarrolladores de software para la creación de aplicaciones web utilizan marcos de trabajo que faciliten el desarrollo de este tipo de aplicaciones.

Dentro del ambiente de desarrollo de software para el término "Marcos de Trabajo" o Framework, como se conoce en inglés, podemos encontrar algunas definiciones como: "En sistemas orientados a objetos, un conjunto de clases que incluye el diseño de soluciones abstractas a un determinado número de problemas similares", (4) y "Una aplicación reusable, semi o completamente que puede especializarse para producir aplicaciones personalizadas". (5)

También según el sitio The SOA Agend, Marco de trabajo es un concepto sumamente genérico, se refiere a "ambiente de trabajo, y ejecución". En general los marcos de trabajo son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). (6)

Recientemente, el interés en reutilizar software ha sido cambiado de la reutilización de componentes simples a diseño de sistemas enteros o estructuras de aplicaciones. Un software que pudiera ser reutilizado en este nivel para la creación de aplicaciones completas es llamado *marco de trabajo*. Los

marcos de trabajo son basados en la idea que deberían permitir la producción fácil de un conjunto de sistemas específicos pero similares, dentro de un cierto dominio comenzando desde una estructura genérica. Brevemente, los marcos de trabajo son arquitecturas genéricas integradas por un extensible conjunto de componentes. Además, los marcos de trabajo pueden contener a otros que representen subconjuntos de componentes de un sistema más grande. (7)

En resumen los marcos de trabajo son estructuras tecnológicas que dan soporte para que otras aplicaciones puedan ser organizadas y desarrolladas, ofreciendo soluciones genéricas a determinadas situaciones o problemas pero que pueden ser fácilmente adaptables a casos más específicos, pueden incluir además soporte de programas y librerías. En fin están creados para facilitar el desarrollo de software.

A continuación se realiza un estudio de los marcos de trabajo existentes con similitud en el tratamiento de los documentos primarios.

1.3.1 Framework .NET

.NET Framework es la **plataforma de desarrollo** de código administrado de Microsoft. Está formado por una serie de herramientas y librerías con las que se pueden crear todo tipo de aplicaciones, desde las tradicionales aplicaciones de **escritorio** hasta aplicaciones para **XBOX** (XNA) pasando por **desarrollo web** (ASP.NET), desarrollo para **móviles** (compact framework), aplicaciones de **servidor** (WPF, WCF), etcétera.

El marco de trabajo .NET de Microsoft proporciona un entorno eficaz para crear y mostrar documentos de gran calidad. Las características mejoradas que admiten tanto documentos fijos como dinámicos y controles de vista avanzados, aportan a las aplicaciones .NET Framework un nuevo nivel de calidad y experiencia del usuario. La posibilidad de administrar con flexibilidad una representación en memoria de un documento es una característica clave de .NET Framework. Por su parte, la capacidad de guardar y cargar con eficacia documentos de un almacén de datos es algo imprescindible en casi todas las aplicaciones. El proceso de convertir un documento de una representación en memoria interna en un almacén de datos externo se denomina serialización. El proceso inverso de leer un almacén de datos y volver a crear la instancia en memoria original se denomina deserialización.

Serialización de documentos en .NET

En una situación ideal, el proceso de serializar un documento a partir de su representación en memoria y de deserializarlo para volver a crear una instancia de él en memoria resulta transparente a la aplicación. La aplicación llama a un método serializador de "escritura" para guardar el documento, mientras que un método deserializador de "lectura" obtiene acceso al almacén de datos y vuelve a crear la instancia original en memoria.

Con frecuencia, las aplicaciones proporcionan varias opciones de serialización que permiten al usuario guardar documentos en un soporte diferente o con un formato diferente. Para la aplicación, la serialización define una interfaz que aísla los detalles del soporte de almacenamiento dentro de la implementación de cada serializador concreto. Además de las ventajas de encapsular los detalles del almacenamiento, las API.NET FrameworkSystem.Windows.Documents.Serialization proporcionan otras características importantes.

Características de los serializadores de documentos de .NET Framework 3.0

- El acceso directo a los objetos de documento de alto nivel (árbol lógico y visual) permite el almacenamiento eficaz de contenido paginado, elementos 2D y 3D, imágenes, multimedia, hipervínculos, anotaciones y otro contenido de compatibilidad.
- Funcionamiento sincrónico y asincrónico.
- Compatibilidad con los serializadores de complemento con funciones mejoradas:
    - Acceso en todo el sistema para su uso por parte de todas las aplicaciones .NET Framework.
    - Sencilla capacidad de detectar complementos de aplicaciones.
    - Implementación, instalación y actualización sencillas de complementos personalizados de otros fabricantes.
    - Compatibilidad de la interfaz de usuario con la configuración y las opciones personalizadas en tiempo de ejecución.

Reportes en .NET

Crystal Reports aunque no forma parte exactamente del Framework .NET si está incluido o vinculado a Visual Studio .NET, IDE que sin dudas complementa al marco de trabajo. Puede usar Crystal Reports si distribuye los archivos de informe con la aplicación y crea un proyecto de instalación que contenga módulos de combinación específicos para Crystal Reports. Puede elegir entre compilar los archivos de informe en la aplicación o distribuirlos por separado.

Cada proyecto de configuración contiene una aplicación y sus componentes necesarios. Los componentes de una implementación incluyen archivos de informes, archivos de redistribución de Crystal Reports y el paquete de redistribución del Framework NET.

Crystal Reports puede ser utilizado para añadir funciones de elaboración de informes a una aplicación o a un servicio Web. Puede crear un informe empezando desde cero o utilizar uno de los Asistentes de Crystal Reports para obtener ayuda en el proceso de diseño.

Los informes creados se pueden almacenar tanto en las aplicaciones Web como en las de Windows y pueden utilizar varios orígenes de datos pero debe seleccionar primero el origen de los mismos al que hará referencia el informe. También puede publicar un informe de Crystal como un servicio Web de informes en un servidor Web.

Independientemente de lo que decida hacer con el informe, puede modificarlo en el Crystal Report Designer, el cual le permite diseñar y modificar los informes del Entorno de programación integrado (IDE) de Visual Studio .NET.

Crystal Reports ofrece la posibilidad de crear informes, conectarlos a un control CrystalReportViewer e interactuar mediante programación con estos informes. Crystal permite una fácil manipulación de los datos y proporciona varias opciones para seleccionar registros, agrupar datos en un informe, seleccionar registros basados en rangos sencillos de fechas o comparaciones, crear fórmulas complejas, identificar los registros que se van a incluir, llevar a cabo cálculos, marcar los datos para llamar la atención y darles formato, lo que incluye cambios en la organización, presentación y diseño de un informe, así como

también la apariencia del texto, objetos o secciones completas de informes que permitan darle a los mismos una apariencia profesional.

Stimulsoft Reports.Ultimate

Una herramienta con la que cuenta el Framework .NET para generar informes es Stimulsoft Reports.Ultimate, la cual es una solución completa para hacer los informes para la plataforma. NET Framework. El producto incluye un conjunto completo de herramientas para crear informes en WinForms, ASP.NET, Silverlight y entornos WPF. Report Designers, se puede ejecutar en tiempo de diseño y de ejecución, incluyendo un diseñador de informes únicos para la Web. Viewers para la visualización de informes. Potente sistema de informes de exportación, que es compatible con muchos tipos diferentes de formatos. Simple pero muy potente motor de informes. Uno de los principios básicos del uso de Stimulsoft Reports.Ultimate es su diferente tecnología, pero con enfoques comunes en la creación de informes. Cuando la migración de aplicaciones a la nueva tecnología, los principios de trabajo con informes permanecen sin cambios.

Los reportes.

Un complejo, pero simple generador de informes. ¿Es posible combinar dos características opuestas en uno? Para hacer todos los informes, Stimulsoft Reports.Ultimate utiliza un motor de software único. Se pueden crear informes simples, complejos y otros muy complicados. Cualquier informe se construirá de la manera más precisa y rápida posible. El motor de esta herramienta de informes no es sólo potente, sino también muy funcional. Gran variedad de componentes: los gráficos, cubos OLAP, cuadros. Cada componente se enriquece con múltiples propiedades. Esta herramienta pueden satisfacer los requisitos más severos.

Visor de reportes.

Como parte de Stimulsoft Reports.Ultimate están disponibles ocho visualizadores de informes debido a que no se quiere limitar a los usuarios en el momento de mostrar los informes en sus aplicaciones. Los informes pueden tener diferentes interfaces, incluso pueden ser mostrados como para imprimirlos con una impresora sencilla y de matriz de puntos. Todos los visores de informes disponibles utilizan el mismo

principio de la estructura de la interfaz y el mismo formato de archivo. No es necesario convertir los informes o copiarlos.

Diseñador de reportes.
Existen tres diseñadores de informes para la plataforma. NET Framework con algunas diferencias entre ellos. La diferencia radica fundamentalmente en las tecnologías que utilizan para crear los informes. Uno de los diseñadores de informes es un componente Web único, de gran utilidad para trabajar en el entorno Web y el cliente necesita tener solamente el Adobe Flash Player. Todos los diseñadores tienen una estructura similar de interfaz de informes, que, a su vez, es estándar y tiene gran similitud con el Office de Microsoft.
Resultado.
Se puede cambiar el contenido de los componentes directamente en el visor. Puede ejecutar el diseñador de informes y editar una página del informe. Puede añadir o eliminar páginas. Es posible exportar el informe a más de 30 formatos de datos diferentes. Los más populares formatos de datos tales como PDF, Word y muchos otros. También se puede guardar el informe con el fin de verlo en el futuro utilizando el formato de archivo interno de Stimulsoft Reports.Ultimate. Además es posible cifrar el informe. (8)

1.3.2 Spring Framework

Pentaho Reporting Enterprise Edition es una opción preferida para los desarrolladores, integradores de sistemas y compañías de software que buscan ofrecer robustas capacidades de informes basados en java. Está bien integrado con el Spring Framework.
Pentaho Reporting Enterprise Edition permite a las organizaciones acceder fácilmente, dar formato y distribuir información a los empleados, clientes y socios. Pentaho Reporting permite el acceso a la información en OLAP relacional, o fuentes de datos basados en XML, y proporciona una salida en formatos populares, incluyendo Adobe PDF, HTML, Microsoft Excel, formato de texto enriquecido, o llano. Otras funciones de formato dan los diseñadores de informes la capacidad de resaltar excepciones, crear grupos, agregar totales y subtotales. (9)

Ventajas

Mayor rapidez en el desarrollo y entrega de nuevas aplicaciones.

Mayor estabilidad de la aplicación con reducción del tiempo de pruebas.
Aplicaciones visuales y de presentación de la información más ricas.
Mayor funcionalidad y productividad en las características de presentación de informes.
Un mayor valor de negocio en las aplicaciones de usuario final a través de informes integrados.

Características
Informes completamente integrables y extensibles basados en el motor de Java.
Integración con Spring Framework.
Interfaz personalizable. (9)

1.4 Proceso de desarrollo de software

Los elementos y relaciones de un proceso de desarrollo de software deben responder Quién debe hacer Qué, Cuándo y Cómo. Esto se logra modelando las interacciones y relaciones que suceden entre los roles, las actividades que estos desarrollan y los artefactos que se generan o actualizan durante el proceso.

Quién: Las personas participantes en el proyecto de desarrollo desempeñando uno o más roles específicos.

Qué: Un artefacto es producido por un rol como resultado del desarrollo de sus actividades. Los artefactos se especifican utilizando notaciones. Las herramientas apoyan la elaboración de artefactos.

Cómo y Cuándo: Las actividades son una serie de pasos que lleva a cabo un rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos artefactos.

El Centro para la Informatización de Gestión de Entidades (CEIGE) utiliza su propio modelo de desarrollo de software que describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones. Es un modelo basado en componentes, iterativo e incremental y utiliza técnicas de prototipado con el objetivo de desarrollar una definición mejorada de los requisitos del usuario para el sistema.

Desarrollo iterativo e incremental: Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una

parte del sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada iteración hasta obtener el producto de software completo en la última iteración. En cada iteración se obtiene como resultado un incremento. (10)

Desarrollo basado en componentes: Lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos a aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo. (10)

1.4.1 Artefactos y roles

El modelo de desarrollo define una serie de roles y artefactos que genera cada rol que son mostrados en el siguiente gráfico.

Roles	Artefactos
Jefe de Línea de Desarrollo	Plan de Iteración Plan de Gestión de Riesgos Plan de Trabajo individual de los integrantes de la Línea
Planificador	Plan de Iteración Plan de Trabajo Individual de los profesionales Definición del cronograma de desarrollo
Arquitecto de Sistema	Plan de Trabajo Individual Diagrama de componentes Prioridad de los componentes Agrupación Requerimientos - Componentes Informe de Integración
Arquitecto de Datos	Plan de Trabajo Individual Modelo de Datos Descripción del Modelo de Dato
Analista Principal	Plan de Trabajo Individual Mapa de Procesos de la Línea
Analista	Plan de Trabajo Individual Modelo de procesos de negocio Descripción de procesos de negocio Modelo Conceptual

	Prototipo de IU Especificación de requisitos. Casos de Prueba
Especialista de Calidad	Plan de Trabajo Individual Plan de pruebas Casos de Prueba Registro de No Conformidades
Especialista Funcional	Plan de Trabajo Individual Casos de Prueba Validación de Procesos y Requisitos
Desarrollador	Plan de Trabajo Individual Implementación de componentes Descripción de los componentes
Diseñador de Lógica del Negocio	Diagrama de Clases Descripción del Diseño de Clases

1.5 Tecnologías empleadas

Las tecnologías son un conjunto de conocimientos técnicos que son utilizados para el desarrollo de software, que posibilitan la satisfacción y adaptación de las necesidades a las que se enfrentan los desarrolladores. En el caso del centro CEIGE fueron definidas algunas que serán descritas a continuación entre las que se encuentran los lenguajes de programación y los marcos de trabajo a utilizar.

