UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

TÍTULO: Repositorio de Componentes para el Departamento de Señales Digitales

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR: Jorge Luis Hernández Legrá

TUTOR: Ing. Solangel Rodríguez Vázquez

La Habana, 28 de Junio del 2011

"Año 53 de la Revolución"

DEDICATORIA

Dedico el presente trabajo de diploma, a mi mamá por todo el amor, apoyo y comprensión que me ha brindado durante toda mi vida, por luchar día a día sin cansancio para que yo pudiera estar aquí, por darme todo lo que tengo hoy.

A mi papá por darme la educación que tengo hoy, porque gracias a él he llegado hasta aquí, por confiar en mí, por estar ahí cuando lo he necesitado a pesar de poseer un carácter fuerte.

A mi hermana que es lo más grande que tengo en la vida.

A mi abuela Rosa, por todo el apoyo que me brindó en todo el transcurso de mi vida estudiantil.

En fin a toda mi familia que de una forma u otra aportaron su granito de arena para que yo pudiera llegar hasta donde estoy y cumplir mi sueño.

AGRADECIMIENTOS

A Dios.

A mi mamá por sus consejos y por tantas noches de desvelo que tuvo pensando en mí.

A mi papá por confiar en mí, por ser paciente y apoyarme en todas mis decisiones.

A mi hermana, porque a pesar de su corta edad hacia lo posible por aconsejarme y hacerme reír en medio de las adversidades.

A mis tía (o) s Yanet, Sara, Dania, Hernán, Fernando y Abelito por estar ahí cada vez que necesitaba escuchar un consejo más.

A mi prima Lisbet que nunca dejó de escribirme para darme ánimo y que en los últimos días la tenía loca revisándome el documento.

A Dora y Alcides porque a pesar de tantos dolores de cabeza que le di en estos 5 años nunca dejaron de apoyarme.

A mis tutores, por todo el trabajo que hicieron a lo largo de este curso.

A mis amigos (a) s que llegaron a convertirse en hermanos (a) s en el transcurso de la carrera Maylin, Magalis, Yelen, Fernando, Rocny, Roberto, Iván, Rayner, gracias por haberme aguantado estos 5 años, por haber estado ahí presente cuando los necesité. En fin a todos esas personas que de una forma u otra contribuyeron en mi formación como Ingeniero, a todos Muchas Gracias.

RESUMEN

El desarrollo de software basado en componentes se ha convertido actualmente en uno de los paradigmas más efectivos para la construcción de aplicaciones de software. En correspondencia a lo anteriormente planteado la presente investigación realizada en el Departamento de Señales Digitales de la Universidad de las Ciencias Informáticas (UCI), tuvo como objetivo desarrollar un repositorio de componentes reutilizables, para uso de los desarrolladores de software, con el afán de suplir la demanda de sistemas informáticos. En el repositorio propuesto se incluyen el código fuente, los archivos ejecutables, así como una clara documentación tanto de su uso como de la descripción de su interfaz. Provee la recuperación de la información y resuelve el problema de la búsqueda rápida. En cumplimiento del objetivo propuesto, los principales resultados se encuentran asociados a la implementación de un nuevo modelo de producción basado en componentes como una herramienta necesaria para almacenar y gestionar los mismos desarrollados en el departamento, así como el control de todas sus versiones.

PALABRAS CLAVES: Componentes, repositorio, software, repositorio de componentes.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE FIGURAS

INTRODUCCIÓN

A lo largo de su desarrollo, el hombre aprendió a reutilizar el conocimiento en su beneficio. Se percató que al usar algunas ideas, experiencias o herramientas adquiridas en el transcurso de su evolución, evitaría cometer los mismos errores que en el pasado, logrando así realizar diversas tareas con mayor facilidad. Aprovechándose de esto fue perfeccionando sus instrumentos y el lugar donde los construía. A mediados del siglo XVIII surgieron las fábricas con la revolución industrial, dotadas de maquinarias y herramientas necesarias para la fabricación u obtención de ciertos productos o la transformación industrial de una fuente de energía.

Las fábricas continuaron desarrollando sus formas de producción, en 1913 Henry Ford, dueño de una fábrica de automóviles pone en práctica por primera vez una nueva técnica para producir sus autos llamada producción en serie. Consistía en que un trabajador al permanecer en un área de trabajo específica le llegarían a sus manos los componentes (siendo estos los que componen o entran en la composición de un todo) o piezas para así realizar su labor. Este nuevo método transformó para siempre el modelo de producción que existió hasta entonces. **(1)**

En 1969 se crea la primera fábrica de software (organización estructurada creada para el desarrollo de software, con procesos estandarizados, y mejorables continuamente llamada Hitachi Software Works materializando así la concepción dada por Robert William Bemer un año antes. **(2)** Los sistemas informáticos resultaban más complejos y variables a causa del desarrollo constante de las Tecnologías de la Información y la Comunicación (TIC's). Esto llevó a que las fábricas de software, se dieran a la tarea de buscar un modelo de desarrollo que les garantizara construir un sistema en tiempo récord. Se pretendía además cumplir con los estándares de calidad y reducir, al máximo, los esfuerzos humanos y económicos. Todo esto propició el progreso de lo que hoy se conoce como la Ingeniería de Software basada en Componentes (ISBC).

ISBC es un proceso que se centra en el diseño y construcción de sistemas a partir de computadoras que emplean componentes de software reutilizables. **(3)** De ahí que la elaboración de una aplicación o sistema se convierte en la búsqueda y ensamblaje de los componentes más convenientes; esto trae consigo la necesidad de tener almacenado los componentes previamente elaborados.

Debido a la variedad del mercado informático hoy en día existen disimiles empresas que comercializan software, algunas de ellas poseen micro fábricas que utilizan éste modelo basado en componentes para realizar sus productos, ejemplo de ellas tenemos a Microsoft, Apple. Nuestro país, a pesar de no poseer una gran economía ni infraestructura tecnológica, no se encuentra aislado de este proceso, poco a poco se abre camino en este tipo de industria.

La Universidad de las Ciencias Informáticas (UCI) juega un papel fundamental en el proceso de informatización de la sociedad, de ahí que protagonice la mayoría de los cambios que en el país acontecen. De igual forma el Departamento de Señales Digitales de la Facultad 6 se encuentra estrechamente ligado a este proceso.

Por el aumento de compromisos de exportación de software del departamento y la escases de recursos humanos, después de hacer un análisis exhaustivo de modelos de producción, se define utilizar un modelo de fábrica de software personalizado el cual tiene como característica que basa la línea de producción en componentes públicos y reutilizables. Sin embargo los componentes con sus respectivas documentaciones se realizan por separado en cada proyecto del departamento, lo que trae consigo que al realizar el ensamblaje de un producto se tenga que recurrir a cada proyecto en busca de los componentes que tengan realizados para ajustar toda la documentación además de la arquitectura de los mismos, retrasando así todo el proceso. Todo lo planteado anteriormente conlleva al siguiente **problema a resolver** ¿Cómo almacenar y gestionar los componentes realizados en el Departamento de Señales Digitales?, como **objeto de estudio** tenemos el proceso de identificación y clasificación de componentes, planteándose como **objetivo general** desarrollar un repositorio de componentes para el Departamento de Señales Digitales. Delimitando como **campo de acción** la informatización del proceso de identificación y clasificación de componentes. La **idea a defender** parte de la siguiente premisa: con el desarrollo de un repositorio de componentes en el departamento de Señales Digitales se optimizará el uso de los componentes desarrollados.

Para darle cumplimiento a dicha investigación se generan las siguientes **tareas de la investigación**:

1. Caracterizar los procesos de almacenamiento y gestión de archivos.
2. Caracterizar los procesos necesarios para el almacenamiento y gestión de componentes.
3. Describir el estado del arte de repositorios de componentes.
4. Determinar las tecnologías y herramientas a utilizar para el desarrollo de la aplicación.

5. Analizar y diseñar el sistema Repositorio de Componentes.
6. Implementar sistema de Repositorio de Componentes.
7. Validación de la propuesta realizando pruebas de desarrollador.

Para lograr una mejor realización de la investigación se hizo uso de los métodos teóricos y empíricos siguientes:

1. Métodos Teóricos
   - Histórico – Lógico: Se utilizará para estudiar todo lo relacionado con los antecedentes de los repositorios de componentes.
   - Analítico- Sintético: Se utilizará en el análisis de toda la bibliografía existente extrayendo las ideas fundamentales de estas.
2. Métodos Empíricos.
   - Observación: Se utilizará para caracterizar detalladamente como se realiza en la actualidad el proceso de almacenamiento y gestión de los componentes realizados en el departamento de señales digitales, identificando las principales características para desarrollar un sistema de acuerdo a las necesidades existentes.

El presente trabajo queda estructurado de la manera siguiente:

CAPÍTULO 1: Fundamentación teórica. En este capítulo se plantean todos los elementos teóricos que sustentan la investigación, así como las principales definiciones, siendo estas de gran ayuda para lograr entender con mayor facilidad el problema planteado.

CAPÍTULO 2: Tendencias y tecnologías actuales a desarrollar. En este capítulo se planeta todo lo relacionado con las tecnologías actuales que se utilizarán para darle solución a la situación trazada.

Capítulo 3: Presentación de la solución propuesta. El capítulo propone una solución al problema planteado, desarrollándose el modelo conceptual para lograr una mayor comprensión de los procesos, se definen los requisitos funcionales y no funcionales, así como los actores, casos de usos del sistema y sus descripciones.

Capítulo 4: Construcción de la solución propuesta. Se implementa la solución planteada, tratando los aspectos fundamentales de la arquitectura.

CAPÍTULO 1: Fundamentación Teórica del repositorio de componentes

1.1. Introducción

En este capítulo se abordarán temáticas y conceptos relacionados con el domino del problema para lograr así un mayor entendimiento. Se especificará y argumentará el estado del arte con relación al objeto de estudio y campo de acción.

1.2. Gestión y almacenamiento de archivos

Hoy en día, paralelo al aumento del desarrollo en las TIC's aumenta el volumen de información disponible con la que se trabaja, por lo que se hace necesario que ésta se encuentre disponible en todo momento y no dependa del dispositivo físico que proporciona su almacenamiento. Por ésta razón el almacenamiento y la gestión de la información se hacen indispensables, siendo este el escenario principal para la realización de un sistema de almacenamiento y gestión de archivos y componentes orientados a la Factoría de Software.

Para un mejor entendimiento del proceso de almacenamiento y gestión de archivos, es necesario primeramente conocer su definición.

Existen diversas definiciones sobre lo que es un archivo, a continuación se citarán dos de ellas. Según la Real Academia de la Lengua Española un *archivo* no es más que un espacio que se reserva en el dispositivo de memoria de un computador para almacenar porciones de información que tienen la misma estructura y que pueden manejarse mediante una instrucción única. Por ejemplo, la documentación generada en el transcurso de la realización de un sistema, los diagramas generados en las distintas fases por la que pasa el sistema o las clases resultantes de la implementación de este.

Para la realización de ésta investigación después de un estudio minucioso de las definiciones se asumirá como definición de *archivo* a:

Un conjunto de datos que se almacenan bajo un determinado formato; puede ser guardado en el disco duro de la computadora o en algún otro medio de almacenamiento como disquete, disco compacto, memorias extraíbles, entre otras. **(4)**

Almacenamiento de archivos

Un ordenador puede almacenar información en varios tipos de soporte, como los discos magnéticos y los ópticos, así como en otros dispositivos como el almacenamiento (de manera temporal, RAM[1]) y las tarjetas de almacenamiento portátil con interfaz USB[2]). El sistema operativo se encarga de brindar una vista lógica uniforme para el almacenamiento de la información, proporcionando una abstracción de las propiedades físicas de los elementos de almacenamiento, con el fin de definir una unidad lógica de almacenamiento: el archivo. **(5)**

¿Qué es un sistema de gestión de archivos?

Un sistema de gestión de archivos es el software del sistema que proporciona servicios a usuarios y aplicaciones para el uso de archivos. Normalmente la única forma en que un usuario o aplicación puede acceder a los archivos es mediante el sistema de gestión de archivos. **(6)**

La Real Academia de la Lengua Española define que un *componente* no es más que algo que compone o entra en la composición de un todo, ahora bien el Instituto de Ingeniería de Software de los Estados Unidos define que un *componente de software* está formado por varios tipos de archivos, este se caracteriza por tener una implementación opaca de una funcionalidad, está sujeto a composición por terceros y conforme con un modelo de componentes. **(7)**

Conocida ya la definición de archivo y de componente de software nos queda saber cómo trabajar con estos y que operaciones podemos realizar, así como saber sus principales características.

1.2.1 Características de los procesos de almacenamiento y gestión de archivos

En la actualidad todos los sistemas de gestión de archivos tienen como principal objetivo:

- Cumplir con las necesidades de gestión de datos y con los requerimientos del usuario
- Garantizar que los datos de los archivos sean válidos
- Optimizar el rendimiento en términos de productividad y tiempo de respuesta

[1] *random-access-memory (memoria de acceso aleatorio).*

[2] *Universal Serial Bus (*bus universal en serie).