1.5.1 Lenguajes de programación

JavaScript

Javascript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. Muchos confunden el Javascript con el Java pero ambos lenguajes son diferentes y tienes características singulares. Javascript tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. Se puede decir que Javascript es un lenguaje interpretado, basado en prototipos. Es utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con Javascript se pueden crear diferentes efectos e interactuar con los usuarios. Este lenguaje posee varias características, entre ellas podemos mencionar

que es un lenguaje basado en acciones que posee menos restricciones. Además, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros. Es necesario resaltar que hay dos tipos de JavaScript: por un lado está el que se ejecuta en el cliente, este es el Javascript propiamente dicho, aunque técnicamente se denomina Navigator JavaScript. Pero también existe un Javascript que se ejecuta en el servidor, es más reciente y se denomina LiveWire Javascript. (11)

XML

XML (eXtensible Markup Language), no es, como su nombre podría sugerir, un lenguaje de marcado. Es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados. Originalmente diseñado y desarrollado por el W3C para afrontar los retos de la publicación electrónica a gran escala, XML también está desempeñando un papel cada vez más importante en el intercambio de una amplia variedad de datos en la Web y en otros lugares. Su aplicación no está solamente enmarcada en Internet a pesar de ser este el medio donde más se utiliza. También puede ser utilizado en bases de datos, editores de texto, hojas de cálculo, entre otras aplicaciones. Sus creadores lo proponen como un estándar para el intercambio de información estructurada entre diferentes plataformas. (12)

PHP

PHP, acrónimo recursivo que significa PHP Hypertext Pre-processor, es un lenguaje de programación de alto nivel, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en la interpretación del lado del servidor. Posee un gran parecido con otros lenguajes de programación estructurada como C y Perl. Es multiplataforma, por lo que las aplicaciones creadas con este lenguaje pueden ser migradas de un sistema operativo a otro sin complicaciones. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso y posee una gran comunidad de desarrolladores que intercambian experiencias, de esta forma cuando se presenta un problema, es muy fácil obtener documentación para darle solución de forma rápida y sin costo alguno. Quizás una de sus mayores desventajas radica en que promueve la creación de código desordenado, por lo que lo hace muy complejo de mantener. (13)

1.5.2 Lenguaje de modelado

UML 2.1

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para emplearse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (14)

1.5.3 Librerías y Marcos de trabajo

Zend Framework 1.8

Zend Framework es un framework de código abierto para desarrollar aplicaciones web. Utiliza código 100% orientado a objetos. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. Ofrece un gran rendimiento y una robusta implementación MVC (Modelo-Vista-Controlador), una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (15)

Doctrine 0.11

Doctrine es un potente y completo sistema ORM (“mapeador relacional de objetos”, por sus siglas en inglés) para PHP con un DBAL (siglas en inglés de “capa de abstracción de bases de datos”) incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Se encuentra en la parte superior de una poderosa Capa de Abstracción de Base de Datos (DBAL). Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL llamada Doctrine Query Language (DQL), inspirada en Hibernate HQL. Proporcionando a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. (16)

Ext JS 2.2

Es una librería Java Script de código abierto y alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas y multiplataforma. Proporciona ricas interfaces de usuario, con muchas facilidades de uso y similitudes a las aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es compatible con la mayoría de los navegadores, evitando tener que validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note. (17)

Sauxe

Sauxe 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. Está desarrollado sobre lenguaje de programación PHP y en la capa de presentación utiliza Java script y HTML. Implementa una arquitectura en capas aunque en una de sus capas contenga un modelo vista controlador. En la capa de presentación utiliza ExtJs por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional. En la capa de negocio

utiliza Zend Framework por su nivel de flexibilidad en la integración con la capa superior, inferior y vertical de la arquitectura, otra de las ventajas que brinda es la sencillez a la hora de extender sus componentes, de estas extensiones surge ZenExt Framework desarrollada por el Centro Soluciones Gestión y el Centro de Desarrollo y Asimilación de Tecnologías de la UCID con el objetivo de crear un marco de trabajo extensible y configurable centrando el desarrollo de las aplicaciones en la lógica del negocio y en las interfaces de usuario y alejando a los programadores de los detalles arquitectónicos, con soporte para entornos multi-entidad y multi-sistema para una arquitectura de sistema orientada a componentes. (18)

1.6 Herramientas de desarrollo

Visual Paradigm 6.4

Visual Paradigm para UML es una herramienta UML (Lenguaje Unificado de Modelado, por sus siglas en inglés) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a agilizar la construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Soporta UML versión 2.1, permite modelado colaborativo con CVS y Subversion, generación de código, ingeniería inversa, generación de bases de datos (transformación de diagramas entidad-relación en tablas de la base de datos), importación y exportación a ficheros XML, distribución automática de diagramas, entre otras características. (19)

PostgreSQL 8.3

PostgreSQL es un sistema de gestión de bases de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las

características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Incluye características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos. (20)

Zend Studio

Zend Studio es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Se ha diseñado para maximizar la productividad de los desarrolladores por lo que le permite desarrollar y mantener el código de una forma más rápida, así como resolver los problemas de aplicación y mejorar la colaboración en equipo. Zend Studio fue diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, Javascript y XML. Entre sus características se encuentran algunas como el soporte para PHP 4 y PHP 5, también está Plegado de código (comentarios, bloques de phpDoc, cuerpo de funciones y métodos e implementación de clases). Soporte para control de versiones usando CVS o Subversion (a elección del desarrollador) y soporte para navegación en bases de datos y ejecución de consultas SQL. (21)

Apache 2.2.9

Es un software que permite crear en un ordenador, de una sencilla y rápida forma, un servidor de Protocolos de Transferencia de Hipertexto (HTTP), el cual se encarga de transferir los hipertextos, páginas web o páginas HTML, textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. El Apache HTTP Server es un servidor robusto, de múltiples características y funcionalidades, una herramienta gratuita y de código abierto, lo que proporciona transparencia en todo el proceso de instalación.

Principales características con las que cuenta (22):

- Configuración basada en un poderoso archivo (*httpd.conf*).
- Soporte del protocolo HTTP y autenticación basada en la web.

- Soporte de host virtuales: Apache es uno de los primeros servidores web en soportar tanto host basados en IP como host virtuales.
- Integración de Perl.
- Soporte de scripts PHP: Apache ofrece un amplio soporte de PHP utilizando el módulo *mod_php*.
- Soporte de servlets de Java: Puede ejecutar servlets de Java utilizando el premiado entorno Tomcat con Apache.
- Servidor proxy integrado.
- Estado del servidor y adaptación de registros Apache le da una gran cantidad de flexibilidad en el registro y la monitorización del estado del servidor. El estado del servidor puede monitorizarse mediante un navegador web.
- Soporte de Server Side Includes (SSI), Secured Sockect Layer (SSL), Common Gateway Interface (CGI).
- Soporte de FastCGI.

Dadas las características se puede decir que un servidor Apache es altamente configurable y de diseño modular. Posibilita que los administradores de sitios web puedan elegir los módulos que serán incluidos y ejecutados en el servidor. Permite personalizar los mensajes de errores, la creación y gestión de logs, y la negociación de contenido, de este modo es posible tener un mayor control sobre lo que sucede en el servidor. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. La aplicación es prácticamente universal ya que permite ejecutarse en múltiples sistemas operativos como Windows, Novell NetWare y Mac OS X.

NetBeans 6.9

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. (23)

Ventajas:

- Auto-completado y documentación de funciones PHP: Rápido acceso a la documentación de PHP, y si se necesita más información se provee el link directo a la función.
- Auto-completado de código propio: Esto es una consecuencia del punto anterior, al documentar código con el formato esperado, estos serán mostrados.
- Atajos de teclado muy útiles: Permite a través de varios comandos la ejecución de numerosas funcionalidades y existen muchas más como integración con Xdebug, soporte para Symfony, Zend Framework, Smarty, historia local para archivos. Entre otros.

1.7 Conclusiones parciales

En este capítulo se realizó un estudio del estado del arte con el objetivo de ver en el ámbito nacional e internacional, cómo otras aplicaciones solucionan o tratan el tema de la visualización y configuración de documentos primarios. Se hizo una búsqueda entre los marcos de trabajo más utilizados a nivel mundial para adquirir experiencia en el tema, ver posibles soluciones y hacer una valoración crítica al respecto. Además se especifican las herramientas, tecnologías y modelo de desarrollo a emplear para dar solución al problema presentado.

La visualización y configuración de documentos primarios no es tratada en sí en ningún marco de trabajo de los que fueron estudiados, aunque individualmente algunos de ellos tratan otro tipo de documentos, realizan reportes y los muestran ocasionalmente de forma estandarizada. A pesar de esto, esas soluciones no se ajustan a las necesidades del centro CEIGE por la plataforma donde corren, y algunas peculiaridades como los tipos de documentos que se muestran, la no posibilidad de configuración de los mismos por parte de los usuarios, su accesibilidad y estandarización. Por tanto es necesario que se implemente un componente que permita la visualización y configuración de los documentos primarios en el marco de trabajo Sauxe, lo cual permita además que los mismos sean mostrados de forma estandarizada.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

En este capítulo se presentarán algunos artefactos generados como parte del proceso de desarrollo para dar solución al problema presentado. Entre los artefactos se encuentra la descripción de los procesos del negocio y los requisitos funcionales que debe contener el componente que será implementado. También será generado el modelo conceptual y otros artefactos según el modelo de desarrollo del centro CEIGE.

2.1 Descripción de procesos del negocio

Un proceso de negocio es un tipo especial de proceso que describe, desde un punto de vista orientado al mercado, las actividades de una organización. El principal objetivo de los procesos de negocio es satisfacer necesidades de los clientes.

El modelado de procesos de negocio resulta útil en una gran variedad de situaciones, que pueden ser clasificadas en tres grupos: descripción del proceso, análisis del proceso e implementación del proceso. Actualmente se utilizan muy diversas técnicas de modelado, como redes de Petri, diagramas de actividad de UML y otros lenguajes, fundamentalmente gráficos y formales.

Una buena descripción de los procesos del negocio significará que el equipo de desarrollo ha comprendido claramente el funcionamiento del mismo y que podrán ofrecer una solución adecuada al problema, para que la solución este realmente en correspondencia con las necesidades de los clientes.

2.1.1. Descripción de proceso: Visualizar documento primario.

La figura 1 muestra el diagrama del proceso de visualización de documentos primarios, en el que el usuario que solicita consultar un documento primario, cuando no lo tiene a su alcance, lo solicita a otro trabajador para que se lo envíe y así poder consultarlo.

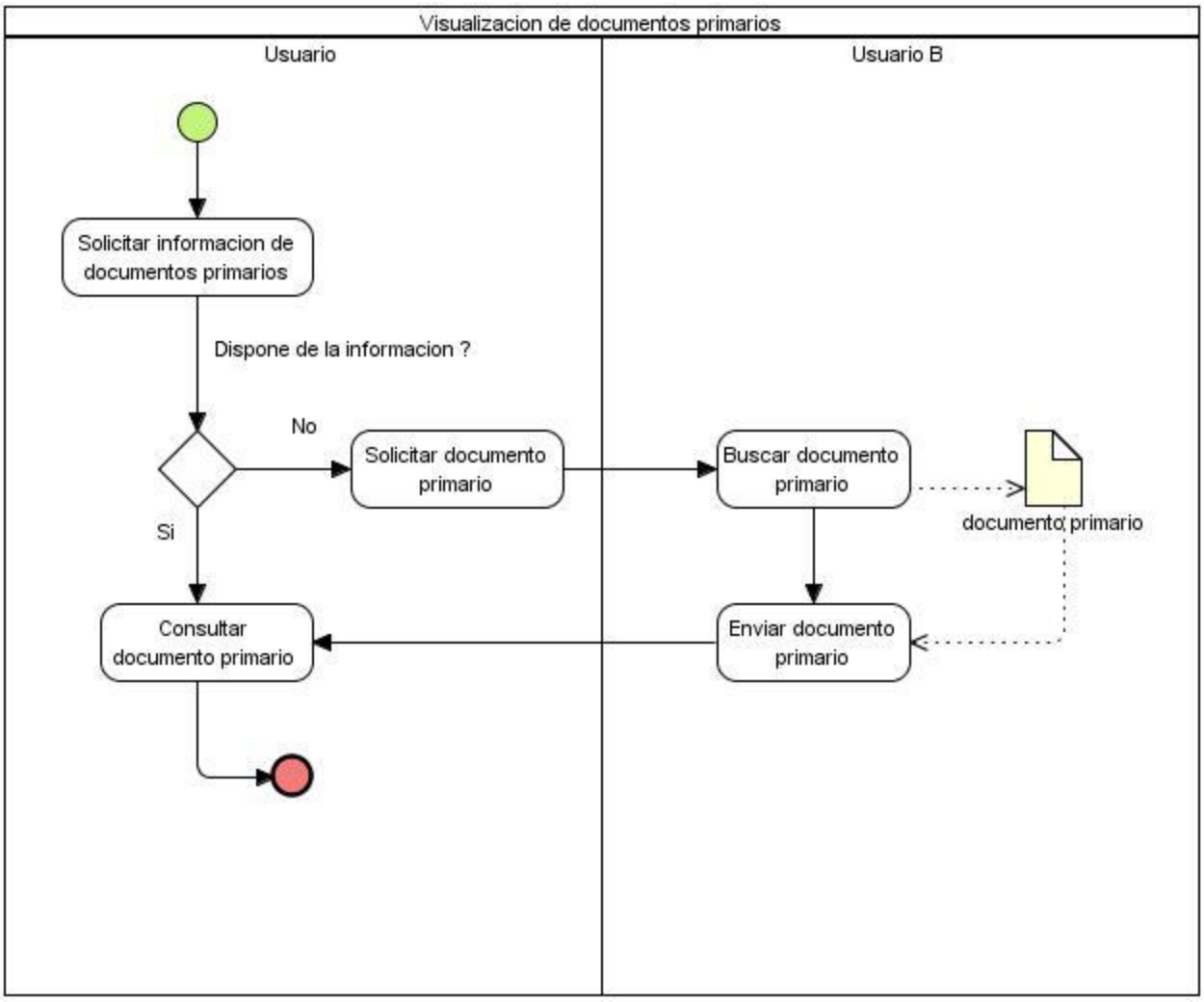


Figura 1: Descripción de procesos del negocio. Visualizar documento primario.