- Ofrecer soporte de Entrada/Salida (E/S) para los distintos dispositivos de almacenamiento
- Minimizar o eliminar la posibilidad de pérdida o destrucción de datos. (6)

Dichos sistemas para lograr ser interactivos como requerimientos básicos presentan:

- Cada usuario debe ser capaz de crear, borrar, y cambiar los archivos
- Cada usuario puede tener acceso controlado a los archivos de otros usuarios
- Cada usuario puede controlar qué tipos de acceso estarán permitidos a sus archivos
- Cada usuario debe poder reestructurar sus archivos de manera adecuada al problema
- Cada usuario debe ser capaz de mover datos entre los archivos
- Cada usuario debe ser capaz de guardar una copia de reserva y recuperar sus archivos en el caso de que hayan sufrido algún daño
- Cada usuario debe ser capaz de acceder a sus archivos mediante un nombre simbólico. (6)

Algunos de estos objetivos y requerimientos se tendrán en cuenta para darle solución al problema planteando.

1.2.2 Características los procesos necesarios para el almacenamiento y gestión de componentes

En la actualidad los procesos de almacenamiento, ordenamiento, descripción, identificación, recuperación de componentes son empleadas las tecnologías digitales. Como ventajas el almacenamiento digital presenta su capacidad de almacenar, recuperar, difundir, gestionar y distribuir gran cantidad de información de disimiles formatos en archivos digitales.

Las características principales de un sistema de almacenamiento y gestión de componentes son la búsqueda y recuperación de archivos de software. Sin embargo, la presencia de algunas otras características, puede facilitar y por lo tanto fomentar su uso. Según lo propuesto por [Ezran et al., 1999], se presentará algunas de las funciones claves que puede hacer un repositorio a disposición de sus usuarios. Vale resaltar que estas funciones no son obligatorias y varían según el contexto en el que se ha insertado y la organización tiene que utilizar un repositorio. **(8)**

Identificación y descripción: para describir un archivo es posible que se le diera un conjunto de características tales como nombre, dominio, palabra clave, y otros que identificar y diferenciarlos de los otros archivos que comprenden el mismo repositorio. Para cada uno de los bienes almacenados

debe ser posible determinar, dentro de una forma homogénea es decir, repositorio, los archivos del mismo tipo deben tener el mismo conjunto de características. Eso no significa, sin embargo, que los diferentes archivos deben tener los mismos valores para este conjunto de características.

Insertar componente: un repositorio debe permitir a los usuarios autorizados insertar nuevos archivos, o incluso nuevas versiones de los mismos. La inserción significa añadir al repositorio un archivo así como su descripción.

Exploración del catálogo: a los usuarios del repositorio se le debe permitir explorar el catálogo de los archivos para que puedan comprender y analizar las características de los archivos disponibles.

Búsqueda en los textos: un repositorio debe permitir que sus usuarios hagan búsquedas más específicas de la descripción de los archivos. Como resultados se obtienen uno o más archivos que cumplen las condiciones deseadas. Observando los resultados, se puede decidir por un mayor detalle o generalización de los criterios anteriores.

Recuperación: después de la identificación del archivo que se desea, un repositorio debe permitir a los usuarios recuperar los archivos para que más adelante se puedan utilizar en un proceso de reutilización.

Organización y búsqueda: como se ha visto anteriormente, la funcionalidad exploración de los catálogos no es suficiente a medida que aumenta la cantidad de archivos disponibles. Además, la búsqueda textual a menudo consume mucho tiempo. Debido a estos factores, algunos repositorios pueden adoptar nuevas formas de organizar las características de los archivos a fin de permitir que la búsqueda en el repositorio se base en otros criterios.

Histórico: es importante para la gestión de un repositorio que puede almacenar información de uso, modificación, creación y supresión de cada uno de los archivos disponibles. Esta la información debe reunir una base histórica para facilitar el análisis y la reutilización de ellos.

Medición: un repositorio puede recopilar estadísticas para facilitar su gestión. Algunas de las estadísticas clave que se pueden adoptar son: frecuencia de acceso al repositorio, cantidad de archivos disponibles, la tasa de recuperación, el porcentaje de búsquedas con éxito, frecuencia con la que se revisa un archivo analizado o modificado, entre otras.

Control de acceso: un repositorio podría adoptar una política de seguridad para determinadas funcionalidades sean accesibles sólo a personas autorizadas. Por ejemplo, se puede definir para un una empresa en concreto la política de seguridad donde la búsqueda de texto esté disponible para todos empleados, pero la recuperación de los archivos esté disponible sólo para los empleados del área de desarrollo.

Gestión de Versiones: un repositorio puede contener múltiples versiones del mismo archivo y por lo tanto, se recomienda que exista algún mecanismo de control de estas versiones y establecer relación entre ellas.

Control de cambios: se recomienda que algunas funciones se proporcionen a hacer cambios en la gestión de un repositorio de archivos. Estas características incluyen procedimientos para solicitar los cambios, las discusiones, y el permiso de ellos. Es de extrema importancia para mantener la coexistencia de un repositorio que los proyectos dirigidos a modificar los archivos se comunicados al responsable de su administración para que el mismo, junto con otros funcionarios puedan revisar los cambios solicitados y mantener la coherencia entre los distintos archivos de software.

Notificación de cambios: es posible que en cualquier momento un archivo sea modificado o que el repositorio experimente algunos cambios en sus funcionalidades principales. Por lo tanto, un repositorio puede proporcionar funciones de notificar a sus usuarios de las modificaciones recientemente ocurridas tales como, la inserción o supresión de los archivos, los cambios en los documentos, disponibilidad de nuevas funciones, nuevas políticas de seguridad, entre otros.

Ya presentadas las principales características de un repositorio es necesario examinar algunos de los aspectos no funcionales que lo caracterizan. Estos aspectos también pueden ser adecuadamente utilizados, lo que permite un mayor incentivo para el uso del repositorio.

Cantidad: un repositorio puede ser simple o pueden existir varios de ellos [Ezranetal, 1999.]. Puede estar diseñado para organizaciones pequeñas, donde no hay una gran división departamental, por lo tanto, los tipos de archivos no son completamente distintos y son entendidos por todos sus usuarios. Para organizaciones más grandes o distribuidas geográficamente sugiere el uso de varios repositorios, cada uno relacionado a un área donde opera la empresa.

En este caso es importante que exista una gestión eficaz y centralizada para evitar incoherencias en la duplicación existentes entre algunos repositorios. El uso de varios de ellos es defendido por

quienes consideran información innecesaria a un grupo determinado de usuarios, pero de gran valor para otras, perjudica y desalientan la práctica de la reutilización en las organizaciones.

Acceso a la red: como la mayoría de empresas están migrando sus modelos de computadora para modelos centralizados, un requisito típico es que el repositorio se puede acceder desde cualquier lugar una red [Ezran et al., 1999]. Esto es particularmente importante cuando los usuarios están distribuidos geográficamente. Lo que generalmente se ha observado en la aplicación de repositorios es el uso de modelos de cliente-servidor en 3 (tres) o más capas. En el modelo de tres una capa es responsable del almacenamiento de los archivos, y el otro para el procesamiento de principales funcionalidades del sistema, como la búsqueda y recuperación y el tercero corresponde a cliente de la aplicación, en el repositorio se puede acceder. El número de capas pueden variar entre diferentes organizaciones. **(8)**

1.3. Objeto de Estudio

1.3.1 Descripción General del Componente de Software

A medida que se va desarrollando la producción de software en el mundo, las diferentes industrias de software van actualizando sus modelos de producción debido a que estos les permiten un fácil manejo de los planes de proyectos, alto nivel de seguridad, buena comunicación con los clientes y una alta calidad del producto desarrollado.

Existen varios modelos de producción de software o modelos de desarrollo de software como comúnmente se conocen, entre los que se destacan:

- Modelo en Cascada.
- Modelo basado en Prototipo.
- Modelo basado en Espiral.
- Modelo basado en Componentes.

Muchas de las empresas que desarrollan software toman como patrón a utilizar el modelo basado en componentes debido a que les brinda facilidades de adaptación al mercado y les permite la adaptación de paquetes de software a partir del desarrollo de funcionalidades adicionales y configuración parámetros. Este modelo esta guiado por procesos que le son de gran importancia

para su funcionamiento y ajuste al medio como son la identificación y clasificación de los componentes ya que constituyen la base fundamental de la producción.

Para identificar un componente de software se debe especificar algunas de sus características más importantes esto garantiza la exclusión de algunos artefactos que pueden ser incluidos en la definición de componentes de software, pero que puedan generar problemas a la hora de contextualizarlos y ubicarlos en un desarrollo basado en componentes. **(9)**

- *Identificable:* Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
- *Auto contenido*: Un componente no debe requerir de la utilización de otros para cumplir la función para el cual fue diseñado.
- *Puede ser remplazado por otro componente*: Se puede reemplazar por nuevas versiones u otro componente que lo remplace y mejore.
- *Con acceso solamente a través de su interfaz***:** Debe asegurar que estas no cambiaran a lo largo de su implementación.
- *Sus servicios no varían***:** Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación puede hacerlo.
- *Documentado***:** Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, entre otras.
- *Genérico:* Sus servicios deben servir para varias aplicaciones.
- *Reutilizado dinámicamente***:** Puede ser cargado en tiempo de ejecución en una aplicación.
- *Independiente de la plataforma***:** Debe ser independiente del Hardware, Software, Sistema Operativo, entre otros. (9)

Otra manera de identificar un componente, es mediante los patrones de diseño, de arquitectura o de análisis. El diseño por componentes de software ha planteado sus propios patrones de análisis para la especificación de un componente, los cuales proveen soluciones en el ciclo de desarrollo de un sistema basado en componentes.

Teniendo en cuenta las definiciones anteriores y las características expuestas referentes a un componente se pude plantear lo siguiente:

CAPÍTULO 1: Fundamentación teórica del repositorio de componentes.

Un componente posee cierto grado de granularidad en dependencia de su tamaño y composición ya que puede estar compuesto por funcionalidades, clases, procedimientos, módulos e incluso aplicaciones. Lo que facilitaría en su programación el que sea reutilizable e interoperable por otros componentes o sistemas en los que su función sea necesaria.

Como resultado de un estudio realizado por los especialistas del departamento de Señales Digitales, se propuso migrar a un modelo de producción basado en la realización de componentes. Este modelo permite que el desarrollo de los sistemas se realice en cortos periodos de tiempo, con alta calidad y bajos costos para los productos en cuestión.

A pesar de que se ha empezado a trabajar en los proyectos del departamento basado en componente y esto ha agilizado de alguna manera la construcción y el desarrollo de los productos, los componentes que en estos se están elaborando no le son de utilidad a los demás grupos de desarrollo, por lo que es muy probable que se esté duplicando esfuerzo en la realización de los mismos ya que un mismo componente puede satisfacer las necesidades de uno o varios proyectos a la vez. De los componentes que se elaboran ni siquiera se realiza una documentación lo suficientemente básica como para saber las funciones principales ni el objetivo de dicho componente, esto incurre en que para poder trabajar con él se debería tener en cuenta a la persona que lo desarrolló y hacer consultas periódicas a dicha persona para poder utilizarlo y comprender su funcionamiento.

Para eliminar dichos tabúes en el mundo se han elaborado diferentes tipos de herramientas que permiten agrupar de cierta manera estos componentes e incluirles ejemplos y documentación que sirva como apoyo para el uso del mismo. Dichas herramientas son los repositorios de componentes que si bien no todos resuelven los problemas actuales al menos ayudan a mejorar y satisfacer algunas de las necesidades existentes.

1.4. Análisis de otras soluciones existentes

Entre las soluciones existentes podemos encontrar:

- ComponentSource[3]

ComponentSource.com es un repositorio realizado con fines de lucro, se dedica a la distribución de componentes de software a nivel internacional y un distribuidor de software de herramienta de desarrollo. Realiza sus ventas en diferentes idiomas, lo que hace que su alcance en el mercado sea mayor. La búsqueda de los componentes pueden realizarse por categorías, autores, o por orden alfabético**.** Organiza sus productos por tipo, plataforma o categorías. Presenta varios módulos entre los que se destacan el módulo de productos más vendidos, más descargados y nuevas versiones.

Como principal desventaja presenta que es privativo, los componentes presentan un alto costo de adquisición, limita su gestión solo a componentes desarrollados en Microsoft .NET / COM y Java/J2EE.

- Repositorio de Software de Apple[4]

Es un repositorio de software perteneciente a la empresa Apple, está estructurado por varios módulos entre ellos están, el de categorías y de archivos. Permite que los usuarios al descargar un producto dejen un comentario si así lo desea.

Presenta como principal desventaja que su gestión está dirigida solo a la búsqueda de software para el sistema operativo MAC OS limitando así su uso, es privativo y carece de funcionalidades.

Repositorio de Software de la Free Software Foundation[5]

Este repositorio lo mantiene activamente la Fundación para Software Libre, contiene un buscador por categorías facilitando así la búsqueda de paquetes de software libres sean o no de GNU, brinda además una lista con todos los paquetes o programas de GNU. De cada categoría muestra los

[3] http://www.componentsource.com

[4] http://www.apple-soft.net

[5] http://directory.fsf.org

paquetes o programas que existen así como una breve descripción del mismo, las versiones existentes y la compatibilidad con las distintas plataformas de trabajo. Entre sus módulos encontramos proyecto más popular en el que muestran los proyectos a los que más acceden los usuarios, una breve descripción de estos además de las versiones existentes.