2.2 Requisitos de software

Comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta las diversas necesidades de los clientes. Existen varias técnicas para la captura de requisitos, entre las que se encuentran, las entrevistas, los talleres, los prototipos y los casos de uso. Los requisitos se pueden dividir en funcionales y no funcionales. Los funcionales son los que el usuario necesita que efectúe el software y los no funcionales

son los "recursos" necesarios para que trabaje el sistema e imponen restricciones al diseño o funcionamiento del mismo. (24)

2.2.1. Requisitos funcionales

Los requisitos funcionales son declaraciones de las funcionalidades que debe proporcionar el sistema. Especifican la manera en que éste debe reaccionar a determinadas entradas y el comportamiento del sistema en determinadas situaciones. También pueden declarar explícitamente lo que el sistema no debe hacer. (24)

Para el sistema propuesto fueron identificados los siguientes requisitos funcionales (RF).

- ✓ RF1: Listar documento primario.
- ✓ RF2: Adicionar documento primario
- ✓ RF3: Modificar documento primario.
- ✓ RF4: Mostrar documento primario.
- ✓ RF5: Buscar documento primario.
- ✓ RF6: Eliminar documento primario.

Requisito funcional: Listar documentos primarios.

Conceptos tratados	**Conceptos**	**Atributos**
	Documentos primarios.	Nombre, subsistema, servicio
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos documento primario.	
Descripción	Para listar los documentos primarios debe existir al menos uno en el XML que son almacenados, y de existir alguno serán listados de forma automática.	
Validaciones	N/A	

Post-condiciones	No procede.
Post-requisito	No procede.

Tabla 1: Descripción de requisitos: Listar documentos primarios

Prototipo de interfaz de usuario: Listar documentos primarios.

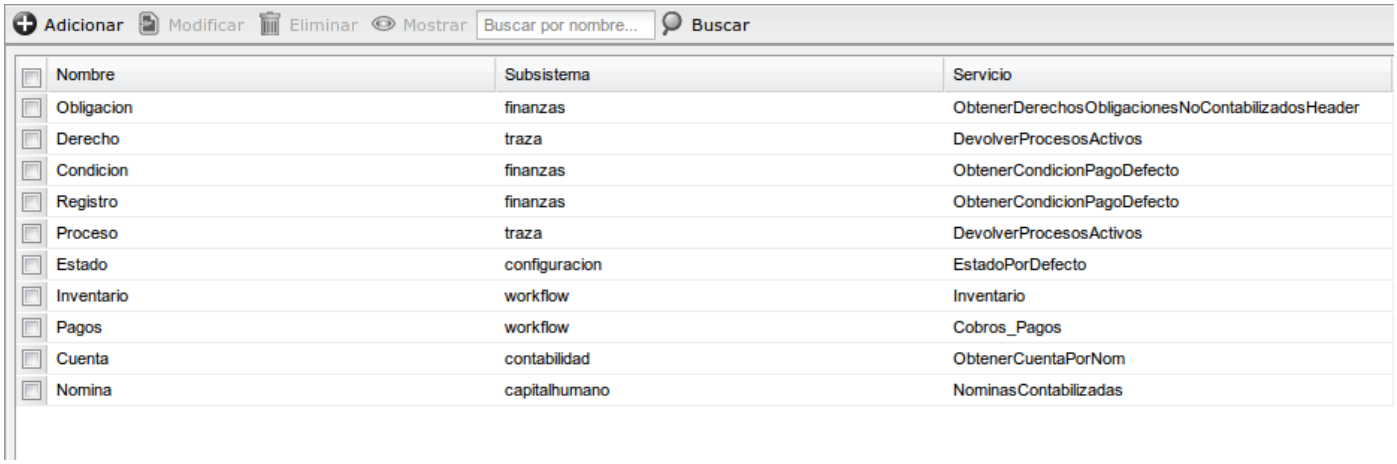


Figura 2: Prototipo de Interfaz. Listar documentos primarios.

Requisito funcional: Adicionar documento primario.

Conceptos tratados	**Conceptos**	**Atributos**
	Documento primario.	Nombre, subsistema, servicio, campos y fuente de los mismos.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un servicio.	

Descripción	Para adicionar un documento primario el usuario debe especificar el nombre del documento primario, el subsistema y el servicio del mismo del cual este nuevo documento obtendrá valores, de los cuales serán definidos su nombre y la fuente de la que proviene, siendo esta una ruta que indica específicamente un parámetro del servicio. El sistema debe verificar la completitud de los datos y validar los mismos. En caso en que no ocurran errores el sistema informa que se adiciono correctamente un nuevo documento primario. De ocurrir algún error el sistema informa del mismo y pide su corrección. El sistema debe permitir la cancelación de esta acción.
Validaciones	No se permite la adición de documentos primarios con el mismo nombre de uno ya existente. No se permiten valores nulos en el campo nombre. No se permite la no selección de un subsistema y un servicio.
Post-condiciones	El sistema actualiza la lista de documentos primarios.
Post-requisito	Listar documentos primarios.

Tabla 2: Descripción de requisitos: Adicionar documento primario.

Prototipo de interfaz de usuario: Adicionar documento primario.

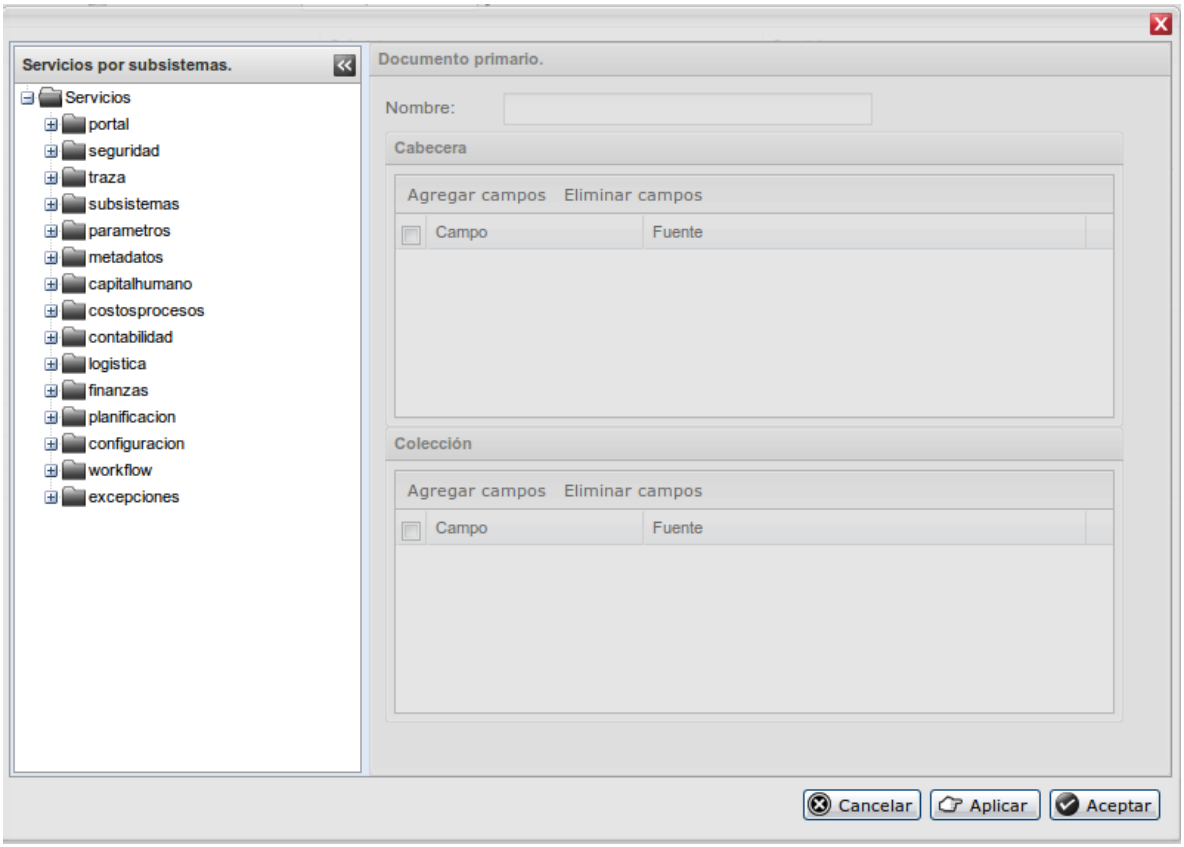


Figura 3: Prototipo de interfaz. Adicionar documento primario.

Requisito funcional: Modificar documento primario.

Conceptos tratados	**Conceptos**	**Atributos**
	Documento primario.	Subsistema, servicio, campos y fuente de los mismos.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un documento	Adicionar documento primario.

	primario.	
Descripción	Para modificar un documento primario el usuario debe seleccionar el documento que desee modificar. El sistema mostrará un formulario con los datos que posee actualmente el documento, todos los datos pueden ser modificados excepto el campo nombre que es el identificador de los documentos. El sistema debe verificar la completitud de los datos y validar los mismos. En caso en que no ocurran errores el sistema informa que se modificó correctamente un nuevo documento primario. De ocurrir algún error el sistema informa del mismo y pide su corrección. El sistema debe permitir la cancelación de esta acción.	
Validaciones	No se permite el cambio de nombre de un documento primario. No se permite la no selección de un subsistema y un servicio.	
Post-condiciones	El sistema actualiza la lista de documentos primarios.	
Post-requisito	Listar documentos primarios.	

Tabla 3: Descripción de requisitos: Modificar documento primario.

Prototipo de interfaz de usuario: Modificar documento primario.

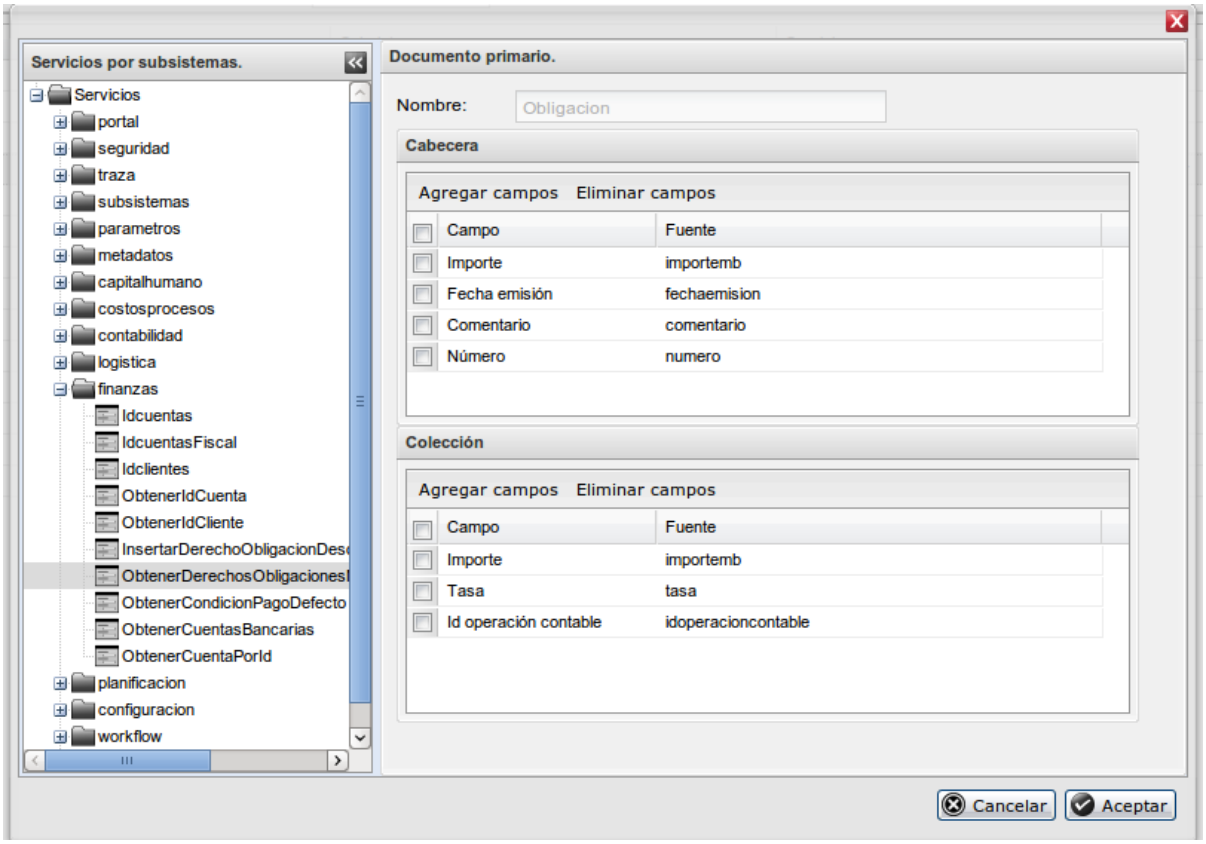


Figura 4: Prototipo de interfaz. Modificar documento primario.

Requisito funcional: Mostrar documento primario.

Conceptos tratados	Conceptos	Atributos
	Documento primario.	Nombre, subsistema, servicio, campos y fuente de los mismos.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un documento	Adicionar documento primario.

	primario.	
Descripción	Para mostrar un documento primario el usuario debe seleccionar el tipo de documento primario a mostrar. Posteriormente el sistema mostrará una ventana solicitando el id del documento primario a mostrar y al hacerlo y enviarlo el sistema mostrará el documento para que pueda ser consultado. El usuario debe poder cancelar la operación.	
Validaciones	No procede.	
Post-condiciones	No procede.	
Post-requisito	Listar documentos primarios.	

Tabla 4: Descripción de requisitos. Mostrar documento primario.

Prototipo de interfaz de usuario: Mostrar documento primario.

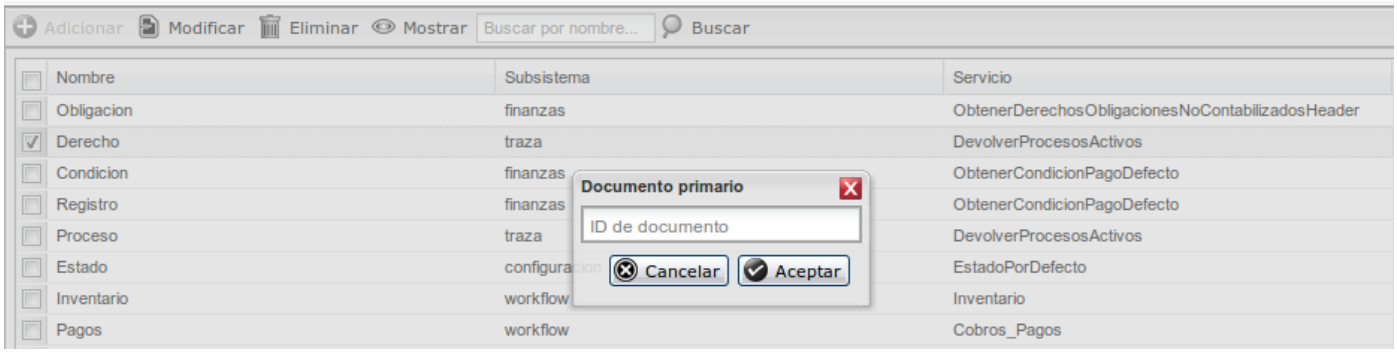


Figura 5: Prototipo de interfaz. Mostrar documento primario.

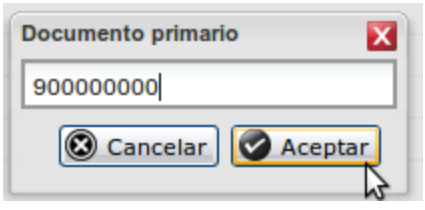


Figura 6: Prototipo de interfaz. Mostrar documento primario. Insertar identificador.

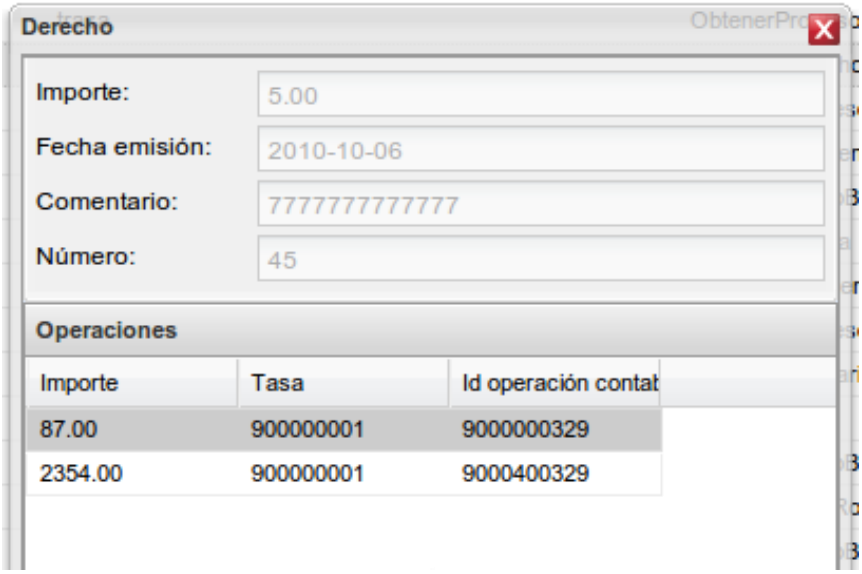


Figura 7: Prototipo de interfaz. Mostrar documento primario. Mostrar datos.

Requisito funcional: Buscar documento primario.

Conceptos tratados	Conceptos	Atributos
	Documento primario.	Nombre.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un documento primario.	Adicionar documento primario.
Descripción	Para buscar un documento primario el usuario debe introducir el parámetro de búsqueda en el campo destinado para ello. El sistema mostrará los resultados de dicha búsqueda y si el usuario desea puede modificarlo o eliminarlo.	
Validaciones	No procede.	
Post-condiciones	No procede.	
Post-requisito	Listar documentos primarios.	

Tabla 5: Descripción de requisitos. Buscar documento primario.

Prototipo de interfaz visual: Buscar documento primario.

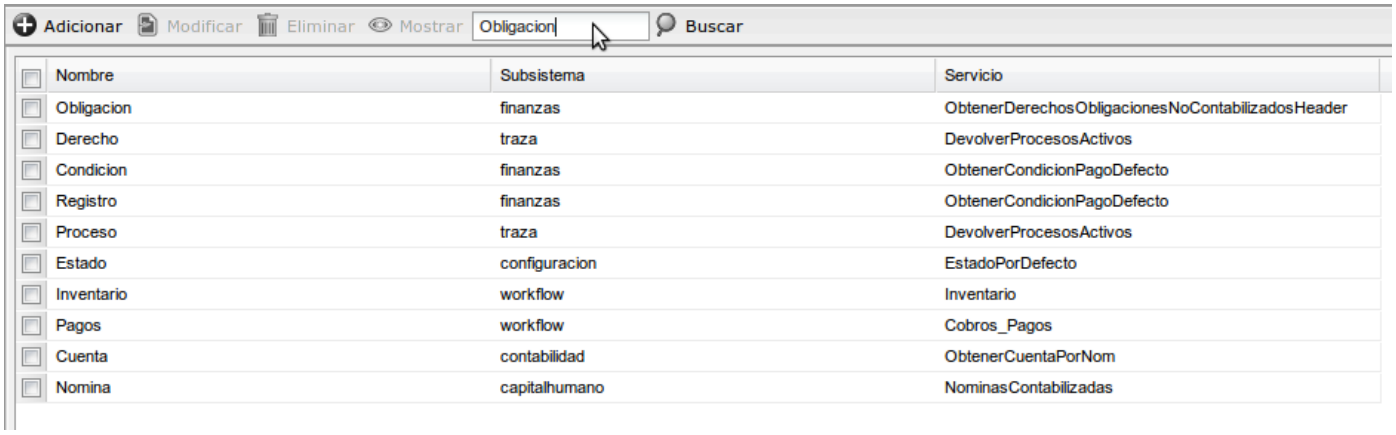


Figura 8: Prototipo de interfaz. Buscar documento primario.

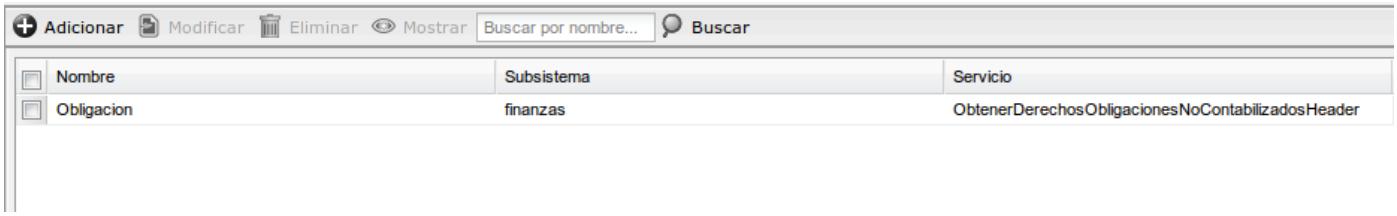


Figura 9: Prototipo de interfaz. Resultado de la búsqueda.

Requisito funcional: Eliminar documento primario.

Conceptos tratados	**Conceptos**	**Atributos**
	Documento primario.	Nombre.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un documento primario.	Adicionar documento primario.

Descripción	Para eliminar un documento primario el usuario debe seleccionar dicho documento primeramente y luego dar clic en el botón eliminar. El sistema mostrará los resultados de dicha búsqueda y si el usuario desea puede modificarlo o eliminarlo.
Validaciones	No procede.
Post-condiciones	No procede.
Post-requisito	Listar documentos primarios.

Tabla 6: Descripción de requisitos. Eliminar documento primario.

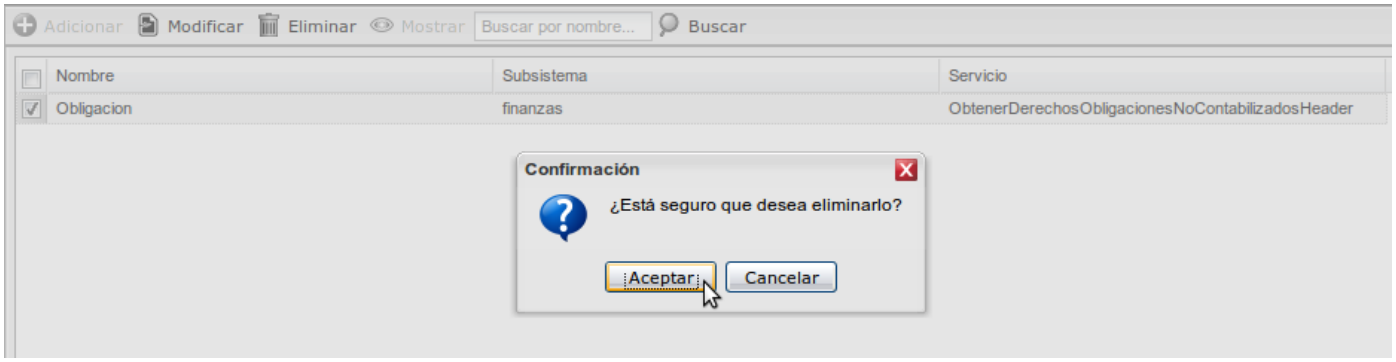


Figura 10: Prototipo de interfaz. Eliminar documento primario.

2.5.1 Requisitos no funcionales

No se refieren a funciones específicas que proporciona el sistema. Son restricciones de los servicios o funciones ofrecidas por el sistema (fiabilidad, tiempo de respuestas, capacidad de almacenamiento, etc.) Generalmente se aplican al sistema en su totalidad. Surgen de las necesidades del usuario (restricciones de presupuesto, políticas de la organización, necesidad de interoperabilidad, etc.)

Apariencia o interfaz externa. El sistema debe presentar una interfaz agradable, sencilla y fácil de usar, de tal forma que el usuario se sienta a gusto en él y pueda explotar al máximo las funcionalidades que brinda para agilizar su trabajo.

Para el sistema propuesto fueron identificados los siguientes requisitos no funcionales (RNF).

Usabilidad

El sistema propuesto estará estructurado de forma sencilla, de manera que pueda ser usado por personas con conocimientos mínimos en el trabajo con una 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

El sistema debe garantizar el control en el acceso, utilizando la autentificación y autorización mediante contraseñas. Además debe garantizar la 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 realizará de forma centralizada por el administrador.

Requerimiento de software

Para el cliente:

- Navegador Mozilla Firefox 3.0 o superior.
- Sistema operativo Windows XP o superior o Linux en cualquiera de sus distribuciones.

Para el servidor:

- Sistema operativo Linux en cualquiera de sus distribuciones.
- Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida.
- Un servidor de base de datos PostgreSQL 8.3.

Requerimientos de hardware

Para el cliente:

- Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM.
- Tarjeta de red.