Como desventaja presenta que es una solución online, lo que quiere decir que no es descargable, carece de funcionalidades y no es adaptable a las necesidades del departamento.

A pesar de que las soluciones existentes analizadas poseen algunas de las características y funcionalidades que se necesitan en el departamento, no pueden ser utilizadas en el mismo pues como se menciona anteriormente son privadas, no presentan todas las funcionalidades que necesita el departamento y muy fundamental se encuentran online, no siendo estas, herramientas que se puedan descargar y personalizar en un ordenador. Es por esta razón que surge la necesidad de crear un sistema que gestione todo lo relacionado con la realización de los componentes en el departamento.

1.5. Conclusiones

En el presente capítulo se presentaron una serie de conceptos los que ayudarán a comprender mejor la investigación desarrollada. Se realizó un análisis profundo sobre el objeto de estudio y las soluciones existentes, llegando a la conclusión de que ninguna de las antes expuestas brinda soluciones a las necesidades del departamento. Por lo que se denota la necesidad de realizar una propuesta de sistema de repositorio de componentes para almacenar de acuerdo a las necesidades y propósitos del departamento, los componentes de software construidos en el mismo.

CAPÍTULO 2: Tendencias y tecnologías actuales a utilizar para el Repositorio de Componentes

2.1 Introducción

En el presente capítulo se aborda todo lo relacionado con las tendencias y las tecnologías actuales en la realización de repositorios de componentes, enfatizando en las que se utilizarán para la realización del sistema propuesto. Este grupo de tecnología está conformado por: La metodología para guiar el desarrollo del sistema, el lenguaje de modelado, el gestor de base de datos, el lenguaje de programación y el entorno de desarrollo.

2.2 Metodología de desarrollo de software

Todo desarrollo de software necesita de una metodología, un conjunto de pasos o procedimientos para guiar su realización. Todo esto contribuye a mantener el proceso de realización del producto ordenado desde su inicio hasta su fin. Hoy en día seguir una metodología de desarrollo nos garantiza mantenernos en competencia pues el mercado de software es muy amplio y variable.

2.2.1 Características generales de RUP

Rational Unified Process (RUP) o como se conoce en español Proceso Unificado de Desarrollo es una metodología que se utiliza en la ingeniería de software. El Proceso Unificado es un marco de trabajo genérico que puede aplicarse a una gran diversidad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. **(10)**

El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema de software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas, usando además para la representación gráfica de los esquemas de un sistema de software el Lenguaje Unificado de Modelado (UML).

Los verdaderos aspectos que definen el Proceso Unificado se resumen en tres: iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura. **(10)**

La metodología de RUP está dividida en 4 fases fundamentales:

- ✓ *Inicio:* El objetivo en esta etapa es determinar la visión del proyecto.
- ✓ *Elaboración:* En esta etapa el objetivo es determinar la arquitectura óptima.
- ✓ *Construcción:* En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial, o sea crear el producto.
- ✓ *Transición:* El objetivo es llegar a obtener el release del proyecto.

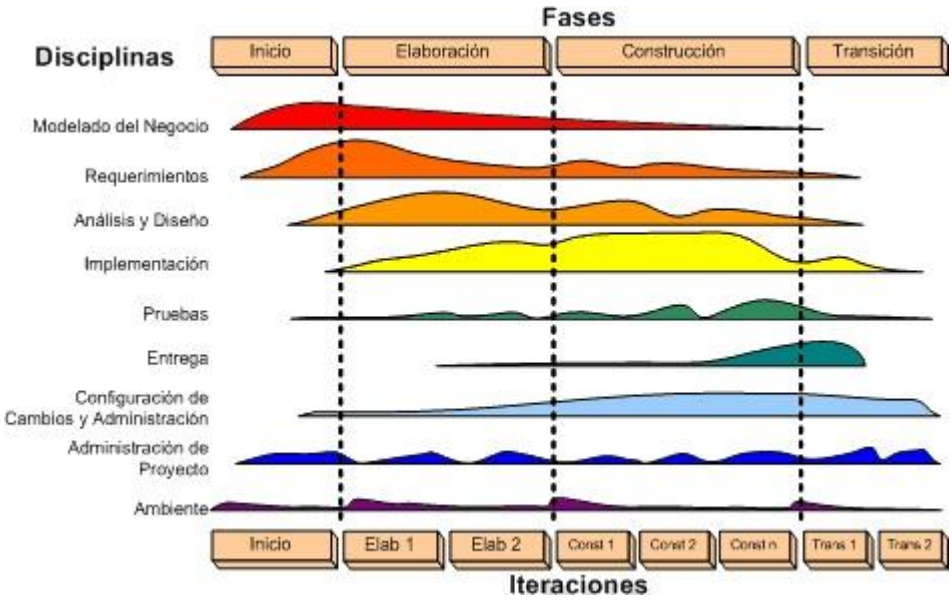


Figura 1 Disciplina, fases, iteraciones del RUP

RUP ofrece una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones. Utiliza una notación de Caso de Uso y escenarios para representar los requisitos. Desarrolla el producto mediante iteraciones con hitos bien definidos, en los cuales se repiten las actividades pero con distinto énfasis, según la fase del proyecto. (11)

Utiliza arquitectura basada en componentes.

✓ La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que posteriormente serán ensamblados para generar el sistema. Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o se desarrollan sus componentes. (11)

Utiliza para el modelado el estándar Lenguaje de Modelado Unificado (UML).

✓ Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema software. Utilizar herramientas de modelado visual facilita la gestión de dichos modelos, permitiendo ocultar o exponer detalles cuando sea necesario. El modelado visual también ayuda a mantener la consistencia entre los artefactos del sistema: requisitos, diseños e implementaciones. (11)

Verifica continuamente la calidad.

✓ Es importante que la calidad de todos los artefactos se evalúe en varios puntos durante el proceso de desarrollo, especialmente al final de cada iteración. En esta verificación las pruebas juegan un papel fundamental y se integran a lo largo de todo el proceso. Para todos los artefactos no ejecutables las revisiones e inspecciones también deben ser continuas. (11)

Gestiona los cambios.

✓ El cambio es un factor de riesgo crítico en los proyectos de software. Los artefactos de software cambian no sólo debido a acciones de mantenimiento posteriores a la entrega del producto, sino que durante el proceso de desarrollo, especialmente importantes por su posible impacto son los cambios en los requisitos. Por otra parte, otro gran desafío que debe abordarse es la construcción de software con la participación de múltiples desarrolladores, posiblemente distribuidos geográficamente, trabajando a la vez en una *release*, y quizás en distintas plataformas. La ausencia de disciplina rápidamente conduciría al caos. La Gestión de Cambios y de Configuración es la disciplina de RUP encargada de este aspecto. (11)**.**

2.2.2 Extreme Programming (XP)

Esta metodología se basa en la idea de que existen cuatro variables que guían el desarrollo de sistemas: Costo, Tiempo, Calidad y Alcance. La manera de encarar los desarrollos avalados por este modelo de desarrollo es permitir a las fuerzas externas (gerencia, clientes) manejar hasta

tres de estas variables, quedando el control de la restante en manos del equipo de desarrollo. Este modelo hace visibles de manera más o menos continua estas cuatro variables. **(12)**.

XP hace hincapié en el trabajo en equipo, los administradores, clientes y desarrolladores son socios iguales en un equipo de colaboración. Implementa un equipo simple, pero efectivo, medio ambiente propicio para llegar a ser altamente productivo. El equipo se auto-organiza en torno al problema a resolver en la forma más eficiente posible. Permite mejorar un proyecto de software en cinco aspectos esenciales: la comunicación, la sencillez, la retroalimentación, el respeto y el coraje. **(13)**

2.2.3 Scrum

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. **(14)**.

2.2.4 ¿Por qué utilizar RUP?

Luego de haber investigado las metodologías existentes se determinó utilizar para la realización del sistema propuesto la metodología RUP debido a que ofrece una alta capacidad organizativa, permitiendo esto realizar grandes proyectos pero a la vez es flexible por lo que permite que se ajuste a cualquier entorno de desarrollo. RUP está basado en la realización de casos de uso, se utiliza cuando se tiene bien clara las delimitaciones de estos, así como las responsabilidades que van a tener en el desarrollo del sistema. Al generar gran cantidad de documentación garantiza que se puedan realizar futuras mejoras al sistema. Sin embargo otras metodologías como BPM están basadas en procesos que están definidos, pero que a su vez no se encuentran delimitadas las actividades internas de dichos procesos. En esta investigación se tienen claras las funcionalidades de la misma, donde comienzan y donde finalizan, están definidas las entradas y las salidas cada una de ellas, por lo que resulta más factible el trabajo con RUP.

2.3 Características de UML a utilizar en la modelación de la solución propuesta

El lenguaje Unificado de Modelado (UML) proporciona un vocabulario y una regla para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema. **(15)**

UML presenta varios objetivos pero se pueden resumir en sus funciones:

- ✓ Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- ✓ Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- ✓ Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión. (16)

Un modelo UML está compuesto por tres clases de bloques de construcción:

- ✓ Elementos: Son abstracciones reales o ficticias (objetos, acciones).
- ✓ Relaciones: Relacionan los elementos entre sí.
- ✓ Diagramas: Son colecciones de elementos con sus relaciones.

UML resuelve de forma satisfactoria el viejo problema del desarrollo de software como es su modelado gráfico. Además ha llegado a una solución unificada basada en lo mejor que había hasta el momento, lo cual lo hace más excepcional. (15)

2.3.1 Herramientas Case

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. (17)

2.3.1.1 Características del Visual Paradigm

Visual Paradigm es una herramienta *CASE* que utiliza UML como lenguaje de modelado. Está diseñada para una amplia gama de usuarios interesados en construir sistemas de *software* fiables con el uso del paradigma orientado a objetos, incluyendo actividades como ingeniería de *software*, análisis de sistemas y análisis de negocios.

Soporta un conjunto de lenguajes, tanto en la generación de código y como de ingeniería inversa sobre Java, C++, CORBA IDL, PHP, XML Schema, Ada y Python. Además, apoya la generación de código C#, VB. NET, código script de Flash, Delphi, Perl, Ruby, entre otros. Ingeniería Inversa también apoya clase Java, .NET, .dll y .exe, JDBC.

2.3.1.2 Rational Rose Enterprise como herramienta CASE.

Rational Rose Enterprise es el producto más completo de la familia Rational Rose. Todos los productos Rational Rose incluyen soporte Unified Modeling Language™ (UML™). *Rational Rose Enterprise* es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java™/J2EE™, Visual C++® y Visual Basic®. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente. (18)

Otras características.

✓ Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
✓ Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
✓ Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
✓ La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
✓ Soporte Enterprise Java Beans™ 2.0.
✓ Capacidad de análisis de calidad de código. (18)

2.3.1.3 ¿Por qué Visual Paradigm?

Después de estudiar y analizar las dos herramientas expuestas anteriormente se determinó utilizar como herramienta para la modelación el Visual Paradigm por las facilidades que esta brinda en el proceso de desarrollo de software. Es una herramienta multiplataforma y gratis en su versión Community por lo que es muy utilizado en el desarrollo de proyectos importantes, mejora el tiempo de construcción de las aplicaciones lo que posibilita realizar el modelado de todo tipo de diagramas de clases, facilita la codificación desde diagramas así como el desarrollo de documentación de una aplicación. Además presenta mucha información en diferentes formatos, tanto audiovisuales como documentación digital.

2.4 Lenguaje de Programación

Para poder controlar el comportamiento físico y lógico de una máquina o para interpretar las instrucciones dadas por los usuarios a estas se necesita un lenguaje propio, este medio de comunicación que permite la relación entre usuarios y máquinas u ordenadores es el lenguaje de programación.

El sistema propuesto estará relacionado con varios lenguajes de programación, estos se encuentran distribuidos en dos grupos, los que se utilizan en el lado del servidor y los del lado del cliente. Los lenguajes utilizados del lado del cliente son utilizados para validar toda la información que se enviará al servidor así como para el tratamiento de la interfaz del sistema. Los del lado del servidor garantizan que toda la información que llega a este sea de forma segura y con calidad.

2.4.1 Lenguaje del lado del Cliente

2.4.1.1 JavaScript

JavaScript es un lenguaje interpretado que se utiliza principalmente en la creación de páginas web dinámicas. Este permite realizar mejoras en la interfaz de usuario, validar datos, crear menús interactivos, animaciones, entre otros. Es interpretado por todos los navegadores existentes en la actualidad, por ser independiente de la plataforma puede ser ejecutado en cualquier ordenador sin importar su sistema operativo.

JavaScript comparte muchos elementos con otros lenguajes de alto nivel. Hay que tener en cuenta que este lenguaje es muy semejante a otros como C, Java o PHP, tanto en su formato como en su sintaxis, aunque por supuesto tiene sus propias características definitorias. Permite diferenciar mayúsculas y minúsculas, por lo que si se escribe alguna expresión en minúsculas, se debe mantener esa expresión en minúsculas a lo largo de todo el programa. También puede encerrar las expresiones que se escriban con una serie de caracteres especiales; estos caracteres se denominan operadores y sirven tanto para encerrar expresiones como para realizar trabajos con ellas. **(19)**

Este lenguaje fue diseñado de forma que se ejecutara en un entorno muy limitado que permitiera a los usuarios confiar en la ejecución de los scripts. De esta forma, los scripts de JavaScript no pueden comunicarse con recursos que no pertenezcan al mismo dominio desde el que se descargó el script. Los scripts tampoco pueden cerrar ventanas que no hayan abierto esos mismos scripts. **(20)**

2.4.1.2 EXT

Ext es una librería JavaScript que permite la creación de aplicaciones web. Surge a principios de 2007 doble licenciada bajo la GPL y la licencia comercial. Contiene componentes UI (elementos configurables y reutilizables) del alto performance y personalizables, modelo de componentes extensibles y un API fácil de usar.

Ext puede ejecutarse en contra de cualquier plataforma de servidor que puede procesar peticiones POST y devolver datos estructurados, opciones más comunes son PHP, Java. .NET y muchos más. **(21)**

Una de las grandes ventajas de utilizar ExtJS es que nos permite crear aplicaciones complejas utilizando componentes predefinidos, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, Chrome, etc.).Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otras librerías. Usar un motor de render como ExtJS nos permite tener además estos beneficios:

- ✓ Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- ✓ Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- ✓ Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

2.4.1.3 Ajax

Ajax acrónimo de Asynchronous JavaScript And XML es una nueva práctica para desarrollar aplicaciones web interactivas. Estas mientras son interpretadas del lado del cliente es decir por el navegador de los usuarios, mantienen una comunicación asíncrona con el servidor lo que permite realizar cambios en las páginas sin necesidad de actualizarlas o recargarlas, aumentando así la interactividad y la velocidad de las aplicaciones web.

Ajax está conformado por las siguientes tecnologías.

- ✓ XHTML y CSS, para crear una presentación basada en estándares
- ✓ DOM, para la interacción y manipulación dinámica de la presentación
- ✓ XML, XSLT y JSON, para el intercambio y la manipulación de información
- ✓ XMLHttpRequest, para el intercambio asíncrono de información
- ✓ JavaScript, para unir todas las demás tecnologías (22)

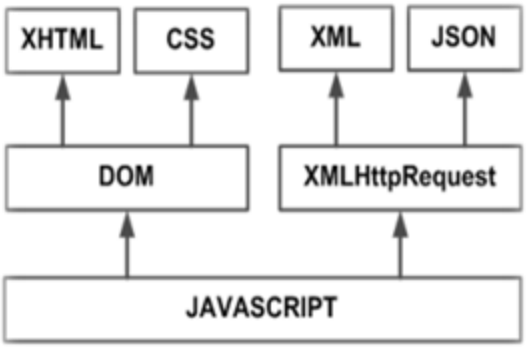


Figura 2 Tecnologías agrupadas bajo el concepto de AJAX.

En las aplicaciones web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, entre otras opciones.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario. (22)

En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX: (remitirse a la figura 3)

Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto. (22)

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano. Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor. (22)

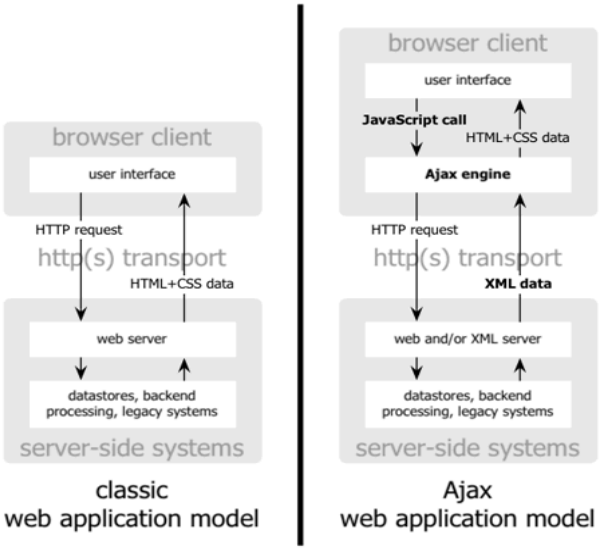


Figura 3 Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX.

El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interacción síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX. (22)

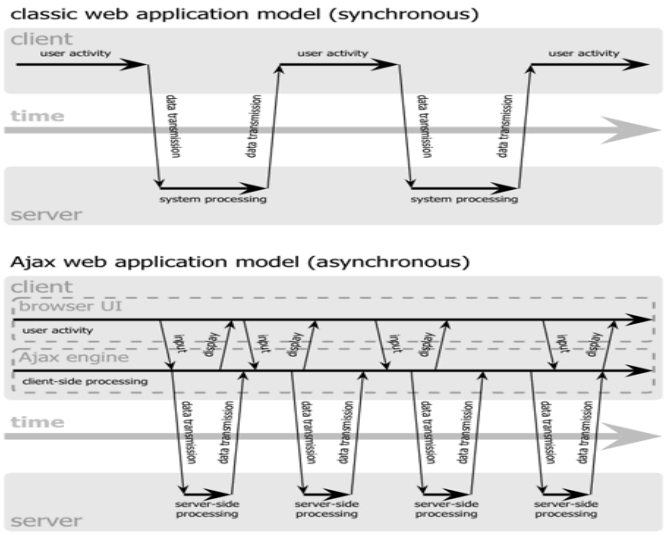


Figura 4 Comparación entre las comunicaciones síncronas de las aplicaciones web tradicionales y las comunicaciones asíncronas de las aplicaciones AJAX.

2.4.1.4 HTML

HyperText Markup Language (Lenguaje de Marcado de Hipertexto) o como se conoce por sus siglas HTML es el lenguaje con el que se desarrolla gran parte de las páginas web. Se usa principalmente para describir la estructura y el contenido en forma de texto además de otros elementos que componen una página web. Este lenguaje es reconocido por todo el mundo y sus normas están definidas por una empresa llamada World Wide Web Consortium, esto permite que una misma página HTML sea visualizada de forma similar en todos los navegadores existentes en cualquier sistema operativo.

2.4.2 Lenguaje del lado del Servidor

2.4.2.1 Personal Home Page (PHP)

PHP es un lenguaje interpretado del lado del servidor utilizado en la realización de páginas web dinámicas soportado por HTML.A diferencia de otros lenguajes como Java o JavaScript, PHP se ejecuta en el servidor permitiendo acceder a los recursos que se encuentran en el servidor, las instrucciones realizadas en PHP son ejecutadas en el servidor y el resultado de estas se envía al cliente por medio de una página HTML.

Por ser un lenguaje libre posee una gran cantidad de características entre ellas:

- ✓ Soporte para una gran variedad de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras
- ✓ Integración con varias bibliotecas externas
- ✓ Multiplataforma(Linux, Microsoft Windows, Mac OS X, RISC OS, Unix)
- ✓ Soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y muchos otros
- ✓ Soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros **(23)**

Además de estas funciones posee otras implementadas que agilizan su comprensión así como su uso en el desarrollo de las aplicaciones. Por su propiedad de ser multiplataforma ha tenido una gran aceptación en la comunidad de desarrollo de aplicaciones web a nivel mundial.

2.5 Framework

Básicamente un framework es una especie de esqueleto, un conjunto librerías o bibliotecas las cuales contienen muchas clases útiles, reutilizables y que nos permiten basarnos en ellas para escribir nuestras aplicaciones.

2.5.1 Symfony

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación **(24)**.

El patrón de arquitectura que usa Symfony para separar los datos, la interfaz de usuario y la lógica en sus aplicaciones es el modelo-vista-controlador. El modelo es el componente que se encarga del acceso a los datos, la vista se encarga de mostrarles a los usuarios la información tomada del modelo en páginas web y el controlador responde a eventos lo que quiere decir que las peticiones realizadas por los usuarios las transforma en operaciones sobre la vista y el modelo.

Symfony para el acceso a la base de datos utiliza dos herramientas ORM (object-relational mapping) estas son Propel y Doctrine, en la última versión estable en la que se propone implementar el sistema se utiliza para el mapeo de la base de datos Doctrine. Su principal ventaja sobre Propel es el rendimiento en ejecución y la forma tan concisa en la que se pueden escribir consultas muy complejas. **(25)**

Actualmente Symfony es uno de los framework más usados en internet por los desarrolladores de aplicaciones web por su profesionalidad y su alta seguridad. Cuenta con bastante documentación en varios idiomas además de estar desarrollado completamente en php5 y ser multiplataforma.

2.6 Entorno de Desarrollo Integrado (IDE)

Los IDE no son más que varios programas agrupados en una herramienta, siendo de gran utilidad para los programadores a la hora de desarrollar sus propias aplicaciones. Podemos decir además que pueden llegar a estar destinados a uno o varios lenguajes de programación.

2.6.1 Eclipse

Eclipse es una de estas herramientas a las que se conoce como entornos de desarrollo integrado. En ella se pueden desarrollar aplicaciones en diferentes lenguajes de programación, entre los que encontramos C++, Java,Ruby, PHP, entre otros, constituyendo así una potente plataforma de programación, desarrollo y compilación. Tiene editores de sintaxis resaltadas, asistente de etiquetas, entre otros módulos, pero lo más llamativo es que permite la instalación de plugin integrando otros lenguajes de programación o herramientas en el mismo IDE como: el plugin de Aptana, otro IDE que puede ser integrado en Eclipse, el plugin para la programación en PHP, herramientas UML, editores visuales de interfaces, entre otros, todo esto amplía de una forma u otra el entorno de desarrollo.

Si bien Eclipse es multiplataforma no todos los plugin lo son, obligando en ciertos casos a depender de una plataforma específica para desarrollar. Los plugins que se desarrollan por la comunidad no tienen todas las funcionalidades que presentan en otras herramientas comerciales. Consume gran cantidad de recursos en la PC lo que conlleva al desarrollador a depender de un buen hardware para su uso.

2.6.2 NetBeans

El IDE NetBeans es un entorno de desarrollo integrado, ganador de premio disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente web, empresa, escritorio y aplicaciones móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby onRails , Groovy y Grails, y C / C + +.

El proyecto NetBeans es apoyada por una vibrante comunidad de desarrolladores y ofrece amplia documentación y formación de recursos, así como una variada selección de terceros plugins.

NetBeans IDE 6.9 introduce el Compositor de JavaFX, una herramienta de diseño visual para construir visualmente aplicaciones JavaFX interfaz gráfica de usuario, similar a la del constructor Swing GUI para aplicaciones Java SE. Otros puntos destacados incluyen la interoperabilidad OSGi para NetBeans aplicaciones de plataforma y apoyo para el desarrollo de paquetes OSGi con Maven, el apoyo a JavaFX SDK, PHP Zend Framework, y Ruby onRails 3.0, así como mejoras en el editor de Java, Java depurador, seguimiento de problemas, y más . **(26)**

El proyecto NetBeans ofrece una versión del IDE a medida para el desarrollo de sitios web PHP que comprenden una variedad de secuencias de comandos y lenguajes de marcado. El editor de PHP es dinámicamente integrada con HTML, JavaScript y CSS funciones de edición. El IDE NetBeans es totalmente compatible con el desarrollo iterativo, por lo que las pruebas proyectos PHP sigue los patrones clásicos familiar para los desarrolladores web.

Permite crear proyectos basados en PHP en el marco de Zend o Symfony. Filtra y ayuda a ver los comandos de Zend o Symfony, especifica los parámetros de comando, una vista previa del comando, y lo ejecuta. También puede asignar accesos directos a los comandos. Dependiendo del proyecto, la lista incluye ya sea comandos Doctrine o Propel. **(27)**

2.6.3 ¿Por qué NetBeans?

Después de estudiar las características de los Entornos de Desarrollo Integrado (IDE) se llega a la conclusión de que el IDE con el que se le daría solución óptima al objetivo general es el Netbeans. Las versiones más recientes permiten integrar el framework Symfony posibilitando desarrollar de forma más eficiente y organizada nuestras aplicaciones, además de los lenguajes de programación mencionados anteriormente entre los que se encuentran php, javascript entre otros. Por ser usado por una gran parte de los desarrolladores de aplicaciones web del mundo posee mucha documentación en diversos formatos, tanto audiovisuales como en texto. Al poseer una interfaz agradable e intuitiva los desarrolladores se sienten identificados y a gusto. Por todo lo expuesto se propone la utilización de este IDE para el desarrollo de la aplicación propuesta.

2.7 Sistemas Gestores de Bases de Datos

Un sistema gestor de bases de datos no es más que un software que se encarga del almacenamiento y el manejo de los datos de forma sencilla y organizada. Manteniendo así su integridad, control y privacidad.

2.7.1 Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de páginas web pasa lo mismo: como es un sistema muy caro no está tan extendido como otros Sistemas Gestores de Bases de Datos, por ejemplo, Access, MySQL, SQL Server, entre otras. **(28)**

Este gestor de bases de datos es uno de los más potentes que existen hoy en día para crear grandes bases de datos donde la principal característica sea el manejo de forma más fácil y segura la información. Hoy en día Oracle tiene versiones para trabajar en sistemas operativos libres pero mantienen sus licencias propietarias por lo que hay que pagar para poder usarlo.

2.7.2 MySQL

El software MySQL proporciona un servidor de base de datos SQL (Structured Query Language) muy rápido, multi-threaded, multi usuario y robusto. El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido. MySQL es una marca registrada de MySQLAB. El software MySQL tiene una doble licencia. Los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU General Public License o pueden adquirir una licencia comercial estándar de MySQL AB**. (29)**

2.7.3 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[6] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el

[6](*Berkeley Software Distribution*).Es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público.

sistema continuará funcionando. 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. **(30)**

2.7.4 ¿Por qué PostgreSQL?