Para el servidor:

- Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- Al menos 40Gb de espacio libre en disco duro.
- Tarjeta de red.

2.3 Modelo Conceptual

Un Modelo Conceptual puede verse como un mapa de conceptos y las relaciones entre ellos. Este modelo requiere cierto nivel de abstracción, concentrándose en aspectos considerados fundamentales para la comprensión de la situación representada. Este modelo permite una mejor comprensión de lo que está siendo analizado.

Para el problema en cuestión el resultado se expresa en la figura 8.

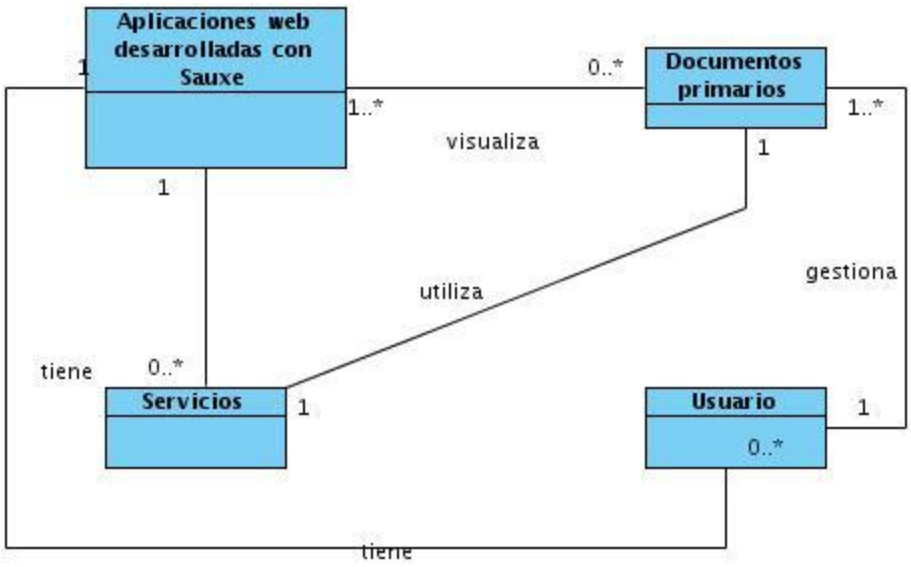


Figura 11: Modelo conceptual

Como muestra la figura 8, los subsistemas tienen de uno a muchos servicios, los cuales son utilizados por los documentos primarios para obtener los datos. Los documentos primarios son gestionados por los usuarios, los cuales solicitan su visualización con determinados datos previamente configurados por el

mismo u otro usuario. La información contenida en un documento primario hace referencia a los datos del mismo referentes a un subsistema.

2.4 Patrones

Patrón arquitectónico Modelo-Vista-Controlador (MVC)

El patrón de arquitectura de modelo-vista-controlador (MVC) divide una aplicación interactiva en tres partes.

El modelo representa las estructuras de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.

La **Vista** es la información presentada al usuario. Puede ser una página Web o una parte de una página.

El **Controlador** actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página, es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo.

Este patrón es empleado en la capa de control del marco de trabajo Sauxe y determina la estructura de los paquetes internos de los componentes a desarrollar.

2.5 Modelo de diseño

El modelo de diseño consiste en producir una representación técnica del software que se va a desarrollar. El diseño es el proceso sobre el que se asienta la calidad del software. Es un proceso iterativo a través del cual se traducen los requisitos en una representación del software. Representa a un alto nivel de abstracción, un nivel que se puede seguir hasta requisitos específicos de datos, funcionales y de comportamiento.

Es un modelo físico (plano de la implementación). No es un modelo genérico, puesto que se realiza específicamente para un sistema y una implementación determinada. Debe ser mantenido durante todo el ciclo de vida del software.

2.5.1 Diagrama de clases

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia y de uso
Un diagrama de clases está compuesto por los siguientes elementos:
Clase: atributos, métodos y visibilidad.
Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

El diagrama de la figura 12 muestra como la página cliente está compuesta por archivos java script los cuales representan la interfaz de usuario de la aplicación. Esta interfaz está compuesta por el formulario, el cual registra los datos que serán enviados al servidor para su manipulación. Finalmente la clase del modelo de datos provee la información solicitada por la controladora y esta se encarga de que la página cliente reciba la misma, separando la interfaz de la controladora y el modelo de datos.

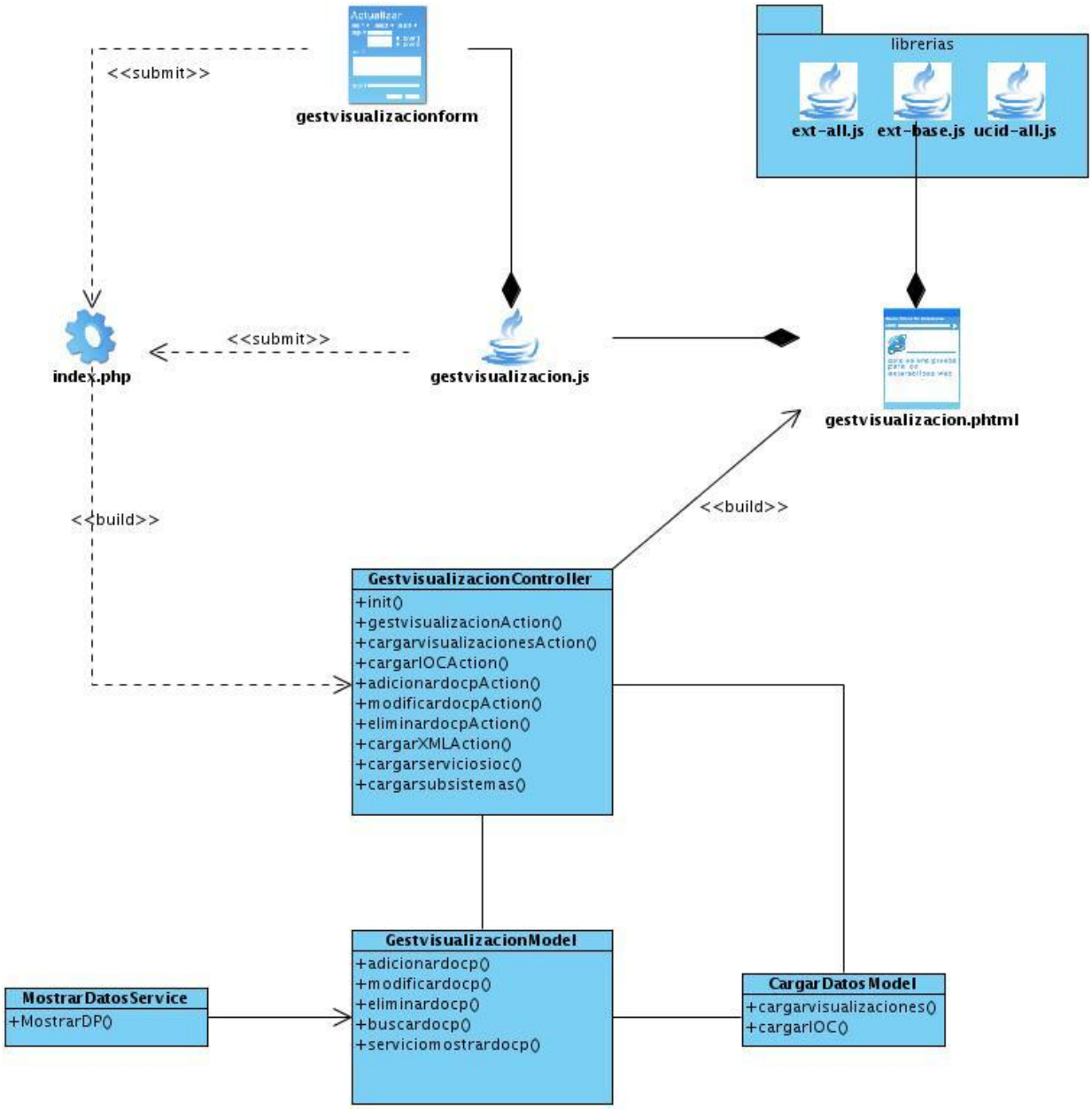


Figura 12: Diagrama de clases.

2.6 Conclusiones parciales

En este capítulo fueron expuestos los artefactos generados durante el diseño de la solución propuesta para el componente de configuración y visualización de documentos primarios en el marco de trabajo Sauxe. También fueron generados los artefactos que propone el modelo de desarrollo orientado a componentes por el cual se rige el equipo de desarrollo, entre los cuales se encuentran: El diagrama de procesos del negocio, el modelo conceptual y el diagrama de clases del diseño. Son descritos los requisitos funcionales y no funcionales que debe cumplir el componente.

Por tanto los objetivos planteados para la realización del presente capítulo han sido cumplidos, ya que resultó en el análisis y diseño de un componente para la configuración y visualización de los documentos primarios.

Una vez concluido el diseño de la solución está lista la base para dar paso a los flujos de implementación y prueba de la misma.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

En este capítulo se realiza el modelo de implementación que da cumplimiento al diseño de la solución realizado en el capítulo anterior, además se especifican un conjunto de validaciones y pruebas que evalúan la calidad del sistema.

3.1 Modelo de implementación

3.1.1 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura para facilitar la lectura, comprensión y mantenimiento del código.
En sistemas complejos como es el caso de Sauxe, y la gran cantidad de personas vinculadas en su desarrollo es importante la aplicación de todos los implicados de ciertos estándares de codificación con el fin de obtener un estándar en la implementación que permita asegurar la calidad del software, obteniendo un código más legible y reutilizable.
A continuación se describen algunos de estos estándares empleados durante la implementación del componente para la visualización de documentos primarios.

3.1.1.1 PascalCasing

El estándar PascalCasing establece que los identificadores, nombres de clases, variables, métodos o funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.
La nomenclatura de las clases del componente de visualización se realizó sobre la base de este estándar, usando palabras compuestas sugerentes acordes al propósito de la misma.

Nomenclatura de clases según su tipo

- **Controllers:** clases controladoras del negocio.

El nombre de la clase controladora debe estar estructurado por el nombre propio de la misma en mayúsculas seguido por la palabra Controller.
Ejemplo: GestionarVisualizacionController

- **Clases de los modelos**
  - ✓ **Business:** Clases modelo del negocio.

Las clases modelo tendrán por identificador el nombre de la tabla en la que trabajan seguido por la palabra Model.

Ejemplo:

GestVisualizacionModel.

  - ✓ **Domain:** clases entidades del dominio.

El sistema no cuenta con este tipo de clases porque los datos persisten en un XML, por tanto no se trabaja con bases de datos y se prescinde de esta clase.

  - ✓ **Generated:** Clases bases del dominio

El sistema no cuenta con este tipo de clases porque los datos persisten en un XML, por tanto no se trabaja con bases de datos y se prescinde de esta clase.

- **Services:** Clases que ofrecen los servicios de los componentes.

Estas clases, de acuerdo con las operaciones que realizan y prestaciones que brindan, se definen con calificativos sugerentes, agregándoles la terminación Service. Ejemplo: MostrarDocumentoService.

3.1.1.2 CamelCasing

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.

Nomenclatura de las funciones

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando la notación CamelCasing y nombres que deduzcan su propósito.

Ejemplo: adicionarDP().

Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: cargarVisualizacionesAction().

3.1.1.3 Notación húngara

Esta convención, también conocida como notación: REDDICK por el nombre de su creador, se basa en definir prefijos para cada tipo de datos según el ámbito de las variables con el fin de brindar mayor información al nombre de la variable, método o función.

La notación húngara fue utilizada para la definición de variables de acuerdo con los siguientes prefijos:

Tipos de datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
Float	flit
Boolean	bool

Tabla 7: Prefijos para la creación de variables.

Nomenclatura de las variables

El nombre a emplear para las variables se escribe siguiendo la notación CamelCasing comenzando con el prefijo según su tipo de datos.

Ejemplo: arrCabecera define un arreglo con los elementos de la cabecera de un documento primario.

3.2 Diagrama de componentes

El diagrama de componentes representa la estructura física del sistema, su agrupación por paquetes y muestra las dependencias entre estos.