Después de haber estudiado las características de los gestores de bases de datos expuestos anteriormente se determinó usar como gestor de bases de datos PostgreSQL por las características expuestas anteriormente, además de que brinda soporte para la implementación de consultas complejas. Con PostgreSQL se logrará mantener la base de datos actualizada en todo momento, confiable y configurable, así como su integración con las demás aplicaciones propuestas.

2.8 Conclusiones

En el capítulo se presentaron los resultados del estudio y la investigación de las tendencias y las tecnologías actuales en el proceso de desarrollo de software, se describieron las características principales de todas las herramientas expuestas. Se determinaron las herramientas que se usarán en la realización del Repositorio de Componentes, la metodología RUP; UML como lenguaje de modelado; Visual Paradigm como herramienta case, PHP como lenguaje de programación, como framework Symfony, para el tratamiento de las interfaces Ext, como gestor de bases de datos PostgreSQL; como IDE de desarrollo NetBeans por la facilidad que tiene de integrar los lenguajes de programación que se escogieron así como el framework.

CAPÍTULO 3: Presentación de la solución propuesta.

3.1 Introducción

En este capítulo se hará una descripción de la solución propuesta a través de la modelación del negocio, se describirán los actores y trabajadores además de los diagramas de casos de uno del negocio y los diagramas de actividades correspondientes. Se detallarán los requisitos funcionales y no funcionales del sistema así como los diagramas de casos de uso y la descripción de los casos de uso que se consideran críticos.

3.2 Modelo de Negocio

3.2.1 Actores negocio

Partiendo de un análisis de los procesos relacionados con la gestión de los componentes en los diferentes proyectos del departamento, se obtuvo toda la información referente al actor que interviene en el negocio.

Actor	Descripción
Cliente	Puede solicitar almacenar o recuperar un compone en cualquier momento que lo desee.
Desarrollador	Puede solicitar la entrega de un componente realizado en cualquier momento.

Tabla 1 Descripción de Actores del Negocio.

3.2.2 Diagrama de Casos de Uso del Negocio

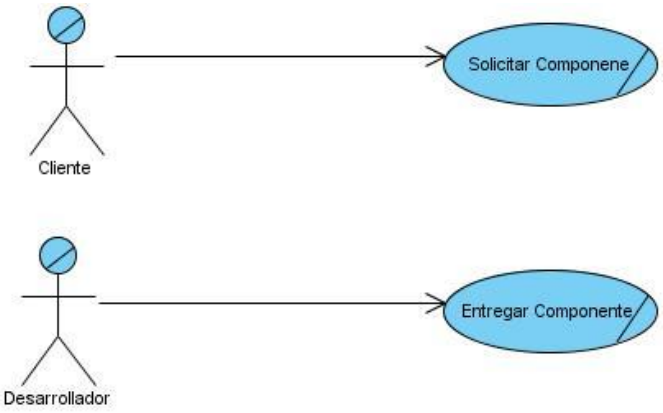


Figura 5 Diagrama de Casos de Uso del Negocio.

3.2.3 Descripción textual de los Casos de Uso de Negocio

Caso de Uso del Negocio	**Solicitar Componente**
Actor	Cliente
Resumen	El caso de uso se inicia cuando el cliente solicita un producto.
Casos de Uso asociados	-

Acción del actor	**Respuesta del proceso de negocio**
El cliente solicita un componente.	1.1 Se verifica la identidad del cliente para determinar si tiene la autoridad requerida. 1.2 Se verifica si existe el componente.

	1.3. Si existe se procede a la entrega del componente.
Otras secciones	Si no existe el componente se le informa al cliente.
Mejoras propuestas.	Se automatizarán la actividad solicitud de los componentes emitida por el cliente, así como el proceso de verificación, entrega y de información al cliente.

Tabla 2 Descripción del Caso de Uso de Negocio Solicitar Componente.

Para lograr comprender el negocio se presenta a continuación la descripción textual, así como el diagrama de actividades.

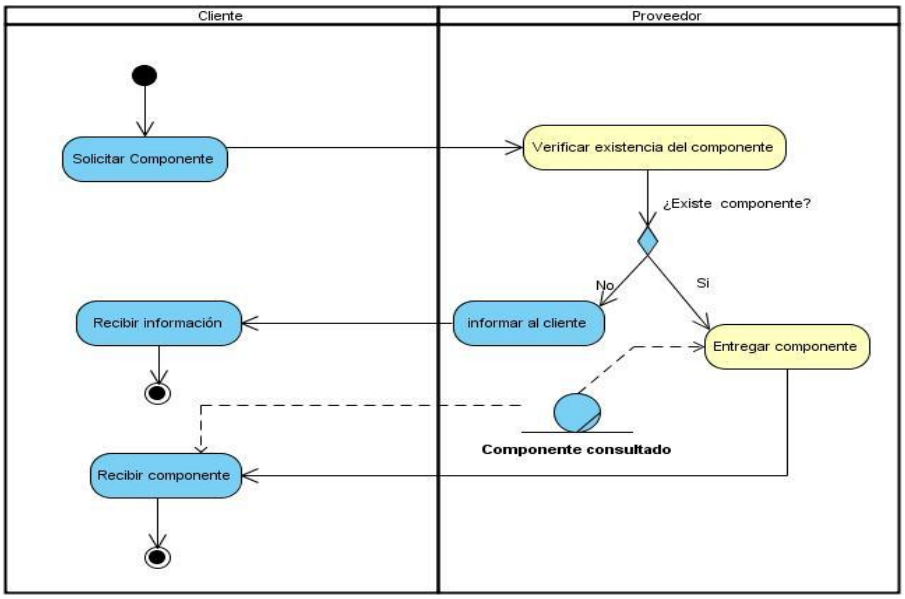


Figura 6 Diagrama de actividades del CUN Solicitar Componente.

Caso de Uso del Negocio	Entregar Componente
Actor	Desarrollador
Resumen	El caso de uso se inicia cuando un desarrollador realiza un componente y desea entregarlo.
Casos de Uso asociados	-

Acción del actor	Respuesta del proceso de negocio
El desarrollador solicita entregar un componente.	1.1 Se verifica la identidad del desarrollador para determinar si tiene la autoridad requerida. 1.2. Si el componente cumple con los requerimientos y normas de calidad se procede a la recepción del mismo.
Otras secciones	Si no tiene la autoridad requerida o no cumple el componente con requerimientos se le informa al desarrollador.
Mejoras propuestas	Se automatizarán la actividad entrega de los componentes emitida por el desarrollador.

Tabla 3 Descripción del Casos de Uso de Negocio Entregar Componente.

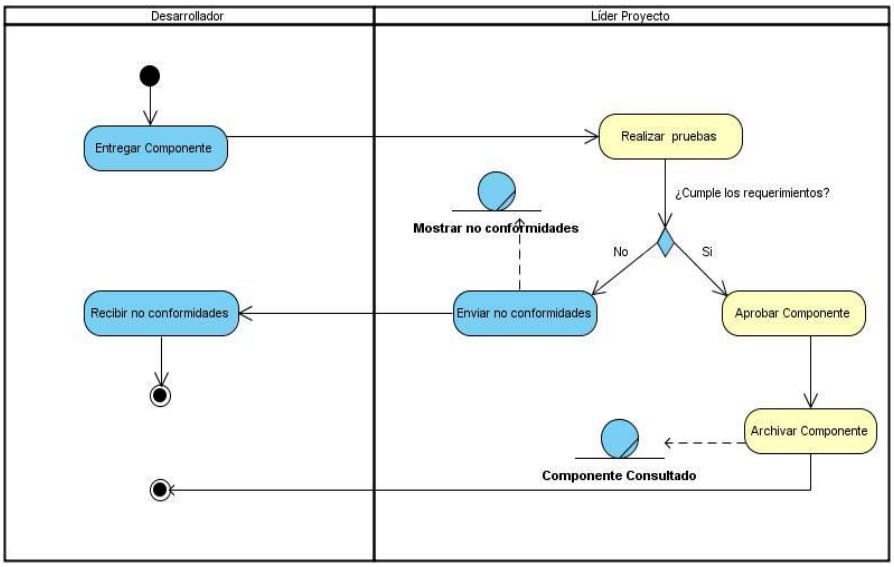


Figura 7 Diagrama de actividades del CUN entregar Componente.

3.3 Requerimientos Funcionales

A continuación se presentan los requerimientos o condiciones que el sistema debe cumplir:

RF 1. Autenticar usuario. El sistema permitirá que los usuarios se autentiquen de forma segura.

RF 2. Gestionar componente

RF 2.1. Insertar componente. El sistema debe ser capaz de permitir almacenar un componente.

RF 2.2. Mostrar componente. El sistema debe ser capaz de permitir mostrar los componentes que existan en la Base de Datos.

RF 2.3. Eliminar componente. El sistema debe permitir eliminar un componente del sistema.

RF 3. Controlar versiones. El sistema debe ser capaz de controlar las diferentes versiones de los componentes almacenados en la base de datos.

RF 4. Gestionar usuarios.

RF 4.1. Adicionar usuario automáticamente. El sistema debe permitir adicionar usuarios automáticamente a la base de datos.

RF 4.2.Modificar roles o permisos. El sistema debe permitir modificar los permisos que presentan los usuarios en la base de datos.

RF 4.3. Eliminar usuario. El sistema debe permitir eliminar usuarios de la base de datos.

RF 5. Notificar errores. El sistema debe permitir notificar los errores que contengan los componentes insertados por los usuarios.

RF 6. Listar componentes. El sistema debe tener una interfaz donde se muestren los componentes almacenados.

RF 7. Realizar búsqueda. El sistema debe permitir realizar búsqueda de los componentes existentes por nombre o descripción.

RF 8. Generar reportes. El sistema debe permitir al administrador generar diferentes reportes. (Últimas versiones agregadas, errores por componentes y versiones, comportamiento histórico de componentes y versiones, entre otros)

RF 9. Realizar descargas. El sistema debe permitir que los usuarios puedan descargar los componentes que se encuentran en el repositorio.

3.4 Requerimientos no funcionales

Los requerimientos no funcionales no son más que las propiedades o cualidades que un producto debe tener.

- ✓ **Requerimientos de Software**

  Para Windows:

  Sistema Operativo: Windows XP o superior.

  Gestor de Base de Datos: PostgreSQL 8.2.

  Servidor Web con soporte para PHP 5.3 o superior

  Para GNU/Linux

  Sistema Operativo: Cualquier distribución de GNU/Linux

Gestor de Base de Datos: PostgreSQL 8.2.

Servidor Web con soporte para PHP 5.3 o superior

- ✓ **Requerimientos de Hardware**

Procesador: Dual Core1.8 GHz.

Memoria RAM: Mínimo, 1GB DDR2 o superior.

Tarjeta de Red: Ethernet a 100 Mbps o superior.

Capacidad de almacenamiento en disco duro: La capacidad de almacenamiento varía en dependencia de las condiciones donde se despliegue el sistema. Su desempeño en un ambiente donde exista un desarrollo elevado de componentes no debe ser inferior a160Gigas. En caso contrario con 80 sería suficiente.

- ✓ **Requerimiento de Seguridad**

Solo tendrá acceso al sistema el personal autorizado de acuerdo a los roles de cada usuario, logrando que solo puedan ejecutar las acciones para las que tienen permisos.

- ✓ **Requerimiento de Disponibilidad**

El sistema debe estar disponible en todo momento, garantizando que pueda ser consultado por los usuarios ante cualquier situación.

- ✓ **Requerimiento de Apariencia o Interfaz Externa**

El sistema debe tener una interfaz amigable, donde el usuario pueda sentirse cómodo y le permita realizar diferentes acciones con facilidad. Además debe ser rápida, flexible y adaptable al entorno.

- ✓ **Requerimiento de restricciones en el diseño y la implementación**

El sistema deberá ser desarrollado como un sistema Web y debe ser funcional en diferentes plataformas.

- ✓ **Requerimientos de Usabilidad**

El sistema podrá ser manipulado por cualquier tipo de usuarios no necesariamente tienen que ser informáticos por lo que todas sus funcionalidades deben ser claras, sencillas y accesibles de manera intuitiva.

3.5 Descripción del Sistema Propuesto

El sistema propuesto para desarrollar es una aplicación web, la cual será de mucha utilidad para el departamento de señales digitales, el cual necesita de un repositorio o servidor para almacenar todos los componentes desarrollados en el mismo así como su documentación.

3.5.1 Descripción de los actores

Actor	Descripción
Cliente	El cliente puede autenticarse e interactuar con el sistema.
Administrador del Repositorio	Es el encargado mantener el sistema en buen estado, manejar las configuraciones del mismo, eliminar los ficheros innecesarios, así como detener y reiniciar los servicios en caso que sea necesario.

Tabla 4 Descripción de actores del sistema.

Casos de Uso del Sistema

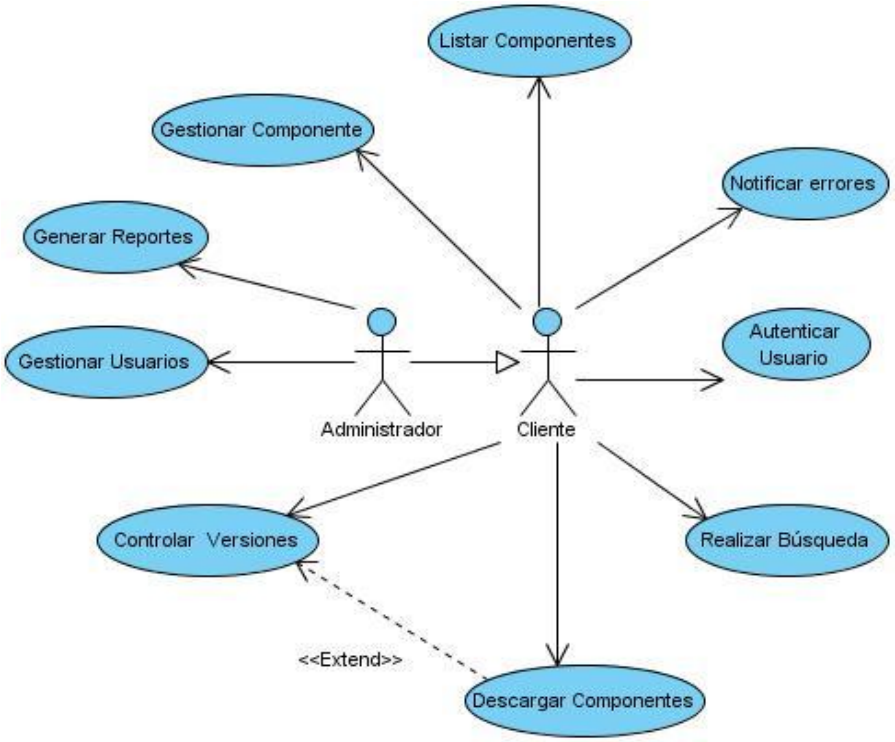


Figura 8 Diagrama de Casos de Uso del Sistema

3.6 Descripciones de los casos de uso del sistema

Se muestra a continuación la descripción para los casos de uso Gestionar Componente, Listar Componente, Gestionar Usuario y Controlar Versiones por la importancia que representan para comprender el funcionamiento del sistema, las descripciones correspondientes al resto de los casos de usos se pueden consultar en los anexos.

3.6.1 Descripciones de casos de uso del sistema Gestionar Componente

Caso de Uso:	Gestionar componente
Actores:	Administrador

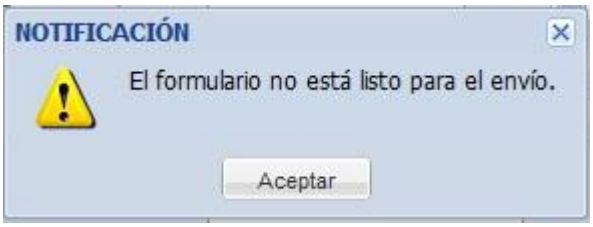
Resumen:	En este caso de uso se adicionan, se muestran los componentes y solo se podrán eliminar por los usuarios que previamente lo insertaron o por el administrador, el cual tiene control total sobre todas las funcionalidades del sistema.
Precondiciones:	Para adicionar un componente el usuario debe estar autenticado ya sea con el rol de administrador o editor.
Referencias	RF2
Prioridad	Critico

Flujo Normal de Eventos		
"Gestionar Componente"		
Acción del Actor		**Respuesta del Sistema**
1	El usuario accede a la opción Nuevo componente.	El sistema muestra una interfaz para que el usuario seleccione una de las siguientes opciones: ✓ Sección nuevo componente. (ver sección) ✓ Sección eliminar componente. (ver sección)
	Sección "Adicionar Componente"	

Actor		Sistema
1	El usuario selecciona la opción adicionar componente.	El sistema muestra la interfaz en la que el usuario debe insertar el componente, ésta contiene varios campos. Nombre(admite solo letras) Versión(debe estar compuesta por #.# o #) Insertar componente(debe ser compactado .rar) Insertar documentación(debe ser compactada .rar) Insertar pruebas (debe ser compactada .rar) Descripción (admite solo hasta 255 caracteres)
2	El usuario inserta el componente así como su documentación y las pruebas.	El sistema verifica que los datos del componente sean válidos.
		El sistema inserta el componente en la base de datos y envía un mensaje de notificación.

Prototipo de Interfaz

Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
1	El usuario inserta los datos del componente incorrectos.	Muestra un mensaje de error.
2	El usuario deja vacío un campo obligatorio.	Muestra un mensaje de error.



Prototipo de Interfaz

Poscondiciones	Se adicionó correctamente el componente en el sistema.

Sección "Eliminar componente**"**		
Precondiciones:	Para eliminar un componente debe estar autenticado con el rol de administrador.	
Actor		**Sistema**
1	El administrador selecciona el componente que desea eliminar.	El sistema verifica que el usuario tenga los permisos para eliminarlo y muestra el componente.
2	El administrador elimina el componente.	El sistema elimina el componente, actualiza la base de datos y muestra un mensaje de notificación.

Prototipo de interfaz administrador

Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
1	El administrador no escoge un componente	Muestra un mensaje de error.

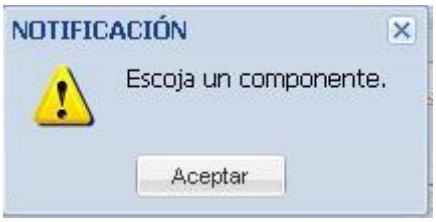
 ***Prototipo de Interfaz***	
Poscondiciones	Se eliminó correctamente el componente del sistema.

Tabla 5 Descripción del caso de uso del sistema Gestionar Componente

3.6.2 Descripciones de casos de uso del sistema Listar Componente

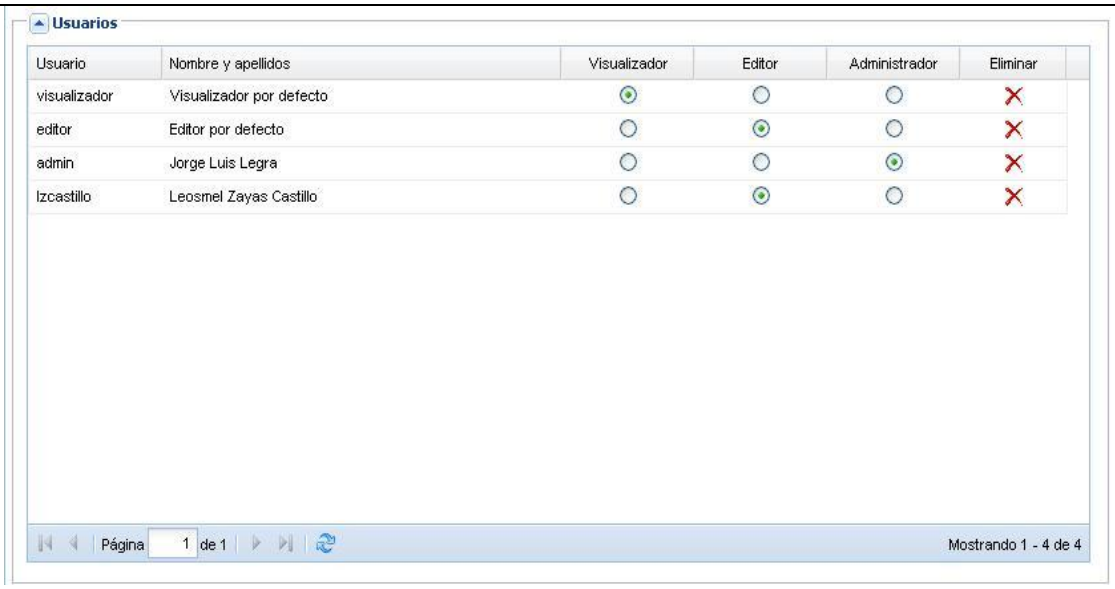
Caso de Uso:	Listar Componente
Actores:	Cliente
Resumen:	Este caso de uso muestra los componentes que se encuentran disponibles en el repositorio.
Precondiciones:	El usuario debe estar autenticado.
Referencias	RF6
Prioridad	Crítico
Flujo Normal de Eventos	
Listar componentes	

Acción del Actor		Respuesta del Sistema
1	El usuario accede a la interfaz listar componentes.	El sistema muestra la interfaz listar componentes.
2	EL usuario selecciona el componente deseado.	El sistema lista todas las versiones del componente.
3	El usuario selecciona la versión del componente que desea ver.	El sistema muestra toda la información relacionada con el componente.
Prototipo de Interfaz 		
Poscondiciones	Quedaron mostrados los componentes que seleccionó el usuario.	

Tabla 6 Descripción del caso de uso del sistema Listar Componente

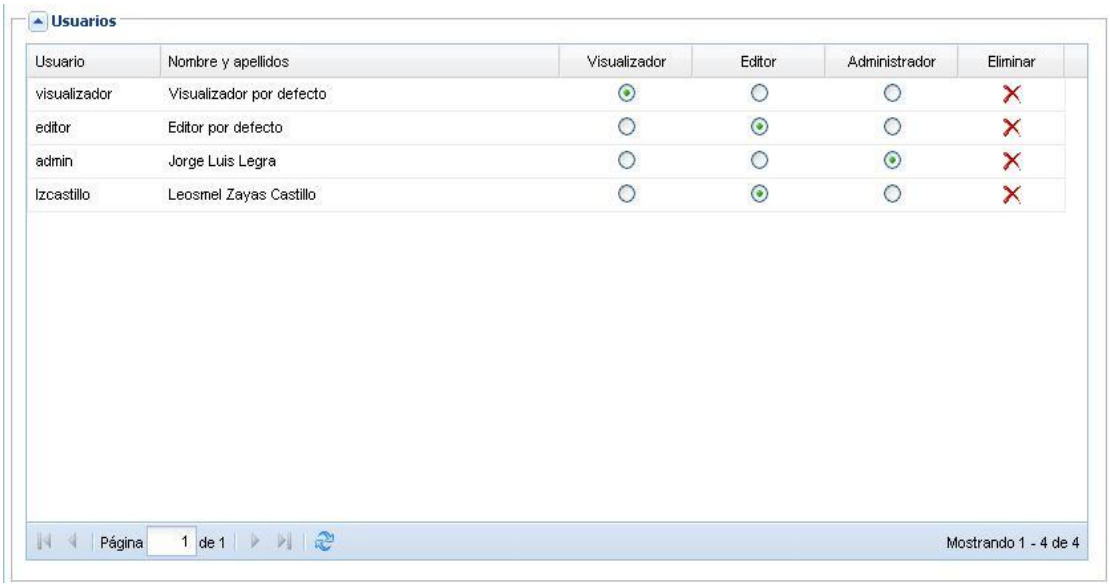
3.6.3 Descripciones de casos de uso del sistema Gestionar Usuario

Caso de Uso:	Gestionar usuario	
Actores:	Administrador.	
Resumen:	En este caso de uso se realizan todas las actividades correspondientes a la gestión de los usuarios en el sistema.	
Precondiciones:	Se debe estar autenticado.	
Referencias	*R4*	
Prioridad	Critica	
Flujo Normal de Eventos		
	Sección "Modificar Permiso"	
1	El administrador selecciona la opción permisos.	El sistema muestra la interfaz gestión de usuarios y permisos.
2	El administrador escoge el usuario al que desea modificar sus permisos.	El sistema almacena las modificaciones.
Prototipo de Interfaz		



Sección "Eliminar usuario"

1	El administrador selecciona la opción permisos.	El sistema muestra la interfaz gestión de usuarios y permisos.
2	El administrador selecciona el usuario a eliminar.	El sistema elimina los datos referentes a ese usuario.



Prototipo de Interfaz

Poscondiciones	Se modificaron los permisos y eliminaron los usuarios.

Tabla 7 Descripción del caso de uso del sistema Gestionar Usuario.

3.6.4 Descripciones del caso de uso del sistema Controlar Versiones

Caso de Uso:	Controlar versiones
Actores:	Cliente
Resumen:	Este caso de uso muestra el control de versiones de los componentes existentes en el repositorio.
Precondiciones:	El cliente previamente debió seleccionar el componente al que desea agregarle una nueva versión.
Referencias	*R3*

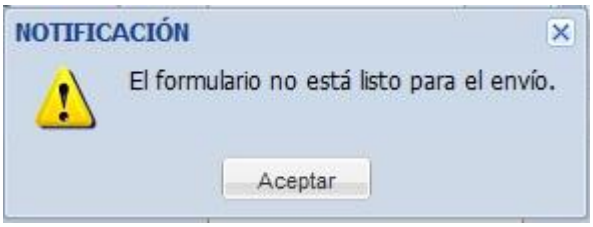
Prioridad	Critica	
Flujo Normal de Eventos		
Control de versiones		
Acción del Actor		**Respuesta del Sistema**
1	El usuario selecciona la opción insertar nueva versión.	El sistema muestra la interfaz en la que el usuario debe insertar una nueva versión del componente, integrada por varios campos. Nombre(debe escoger el nombre del componente) Versión(debe estar compuesta por #.# o #) Insertar componente(debe ser compactado.rar) Insertar documentación(debe ser compactada .rar) Insertar pruebas (debe ser compactada .rar) Descripción(admite solo hasta 255 caracteres)
2	El usuario inserta todos los datos relacionados con la nueva versión.	El sistema verifica los datos e inserta la nueva versión del componente en la

		base de datos y envía un mensaje de notificación.