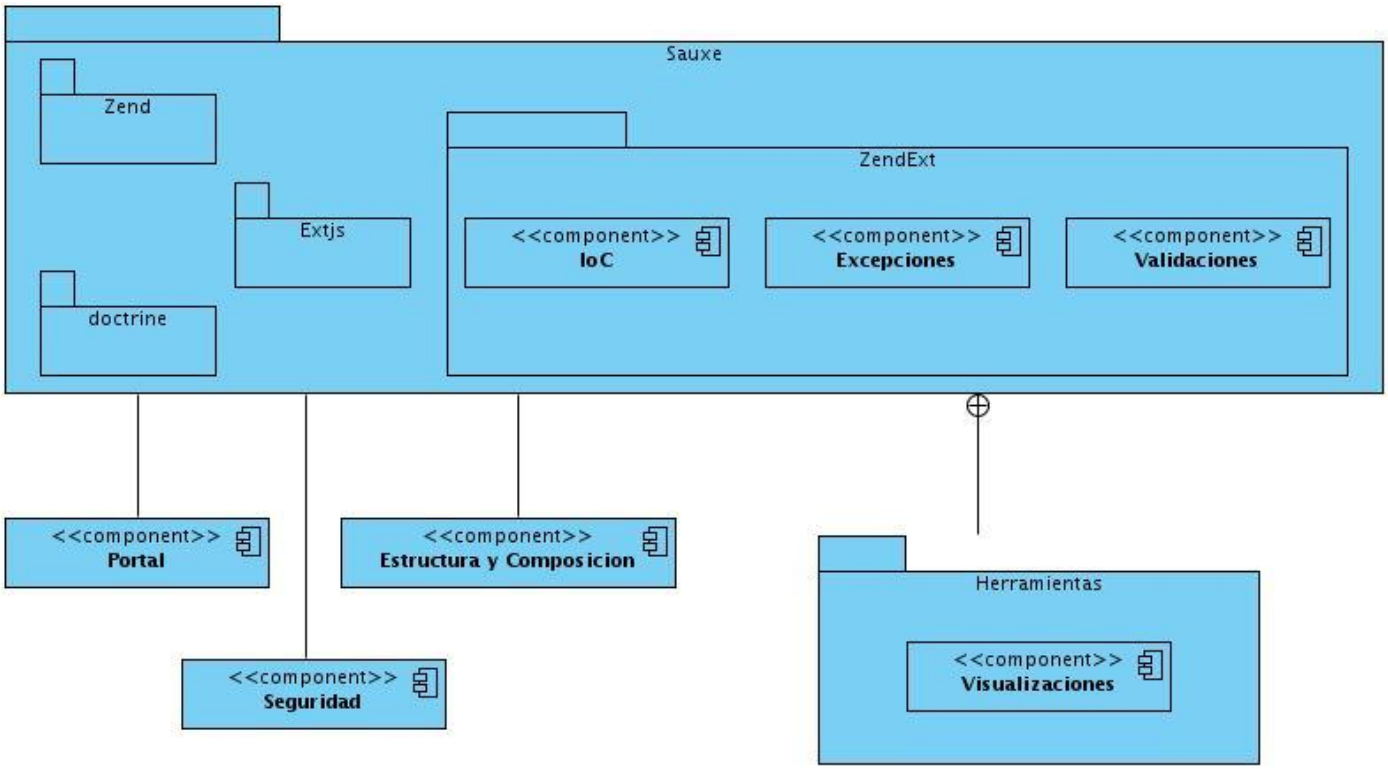


Figura 13: Diagrama de componentes.

3.3 Diagrama de despliegue

El Diagrama de Despliegue es un tipo de diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. En muchos casos el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema.

En este caso, el usuario, desde una computadora de trabajo, podrá acceder al sistema que se encontrará desplegado en un servidor web, donde mismo se encuentre ubicada la aplicación. El servidor estará conectado a un servidor de bases de datos en el cual se almacenará la información de interés para la solución. A continuación se muestra el diagrama de despliegue elaborado.

Figura 14: Diagrama de despliegue.

3.4 Métricas de diseño

Las métricas de software permiten a los desarrolladores evaluar cuantitativamente la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software.

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

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

Las métricas escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior y su relación con los atributos de calidad son las siguientes:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

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.

Tabla 8: Atributos de calidad evaluados por la métrica TOC.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

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

Tabla 9: Criterios de evaluación para la métrica TOC.

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

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 del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 10: Atributos de calidad evaluados por la métrica RC.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Tabla 11: Criterios de evaluación para la métrica RC.

3.4.1 Resultados obtenidos de la aplicación de la métrica TOC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 75% de las clases empleadas en el sistema poseen cinco operaciones o menos lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización). A continuación se muestran los resultados obtenidos.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
CargarDatosModel	2	Baja	Baja	Alta
GestVisualizacionModel	5	Media	Media	Media
MostrarDocumentoService	1	Baja	Baja	Alta
GestionarVisualizacionController	10	Alta	Alta	Baja

Tabla 12: Instrumento de evaluación de la métrica TOC.

Ilustración 1: Leyenda para los gráficos.

Figura 15: Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.

Figura 16: Resultados de la evaluación de la métrica TOC para el atributo Reutilización.

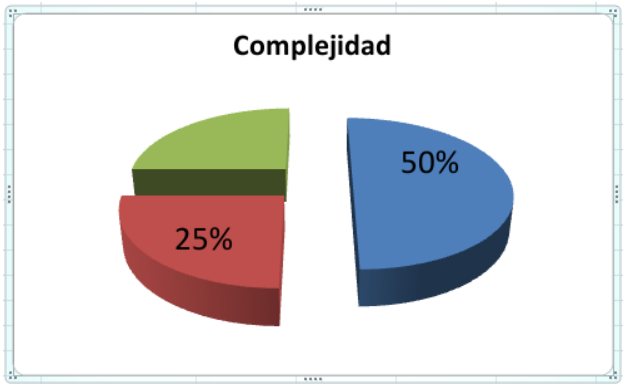


Figura 17: Resultados de la evaluación de la métrica TOC para el atributo Complejidad.

3.4.2 Resultados obtenidos de la aplicación de la métrica RC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 75% de las clases empleadas posee menos de 3 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). A continuación se muestran los resultados obtenidos.

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad mant	Reutilización	Cantidad de pruebas
CargarDatosModel	0	Ningun	Baja	Alta	Baja

		o			
GestVisualizacionModel	0	Ningun o	Baja	Alta	Baja
MostrarDocumentoService	1	Baja	Baja	Alta	Baja
GestionarVisualizacionController	7	Alto	Alta	Baja	Alta

Tabla 13: Instrumento de evaluación de la métrica TOC.

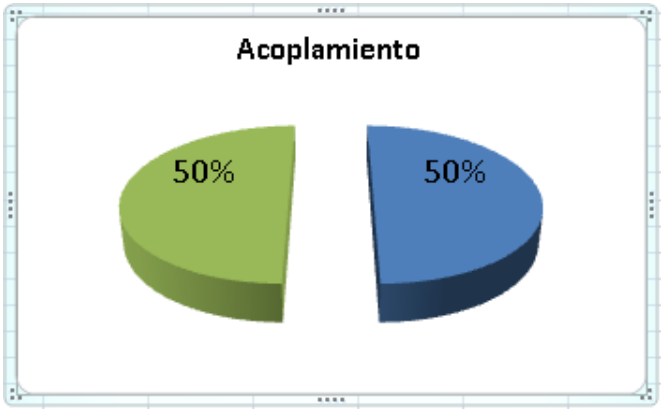


Figura 18: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

Figura 19: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

Figura 20: Resultados de la evaluación de la métrica RC para el atributo Reutilización.

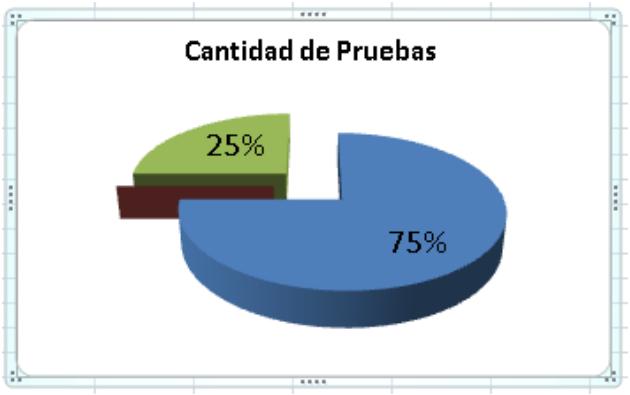


Figura 21: Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

3.4.3 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guión simple (-). Luego se puede obtener un resultado.

A continuación se muestran los resultados obtenidos.

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de Implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1
Cantidad de pruebas	(-)	1	1

Tabla 14: Resultados de la evaluación de la relación atributo/métrica.

Categoría	Rango de valores
Malo	<=0.4
Regular	>0.4 y <0.7
Bueno	>=0.7

Tabla 15: Rango de valores para la evaluación de la relación atributo/métrica.

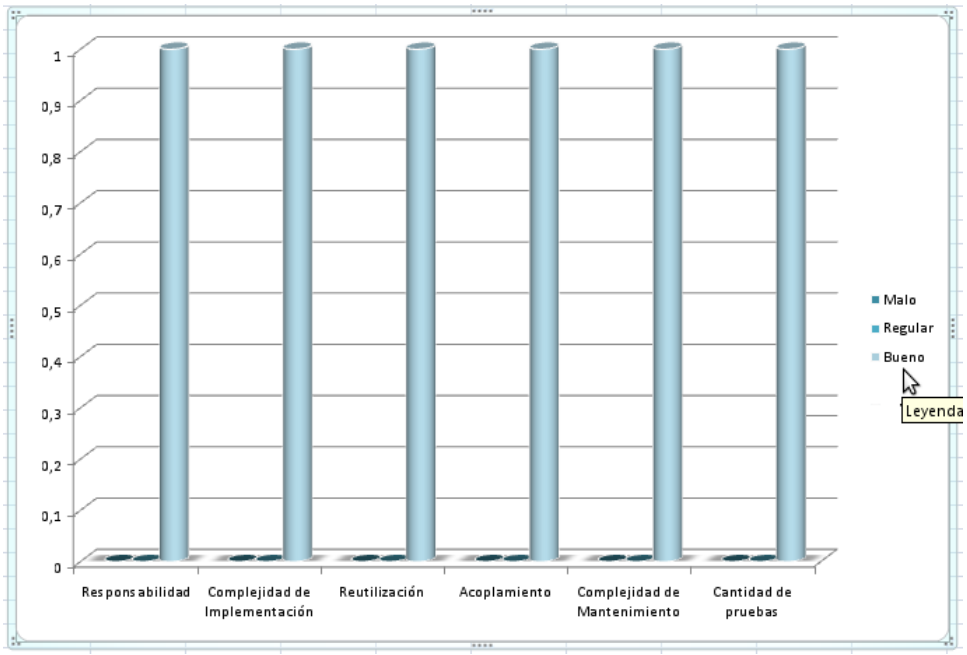


Figura 22: Resultados obtenidos de la evaluación de los atributos de calidad.

3.5 Pruebas

Los casos de prueba son un conjunto de condiciones o variables bajo las cuales se determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

Se pueden hacer muchos casos de prueba con el fin de comprobar si un requisito es satisfactorio o no, pero algunas metodologías indican que debe haber al menos un caso de prueba para cada requisito.
Si la aplicación es creada sin requisitos formales, entonces los casos de prueba se escriben basados en la operación normal de programas de una clase similar.

Los casos de prueba escritos, incluyen una descripción de la funcionalidad que se probará, la cuál es tomada ya sea de los requisitos o de los casos de uso, y la preparación requerida para asegurarse de que la prueba pueda ser dirigida. Los casos de prueba escritos se recogen generalmente en una suite de pruebas.

Se le deben realizar pruebas a todos los artefactos generados durante el proceso de desarrollo de un software, lo que incluye especificaciones de requisitos, diagramas y el código fuente.

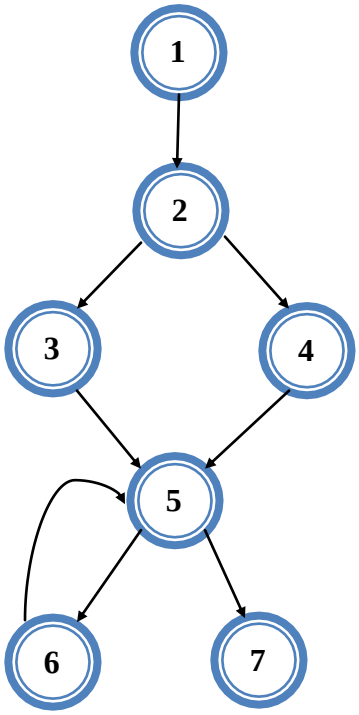
En el presente trabajo se realizaran pruebas a la implementación utilizando pruebas de caja blanca para verificar que el código presenta la calidad suficiente para un correcto funcionamiento del sistema implementado.

3.5.1 Pruebas estructurales o de caja blanca

Las llamadas pruebas de caja blanca, también conocidas como técnicas de caja transparente o de cristal, proponen una manera de diseñar los casos de prueba centrándose en el comportamiento interno y la estructura del programa. De esta manera se examina solo la lógica interna del programa, dejando fuera los aspectos de rendimiento del mismo.