Prototipo de Interfaz

Poscondiciones	Se insertó la nueva versión del componente en el repositorio.

Acción del Actor		**Respuesta del Sistema**
1	El usuario inserta los datos de la nueva versión del componente incorrectos.	Muestra un mensaje de error.
2	El usuario deja vacío un campo obligatorio.	Muestra un mensaje de error.



Prototipo de Interfaz

Poscondiciones	Se adicionó correctamente la nueva versión de componente en el repositorio.

Sección "Eliminar versión de un componente**"**

Precondiciones:	Para eliminar una versión el usuario previamente debió seleccionar el componente al que desea eliminar la versión.

	Actor	**Sistema**
1	El usuario selecciona la versión que desea eliminar.	El sistema verifica que el usuario tenga los permisos para eliminar la versión.
2	El usuario elimina la versión del componente.	El sistema elimina la versión del componente, actualiza la base de datos y muestra un mensaje de notificación.

Prototipo de interfaz

Tabla 8 Descripción del caso de uso del sistema Control de Versiones

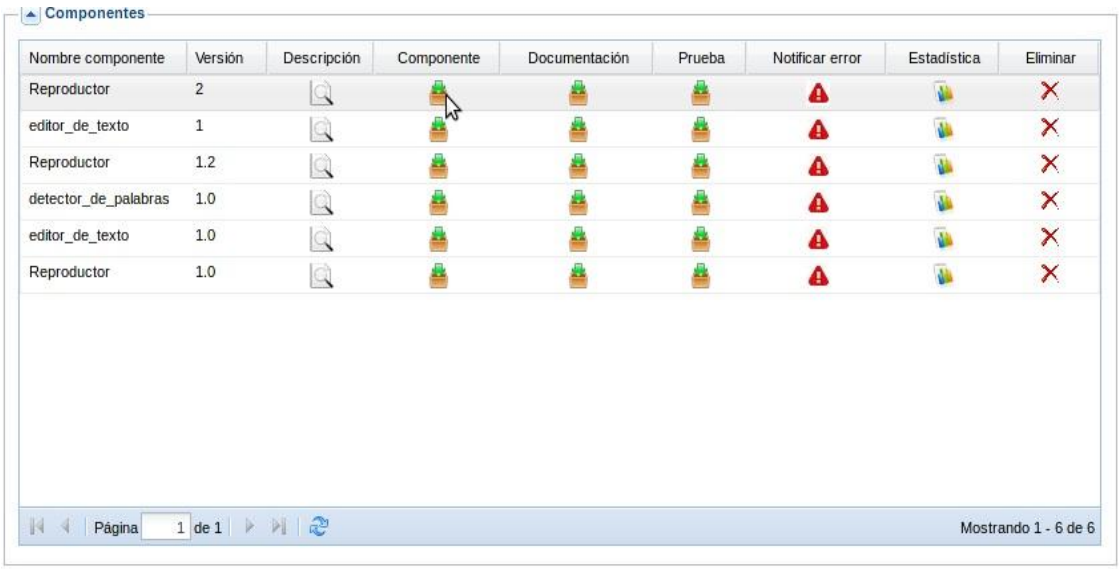
3.6.5 Descripciones de casos de uso del sistema realizar descarga

Caso de Uso:	Realizar descarga
Actores:	Cliente
Resumen:	Este caso de uso se inicia cuando el cliente desea descargar un componente.
Precondiciones:	El cliente debe haber seleccionado el componente que desea descargar.
Referencias	*R9*
Prioridad	Critica

Flujo Normal de Eventos		
Realizar descarga		
Acción del Actor		**Respuesta del Sistema**
1	El usuario selecciona la opción descargar.	El sistema muestra una ventada donde el usuario especifica donde desea almacenar el componente.

2	El usuario selecciona el destino del componente y lo guarda.	

Prototipo de Interfaz



Poscondiciones	Se descargó el componente correctamente.

Tabla 9 Descripción del caso de uso del sistema Realizar Descarga

3.7 Diagrama de Clases del Análisis

La realización del diagrama de clases del análisis es uno de los artefactos que se obtienen al realizar el modelo de análisis. Estos diagramas están compuestos por clases las que se clasifican en Interfaz, Controladora y Entidad.

✓ Las clases interfaz se encargan de modelar la interacción del actor con el sistema.

✓ Las clases controladoras modelan los aspectos dinámicos del sistema, de forma tal, que puedan coordinar las acciones y los flujos de control.

✓ Las clases entidades modelan información que posee una larga vida y que a menudo es persistente. Suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema.

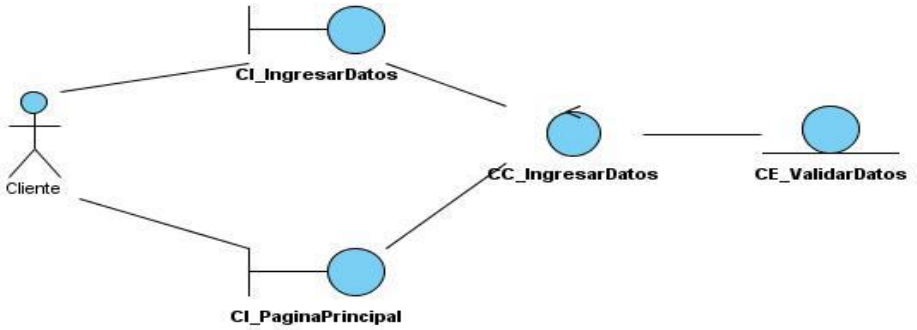


Figura 9 Diagrama de Clases del Análisis Autenticar.

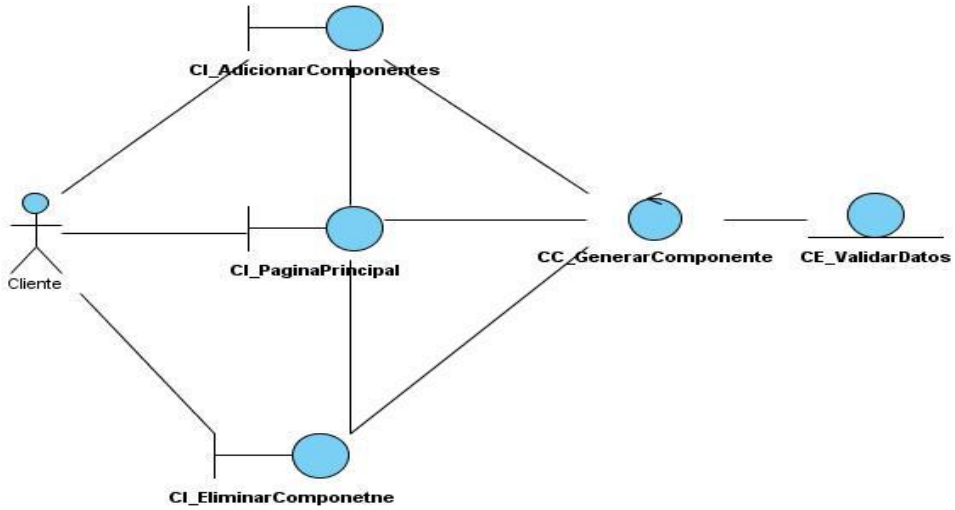


Figura 10 Diagrama de Clases del Análisis Gestionar Componente.

Figura 11 Diagrama de Clases del Análisis Descargar Componente.

3.8 Conclusiones

En este capítulo se presentó la descripción del negocio, los requerimientos funcionales y no funcionales. Se modelaron los diagramas de casos de uso del negocio y del sistema así como los de clases del análisis. Todo esto ayuda a tener una visión más general de la estructura y función del sistema propuesto.

CAPÍTULO 4: Construcción de la solución propuesta

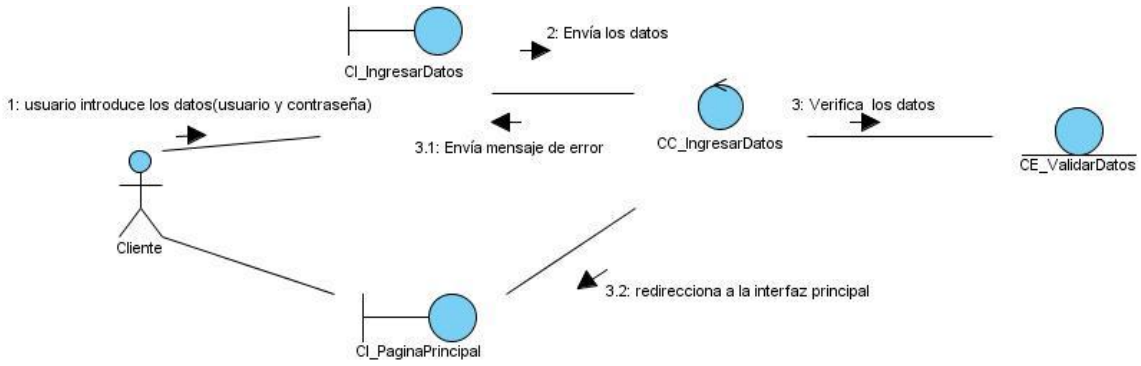
4.1 Introducción

En este capítulo se describirán todos los aspectos fundamentales para la construcción del sistema propuesto como los diagramas de clases del diseño, los que dan punto inicial al flujo de trabajo de implementación. Será explicado además algunos aspectos referidos a la implementación, sus características y funcionalidades.

4.2 Diagramas de Colaboración del Análisis

Un diagrama de colaboración es una forma alternativa al diagrama de secuencia de mostrar un escenario. Este tipo de diagrama muestra las interacciones entre objetos organizadas entorno a los objetos y los enlaces entre ellos. El diagrama de colaboración no muestra el tiempo como una dimensión aparte, por lo que resulta necesario escribir con números de secuencia tanto la secuencia de mensajes como los hilos concurrentes. A continuación se presentará un ejemplo de este tipo de diagrama correspondiente al caso de uso autenticar, los restantes los podrá consultar en los anexos.

4.2.1 Diagrama de Colaboración del Análisis. CU Autenticar



4.3 Diseño

El diseño, lugar donde manda la creatividad, donde los requisitos del cliente, las necesidades de negocio y las consideraciones técnicas se unen en la formulación de un producto o sistema. El diseño crea una representación o modelo de software, pero a diferencia del modelo de análisis (que se enfoca en la representación de los datos, las funciones y el comportamiento requerido), el modelo de diseño proporciona detalles acerca de las estructuras de datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema. **(31)**

4.3.1 Patrones de Diseño

Los patrones de diseño pueden usarse durante el diseño del software. Una vez que se ha desarrollado el modelo de análisis, el diseñador puede examinar una representación detallada del problema que debe resolver y las restricciones que impone el problema.

Patrones arquitectónicos: estos patrones definen la estructura general del software, indican las relaciones entre los subsistemas y los componentes del software y definen las reglas para especificar las relaciones entre los elementos (clases, paquetes, componentes, subsistemas) de la arquitectura.

Patrones de diseño: estos patrones se aplican a un elemento específico del diseño como un agregado de componentes para resolver algún problema de diseño relaciones entre los componentes o los mecanismos para efectuar la comunicación de componente a componente.

Idiomas: a veces llamados patrones de código, estos patrones específicos de lenguaje por lo general implementan un elemento algorítmico o un componente, un protocolo de interfaz específico o un mecanismo de comunicación entre los componentes. **(31)**

4.3.1.1 ¿Qué es un patrón de diseño?

Un patrón de diseño es:

- ✓ una solución estándar para un problema común de programación.
- ✓ una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- ✓ un proyecto o estructura de implementación que logra una finalidad determinada.
- ✓ un lenguaje de programación de alto nivel.

- ✓ una manera más práctica de describir ciertos aspectos de la organización de un programa.
- ✓ conexiones entre componentes de programas.
- ✓ la forma de un diagrama de objeto o de un modelo de objeto.

Existen diversos tipos de patrones, entre ellos encontramos los arquitectónicos y los de diseño. Pressman plantea que los arquitectónicos definen la estructura general del software, indican las relaciones entre los subsistemas y los componentes del software y definen las reglas para especificar las relaciones entre los elementos (clases, paquetes, componentes, subsistemas) de la arquitectura. Por otra parte los de diseños se aplican a un elemento específico del diseño como un agregado de componentes para resolver algún problema de diseño relaciones entre los componentes o los mecanismos para efectuar la comunicación de componente a componente.

Para la construcción de la solución propuesta se utilizarán los patrones GRASP (Patrones Generales de Software para Asignación de Responsabilidades)

Patrón Experto: Al desarrollar un sistema es importante saber asignarle a cada clase las responsabilidades para la que fueron creadas, este patrón recomienda asignarle a una clase sólo aquellas responsabilidades para las que ella contenga la información necesaria para su cumplimiento. Doctrine es la librería externa que utiliza Symfony para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Patrón Creador: Este patrón es muy importante pues al utilizarlo se hace un mejor uso de la memoria del sistema, es de gran relevancia saber asignar la responsabilidad de crear una instancia de una clase sólo a aquella clase que la requiera y así se evita tener objetos repetidos.

Symfony utiliza varios de los patrones GRASP y GOF, dentro de los GRASP se encuentran el *Patrón Controlador y Bajo Acoplamiento.* Los patrones GOF son representados por el patrón *Singleton (*Instancia Única*), Abstract Factory (Fábrica Abstracta), Decorator (Envoltorio) y Composite (Objeto Compuesto).*

El patrón singleton o solitario es utilizado en la clase sfContext la cual almacena una referencia a todos los objetos que forman el núcleo de Symfony debido que se encarga de enrutar todas las peticiones que se realizan a la aplicación.