El empleo de este tipo de pruebas permitirá diseñar casos de prueba que comprueben que todas las sentencias del sistema se ejecuten al menos una vez, además de todas las condiciones, tanto verdaderas como falsas. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
237
238     public function cargarvisualizacionesAction() {
239         $limit = 20; $start = 0; $dir = explode('/index.php', $_SERVER['SCRIPT_FILENAME']); // 1
240         $dir = str_replace('/', DIRECTORY_SEPARATOR, $dir[0]); $dir = str_replace('web', 'apps', $dir); // 1
241         $direccion = $dir . DIRECTORY_SEPARATOR . "comun" . DIRECTORY_SEPARATOR . "recursos" .
242         DIRECTORY_SEPARATOR . "xml" . DIRECTORY_SEPARATOR . "visualizacion.xml"; // 1
243
244         if (!file_exists($direccion)) { // 2
245             $dom = new DOMDocument('1.0', 'UTF-8'); // 3
246             $xml = $dom->createElement("documentos", ''); // 3
247             $dom->appendChild($xml); $files = fopen($direccion, "a+"); // 3
248             $f = fputs($files, $dom->saveXML());  $f = fclose($files); // 3
249         }
250         $xml = simplexml_load_file($direccion);  $docP = $xml->children(); $cantf = count($docP); // 4
251         $arrayVisualizacion = array(); $can = 1; $cantt = 0;      // 4
252             for ($cont = (int) $start; $cont < (int) $start + (int) $limit && $cont < $cantf; $cont++) { //5
253                 $item = array(); // 6
254                 $item['nombre'] = $docP[$cont]->getName(); // 6
255                 $item['subsistema'] = (string) $docP[$cont]->attributes()->subsistema; // 6
256                 $item['servicio'] = (string) $docP[$cont]->attributes()->servicio; // 6
257                 $item['cabecera'] = $docP[$cont]->cabecera->campos; // 6
258                 $can++; $item['coleccion'] = $docP[$cont]->coleccion->campos; // 6
259                 $arrayVisualizacion[] = $item;        // 6
260             }
261         $result = array('cantidad_filas' => $cantt, 'datos' => $arrayVisualizacion);// 7
262         echo json_encode($result); return; // 7
263     }
```



La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. A continuación se realiza el cálculo de la complejidad ciclomática de la funciona correspondiente a la figura XX mediante tres fórmulas descritas, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea correcto.

1. V (G) = (A – N) + 2

Donde “A” es la cantidad total de aristas y “N” la cantidad total de nodos.

V (G)= (8-7)+2

V (G)=3

2. V (G) = P + 1

Donde “P” es la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G)=2+1

V (G)=4

3. V (G) = R

Donde “R” es la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

V (G)=3

Con la correcta realización del cálculo de complejidad se determinó el número de posibles caminos por donde el flujo debe circular y el número de casos pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. El cálculo arrojó como valor 4 por lo que seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- Camino básico #1: 1-2-3-5-7.
- Camino básico #2: 1-2-4-5-7.
- Camino básico #3: 1-2-4-5-6-7.

Para cada camino se realiza un caso de prueba:

- Caso de prueba para el **Camino básico #1:**

Descripción: El primer camino cumplirá con los siguientes requisitos:

Condición de ejecución: El XML no está creado.

Entrada: -

Resultados esperados: Se crea el XML.

- Caso de prueba para el **Camino básico #2:**

Descripción: El segundo camino cumplirá con los siguientes requisitos:

Condición de ejecución: EL XML esta creado y vacío.

Entrada: -

Resultados esperados: No se carga el XML.

- Caso de prueba para el **Camino básico #3:**

Descripción: El primer camino cumplirá con los siguientes requisitos:

Condición de ejecución: El XML está creado y contiene datos.

Entrada: -

Resultados esperados: Se cargan los datos que contiene el XML.

Al aplicar los casos de prueba descritos anteriormente se comprobó que el flujo de trabajo de la función es correcto pues cumplió con las condiciones necesarias de la prueba.

3.5.1 Diseño de casos de prueba

Requisito: Adicionar documento primario

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1.1 Adicionar documento primario.	Adicionar un nuevo documento primario.	EP 1.1: Adicionar documento primario correctamente.	• Se selecciona un elemento (servicio) dentro de las carpetas que representan los subsistemas en el árbol de servicios por subsistemas. • Se activan los campos para insertar datos • Se introducen correctamente los datos. • Se presiona el botón **Aceptar**. • Se presiona el botón **Aceptar** de la ventana de información que muestra el mensaje:" El documento primario fue adicionado satisfactoriamente.". • Se cierra la ventana.

		EP 1.2: Adicionar documento primario incorrectamente.	• Se introducen datos incorrectos (no se selecciona un servicio dentro del árbol). • Se presiona el botón **Aceptar**. • Se muestra el mensaje: "Por favor seleccione un subsistema dentro del árbol.".
		EP 1.3: Adicionar documento primario dejando campos vacíos.	• Se introducen los datos dejando campos vacíos. • Se presiona el botón **Aceptar**. • Se muestra el mensaje de "Por favor verifique, existen campos con valores incorrectos.". • Se marca el campo obligatorio.
		EP 1.4: Adicionar documento primario registrando un nombre ya registrado.	• Se introducen los datos registrando un nombre ya existente. • Se presiona el botón **Aceptar**. • Se presiona el botón **Aceptar** de la ventana de información que muestra el mensaje: "Ya existe un documento primario con el mismo nombre en el XML.".
		EP 1.5: Cancelar.	• Se introducen los datos o no. • Se presiona el botón

			Cancelar. • Se cierra la ventana.
		EP 1.6: Aplicar.	• Se introducen los datos. • Se presiona el botón **Aplicar**. • Se presiona el botón Aceptar de la ventana de información que muestra el mensaje:" El documento primario fue adicionado satisfactoriamente.". • .Se muestra el formulario con los campos vacíos.

Tabla 16: Descripción de caso de prueba del requisito adicionar documento primario.

Requisito: Modificar documento primario

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1.1 Modificar documento primario.	Modificar un documento primario.	EP 1.1: Modificar un documento primario correctamente.	• Se modifican los datos correctamente. • Se presiona el botón **Aceptar**. • Se presiona el botón **Aceptar** de la ventana de información que muestra el mensaje:" El documento primario fue modificado satisfactoriamente.".
		EP 1.2: Modificar un documento primario incorrectamente.	• Se introducen datos incorrectos (se deselecciona un servicio dentro del árbol). • Se presiona el botón **Aceptar**.

			• Se muestra el mensaje: "Por favor seleccione un subsistema dentro del árbol."
		EP 1.3: Modificar documento primario sin hacer cambios.	• Se presiona el botón **Aceptar** sin haber efectuado ningún cambio. • Se muestra el mensaje informativo: "Usted no ha efectuado ningún cambio."
		EP 1.4: Cancelar.	• Se introducen los datos o no. • Se presiona el botón **Cancelar**.

Tabla 17: Descripción de caso de prueba del requisito modificar documento primario.

Requisito: Eliminar documento primario

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1.1 Eliminar documento primario	Eliminar un documento primario de la lista con los existentes.	EP 1.1: Eliminar documento primario.	• El sistema muestra el mensaje: "Está seguro que desea eliminarlo?". • Presionar el botón **Aceptar**. • Se elimina el documento primario.
		EP 1.2: Cancelar.	• El sistema muestra el mensaje: "Está seguro que desea eliminarlo?". • Presionar el botón **Cancelar**.

Tabla 18: Descripción de caso de prueba del requisito eliminar documento primario.

Requisito: Buscar documento primario

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1.1 Buscar documento primario.	Buscar un documento primario en la lista.	EP 1.1: Buscar documento primario	• Se introduce el criterio de búsqueda. • Se presiona el botón **Buscar**. • El sistema muestra el resultado de la búsqueda.

Tabla 19: Descripción de caso de prueba del requisito buscar documento primario.

Requisito: Listar documento primario

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1.1 Listar documento primario	Listar los documentos primarios.	EP 1.1: Listar documentos primarios	• El sistema mostrará las los documentos primarios que hayan sido adicionados o existan ya en el XML.

Tabla 20: Descripción de caso de prueba del requisito listar documento primario.

Requisito: Listar documento primario

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1.1 Mostrar documento primario.	Mostrar un documento primario seleccionado.	EP 1.1: Mostrar un documento primario insertando un identificador correcto.	• El sistema muestra una ventana solicitando el identificador de documento a mostrar. • Insertar número de identificación del documento primario. • Presionar el botón **Aceptar**. • El sistema muestra el documento. • Presionar el botón **Cerrar**

			para salir.
		EP 1.2: Mostrar un documento primario insertando un identificador incorrecto.	• El sistema muestra una ventana solicitando el identificador de documento a mostrar. • Insertar número de identificación del documento primario que no contiene datos o no existe. • Presionar el botón **Aceptar**. • El sistema muestra un mensaje de error informando que no hay datos a mostrar. • Presionar el botón **Aceptar**.

3.6 Conclusiones

En este capítulo fueron expuestos los artefactos generados como parte del modelo de implementación de la solución propuesta, como el diagrama de componentes. Además se describieron los estándares de codificación utilizados. Se validó el diseño propuesto en el capítulo anterior mediante la aplicación de métricas para la evaluación de atributos de calidad, lo cual permitió valorarlo de muy bueno dado los resultados obtenidos. Además se describieron las pruebas estructurales y funcionales realizadas que permitieron comprobar el correcto funcionamiento del sistema.

CONCLUSIONES GENERALES

Con el desarrollo del presente trabajo de diploma se cumplió el objetivo principal de la investigación; desarrollar un componente para la configuración y visualización visual de documentos primarios. Para obtener dicho resultado, se cumplieron los siguientes aspectos:

- Se construyó el marco teórico de la investigación, en el que realizó un estudio de las herramientas similares con el objetivo de reutilizarlas o estudiarlas para adquirir experiencias y comprobar si cumplían las expectativas del centro.
- Se realizó el análisis y diseño de la solución, puesto que al construir el marco teórico se llegó a la conclusión que no existían herramientas que cumplieran las expectativas del centro y se necesitaba el desarrollo de una que abarcara las funcionalidades y objetivos del componente para la configuración y visualización de documentos primarios en el marco de trabajo Sauxe. Este aspecto incluyó la generación de los artefactos que propone el modelo de desarrollo empleado.
- Se realizó la implementación y prueba de la solución, donde se cumplimentó el flujo para el desarrollo del componente. Se validó el diseño propuesto y la implementación, ambos con resultados satisfactorios según las pruebas escogidas y realizadas.

Tras culminados los tres capítulos con los que cuenta la presente investigación se puede afirmar que:

- Las métricas orientadas a las clases empleadas, validaron que se realizó un diseño correcto de la solución.
- Las pruebas estructurales y funcionales realizadas a la aplicación, validaron el correcto funcionamiento de la misma.
- La herramienta obtenida permitirá al centro CEIGE configurar y visualizar de una manera estándar los documentos primarios, así como aumentar su disponibilidad y aminorar el tiempo empleado para la consulta de los mismos pues ahora se encuentran de una forma más ordenada y con los datos realmente importantes para quien los consulta.

Esta herramienta integrada al marco de trabajo Sauxe permitirá que sea usada en otras entidades donde sea desplegado.

RECOMENDACIONES

Con el paso del tiempo, el desarrollo de nuevas tecnologías y la evolución de los negocios, las aplicaciones tienden a ser menos útiles y productivas, es por ello que se recomienda:

- Utilizar este trabajo como bibliografía para posibles investigaciones referentes al tema de los documentos primarios, su configuración y estandarización.
- Se recomienda que el componente sea implantado para su uso cotidiano e introducido en el desarrollo de las aplicaciones web que se desarrollen sobre el Marco de Trabajo Sauxe.
- Continuar perfeccionando la herramienta analizada a partir de nuevos requisitos que puedan surgir como resultado de su explotación.

BIBLIOGRAFÍA

1. Darie, Cristian, y otros. *Building Responsive Web Applications.* s.l. : Packt Publishing, 2006. 1-904811-82-5.

2. Gastaminza, Prof. Félix del Valle. Universidad Complutense Madrid. [En línea] 2007. [Citado el: 24 de Febrero de 2011.] http://www.ucm.es/info/multidoc/prof/fvalle/tema3.htm.

3. Prof. Franco Castellanos, Carlos. *Documentos Primarios.* La Habana : s.n., 2010.

4. Foldoc. Free Online Dictionary of Computing, Imperial College, Department. [En línea] 1995. [Citado el: 3 de 3 de 2011.] http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=framework&action=Search.

5. Foote., Ralph E. Johnson and B. *Designing reusable classes. J.* 1988.

6. The SOA agenda. [En línea] 25 de 09 de 2010. [Citado el: 2 de 3 de 2011.] http://soaagenda.com/journal/articulos/que-son-los-frameworks/.

7. Hardcover, Stephen H. Kaisler. *Software Paradigms.* 2005. 0471483478.

8. Stimulsoft. *Stimulsoft.* [En línea] 2010. [Citado el: 2 de Marzo de 2011.] http://www.stimulsoft.com/ReportsUltimate.aspx.

9. Spring Source. [En línea] 2011. [Citado el: 23 de Febrero de 2011.] http://www.springsource.com/exchange/partners/pentaho.

10. UCID. *Proceso de Desarrollo y Gestión de Proyectos de Software.* La Habana : s.n., 2010.

11. Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'. *Learning Ext JS.* 2008. 978-1-847195-14-2.

12. Flannagan, David. *JavaScript: The Definitive Guide.* 1997. 1-56592-235-2.

13. W3C. *World Wide Web Consortium.* [En línea] 6 de Septiembre de 2010. [Citado el: 25 de Febrero de 2011.] http://www.w3.org/XML/.

14. PHP: Hypertext Preprocessor. [En línea] 2011. [Citado el: 7 de Marzo de 2011.] http://www.php.net/.

15. Zend Framework. [En línea] 28 de Noviembre de 2010. [Citado el: 7 de Marzo de 2011.] http://framework.zend.com/manual/en/introduction.overview.html.

16. Doctrine. [En línea] [Citado el: 7 de Marzo de 2011.] http://www.doctrine-project.org/.

17. Gómez Baryolo, Oiner, Morejón Borbón, Yoandry y García Tejo, Darien. *Marco de Trabajo para Aplicaciones Web de Gestión.* La Habana : s.n., 2010.

18. Visual Paradigm. [En línea] [Citado el: 7 de Marzo de 2011.] http://www.visual-paradigm.com/product/vpuml/provides/.

19. Tigris.org. [En línea] 2009. [Citado el: 7 de marzo de 2011.] http://rapidsvn.tigris.org/.

20. PostgreSQL. [En línea] [Citado el: 7 de Marzo de 2011.] http://www.postgresql.org/about/.

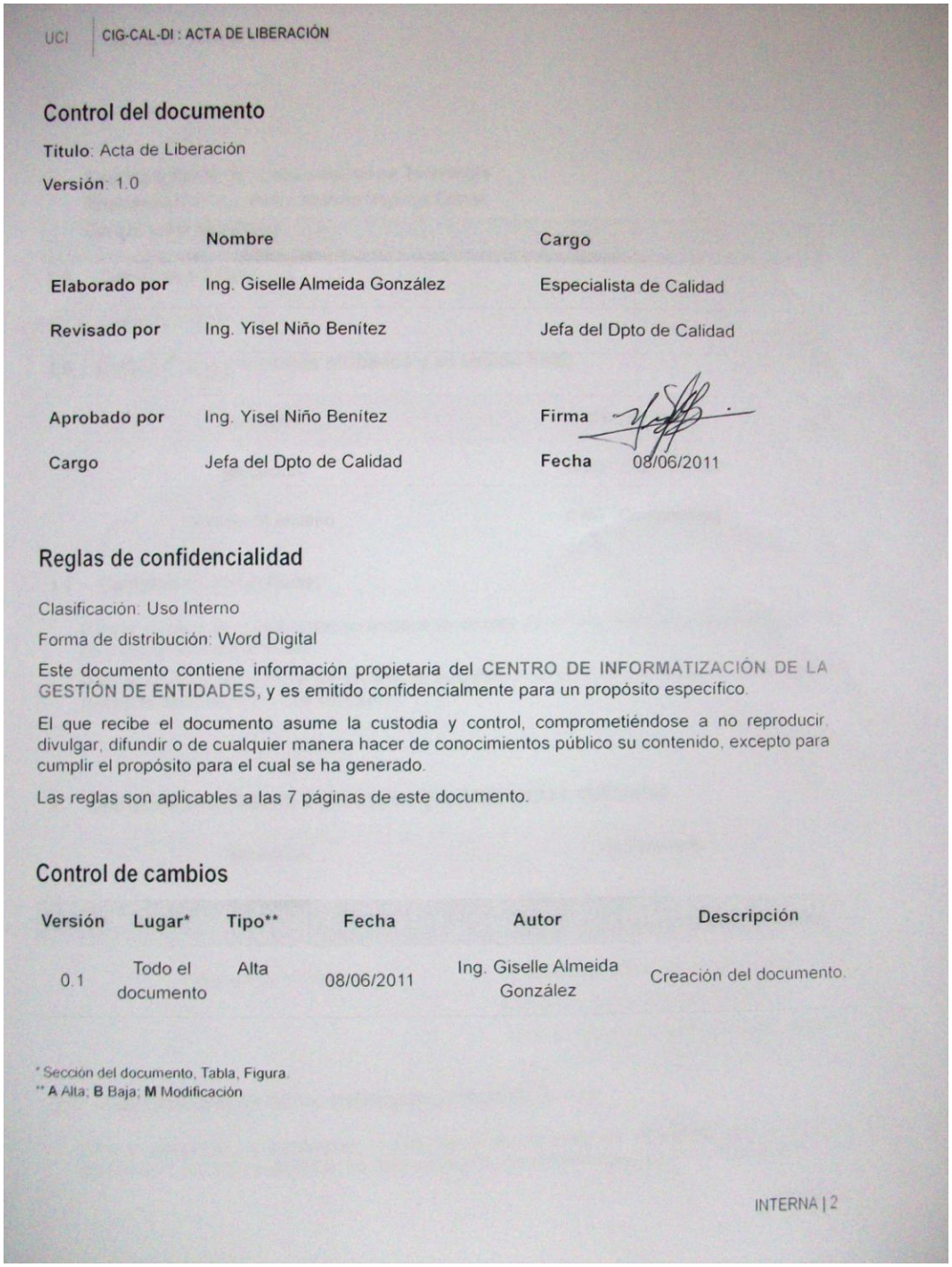
21. Zend Studio. [En línea] 2010. [Citado el: 4 de Marzo de 2011.] http://www.zend.com/en/products/studio/.

22. Kabir, Mohammed J. La biblia Servidor Apache2. s.l. : Anaya.

23. symfony.es. Netbeans. [En línea] [Citado el: 16 de 12 de 2010.] http://www.symfony.es/2009/10/05/netbeans-ya-incluye-soporte-para-symfony/.

24. Lauesen, Soren. Software Requirements. 2002. 0 201 74570 4.

ANEXOS



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Control del documento

Título: Acta de Liberación

Versión: 1.0

	Nombre	Cargo
Elaborado por	Ing. Giselle Almeida González	Especialista de Calidad
Revisado por	Ing. Yisel Niño Benítez	Jefa del Dpto de Calidad
Aprobado por	Ing. Yisel Niño Benítez	**Firma**
Cargo	Jefa del Dpto de Calidad	**Fecha** 08/06/2011

Reglas de confidencialidad

Clasificación: Uso Interno

Forma de distribución: Word Digital

Este documento contiene información propietaria del CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES, y es emitido confidencialmente para un propósito específico.

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimientos público su contenido, excepto para cumplir el propósito para el cual se ha generado.

Las reglas son aplicables a las 7 páginas de este documento.

Control de cambios

Versión	Lugar*	Tipo**	Fecha	Autor	Descripción
0.1	Todo el documento	Alta	08/06/2011	Ing. Giselle Almeida González	Creación del documento.

* Sección del documento, Tabla, Figura.
** **A** Alta; **B** Baja; **M** Modificación

INTERNA | 2

Figura 23: Acta de liberación del producto.

UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Datos del producto

Emitida a favor de: Departamento de Tecnología
Responsable: Ing. Pedro Manuel Nogales Cobas
Cargo: Líder de la línea.

1.5 Clasificado como:

- Aplicación Web.

1.6 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Aplicación	0 No Conformidad
Manual de usuario	0 No Conformidad

1.7 Cantidad de iteraciones:

Para la revisión de la aplicación se emplearon un total de 4 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

Para la revisión del manual de usuario se emplearon un total de 2 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

2 Elementos revisados o probados y herramientas utilizadas

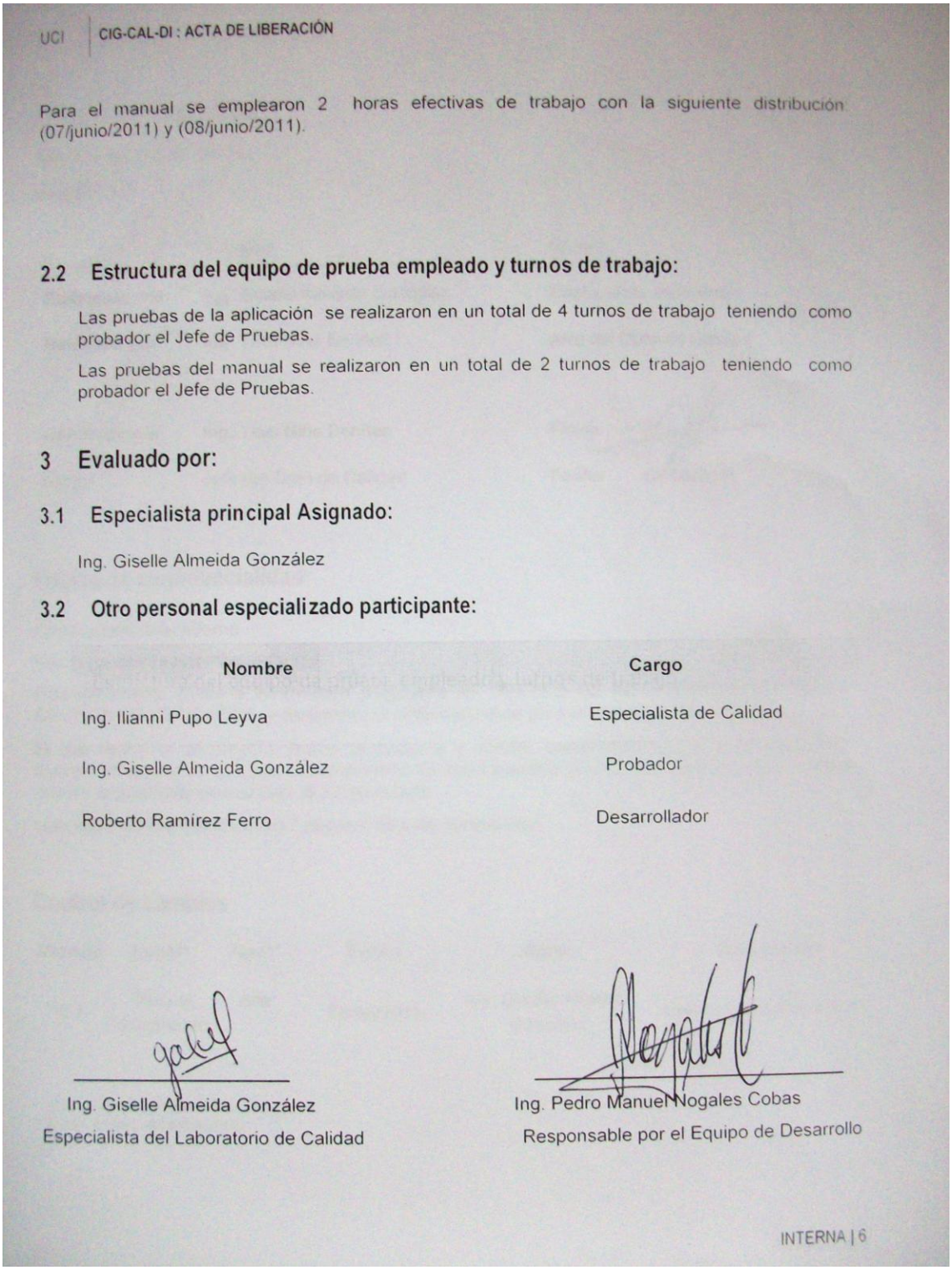
Elemento	Herramienta
Aplicación	Estándar de interfaz Lista de Chequeo de Diseño de Interfaz
Manual de usuario	Estándar para el Manual de usuario Estándar para la documentación Lista de Chequeo del Manual de usuario

2.1 Cantidad total de horas empleadas y rango de fechas:

Para la aplicación se emplearon un total de 11 horas efectivas de trabajo con la siguiente distribución: 3h (31/mayo/2011), 3h (02/junio/2011), 2h (07/junio/2011) y 3h (08/junio/2011).

INTERNA | 5

Figura 24: Acta de liberación del producto.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Para el manual se emplearon 2 horas efectivas de trabajo con la siguiente distribución (07/junio/2011) y (08/junio/2011).

2.2 Estructura del equipo de prueba empleado y turnos de trabajo:

Las pruebas de la aplicación se realizaron en un total de 4 turnos de trabajo teniendo como probador el Jefe de Pruebas.

Las pruebas del manual se realizaron en un total de 2 turnos de trabajo teniendo como probador el Jefe de Pruebas.

3 Evaluado por:

3.1 Especialista principal Asignado:

Ing. Giselle Almeida González

3.2 Otro personal especializado participante:

Nombre	Cargo
Ing. Ilianni Pupo Leyva	Especialista de Calidad
Ing. Giselle Almeida González	Probador
Roberto Ramírez Ferro	Desarrollador

Ing. Giselle Almeida González
Especialista del Laboratorio de Calidad

Ing. Pedro Manuel Nogales Cobas
Responsable por el Equipo de Desarrollo

INTERNA | 6

Figura 25: Acta de liberación del producto

GLOSARIO DE TÉRMINOS

API: La interfaz de programación de aplicaciones es un conjunto de funciones residentes en bibliotecas.
Artefactos: Productos tangibles del proyecto que son creados, modificados y usados por las actividades.
Componente: Unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.
Funcionalidad: Coherencia entre las necesidades detectadas y los resultados que se obtienen con el uso del material. Es lo que un producto puede hacer. Probar la funcionalidad significa asegurar que el producto funciona tal como estaba especificado.
Herramienta: es un objeto elaborado a fin de facilitar la realización de una tarea mecánica.
Logs: Es un registro oficial de eventos durante un rango de tiempo en particular.
Meta-lenguaje: Lenguaje que se usa para hablar acerca de otro lenguaje.
Método: Conjunto de instrucciones a las que se les da un determinado nombre de tal manera que sea posible ejecutarlas en cualquier momento sin tenerlas que reescribir sino usando sólo su nombre.
Métrica: Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.
Proceso: Un proceso se define como un conjunto de tareas, actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido.
Reutilización: Acción de volver a utilizar los bienes o productos.
Sistema: Es un conjunto organizado de objetos o partes que interactúan entre sí y son interdependientes. Se relacionan formando un todo unitario y complejo.
Software: Se refiere al equipamiento lógico o soporte lógico de una computadora, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.
Soporte: Acciones que permiten mantener disponibles los recursos de hardware o software que necesita un producto de software.
Subsistemas: En la misma definición de sistema, se hace referencia a los subsistemas que lo componen, cuando se indica que el mismo está formado por partes o objetos que forman el todo.
Usuario: El usuario es la persona que consume o usa el producto, bien o servicio.
Validación: Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requisitos para una utilización o aplicación especifica prevista.