Abstract Factory (Fábrica Abstracta) Su aplicación se puede evidenciar en la clase sfContext utilizada cuando el *framework* necesita crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea, de esta forma, se simplifica la decisión abstrayéndola dentro de una clase especializada.

Decorator (Envoltorio) Es aplicado a las vistas donde el contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el *layout* decora la plantilla. De manera que el *layout* contiene el código HTML y los elementos comunes a todas las páginas, mientras las plantillas se encargan de mostrar el resultado de las acciones.

Composite (Objeto Compuesto) Symfony lo utiliza en la clase composite.class.php de sfLogpermitiendo procesar de la misma forma elementos individuales y colecciones de elementos.

Podemos generalizar entonces que estos patrones garantizan:

- ✓ que todas las peticiones web sean controladas por un solo controlador frontal
- ✓ que la clase Action herede solamente de sfActions
- ✓ la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia
- ✓ trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí
- ✓ tratar objetos compuestos como si de uno simple se tratase

Todo esto es de gran ayuda para el desarrollador puesto que garantiza que la aplicación sea lo más robusta posible.

4.3.1.2 Patrón arquitectónico

Symfony utiliza uno de los patrones arquitectónico clásicos del diseño web, el Modelo-Vista-Controlador. Este divide la aplicación en 3 capas lógicas.

Modelo: Esta capa administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Esta capa maneja la visualización de la información, es decir que presenta el modelo en un formato adecuado para interactuar, que usualmente es la interfaz de usuario.

Controlador: Esta capa controla el flujo de datos entre la vista y el modelo; es decir que responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. **(32)**

4.3.2 Diagramas de Clases del Diseño

El principal objetivo del diseño es transformar los requerimientos en lo que será el sistema y desarrollar una arquitectura robusta. En el diagrama de clases del diseño se tiene una visión de cómo será el sistema en términos de clases, y cómo quedará estructurado por atributos y métodos.

A continuación se presentan los diagramas de clases del diseño por casos de uso para un mejor entendimiento.

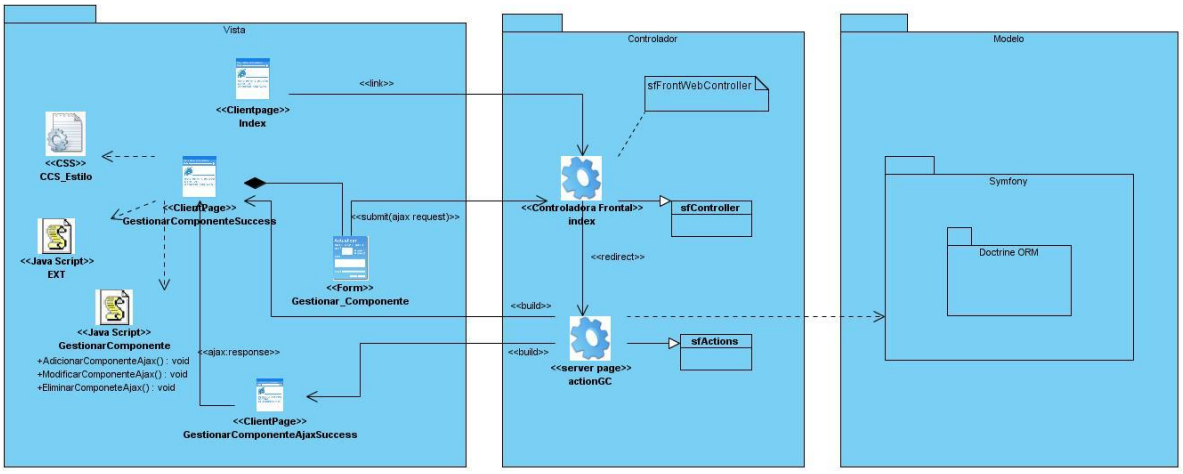


Figura 12 Diagrama de Clases del Diseño Gestionar Componente.

4.3.3 Diseño de la Base de Datos

Para lograr un adecuado funcionamiento del sistema se hace necesario tener un buen diseño de la base de datos. A continuación se muestra el Diagrama Entidad Relación correspondiente al diseño de la Base de Datos.

El Diagrama Entidad Relación muestra las entidades relevantes del sistema así como sus interrelaciones y propiedades.

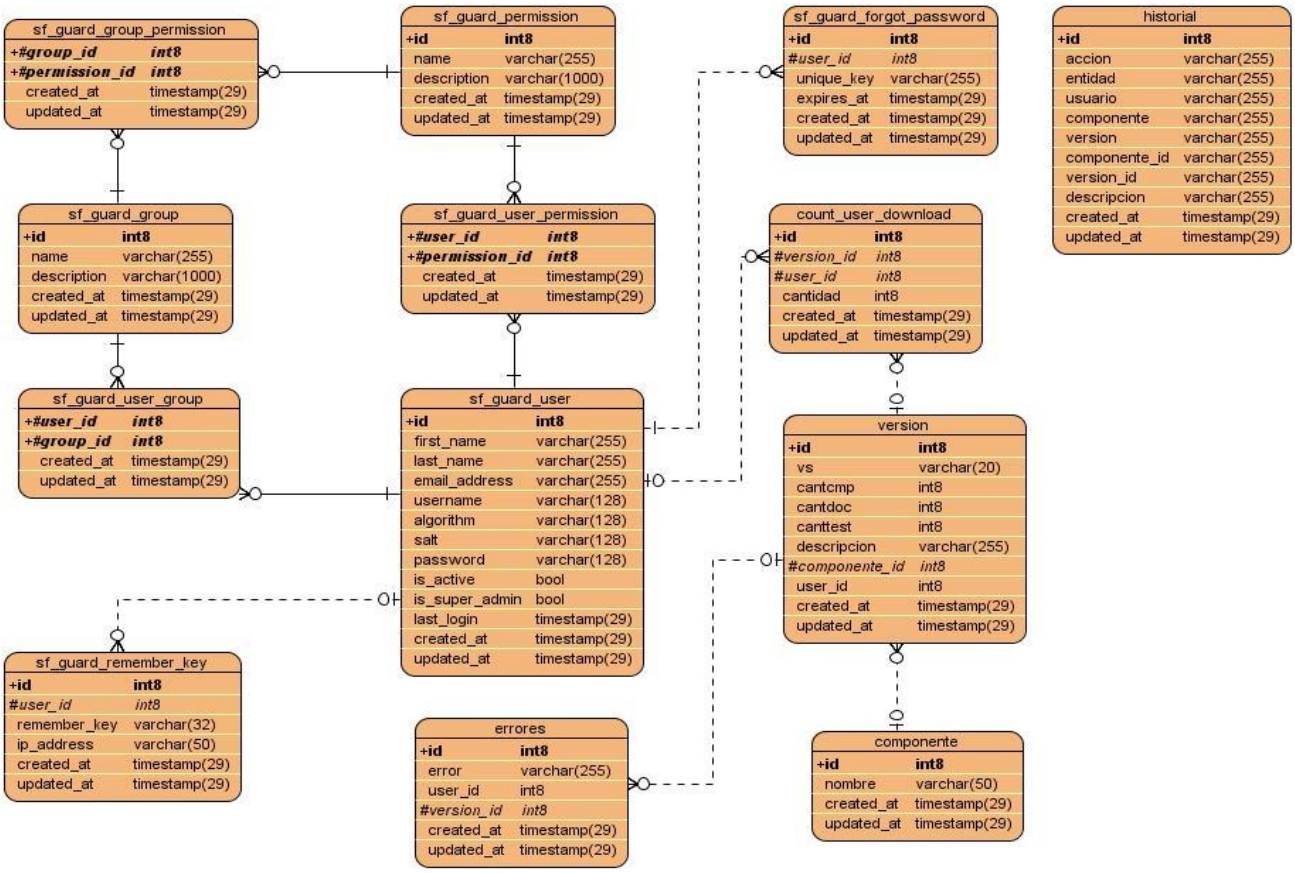


Figura 13 Diagrama Entidad-Relación.

Una vez confeccionado el diseño de la Base de Datos, se escoge para el mapeo de la misma utilizar el objeto relacional Doctrine, el cual ya viene incluido en la versión del framework Symfony utilizado para el desarrollo del sistema.

4.4 Modelo de Implementación

El modelo de implementación inicia con los resultados que se obtuvieron a partir de la modelación del diseño, creando así los componentes físicos del sistema los que se traducen en ficheros de código fuente .php y .js por haber sido implementado con los lenguajes php y javascript.

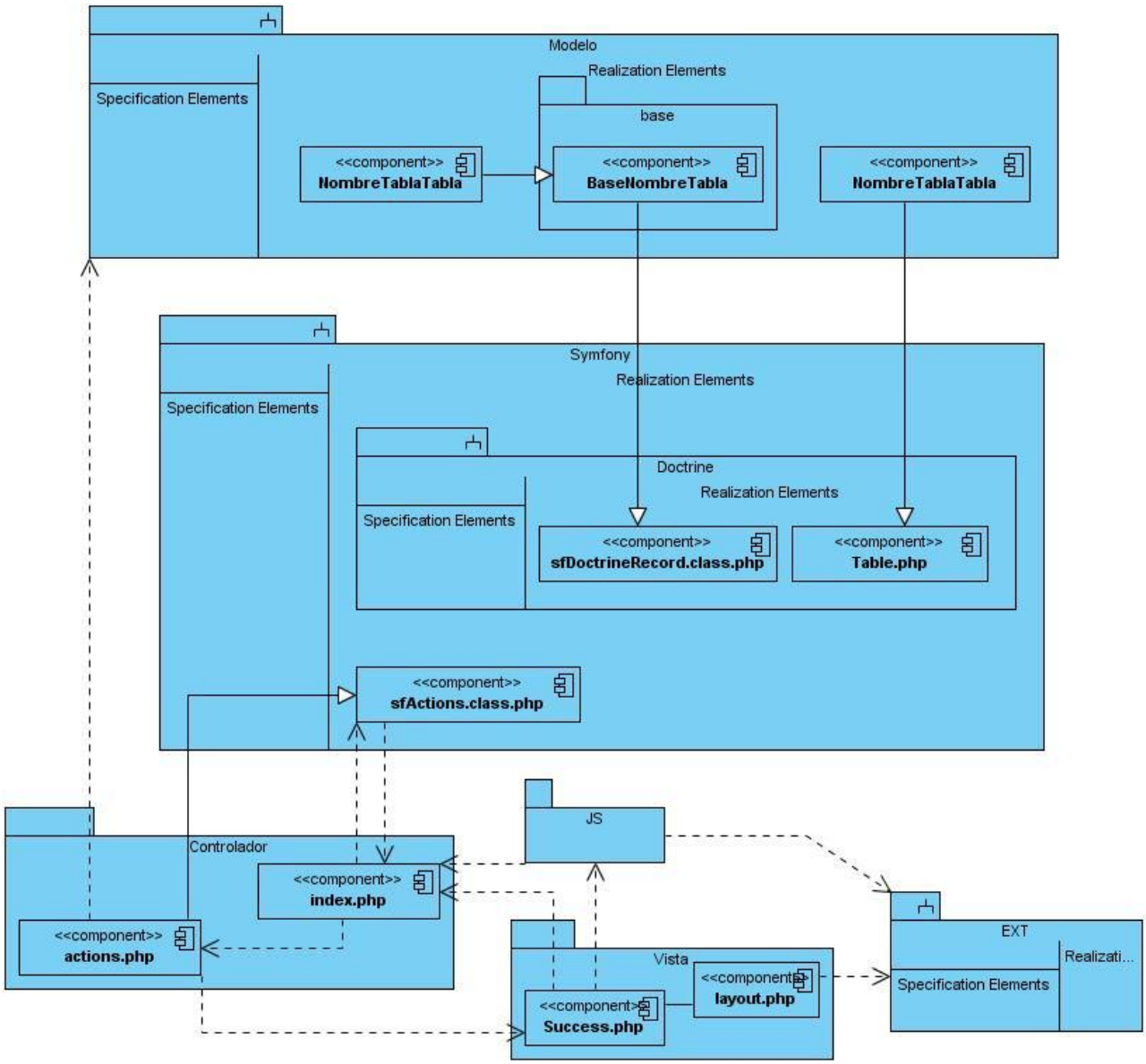


Figura 14 Diagrama de Componentes.

4.4.1 Estándares de Codificación

Los estándares de codificación constituyen el elemento principal por el cual se rigen los desarrolladores para asegurar que el código sea legible para cualquier equipo de trabajo, lo que permite que las tareas realizadas sean más efectivas y contengan menos errores.

A continuación se describen una serie de pautas específicas a seguir para el desarrollo con los lenguajes de programación utilizados en la construcción del sistema:

- Saltos de Línea
    - Añadir un salto de línea después del cierre de los paréntesis de los parámetros.
    - Añadir un salto de línea después un punto y coma, cuando termina la sentencia.
- Concatenación
    - Para concatenar variables se utilizará el operador "." (punto), con un espacio entre medio para mejorar la lectura.
- Espacios y líneas en blanco
    - No se debe poner espacios después de un paréntesis de apertura y antes de un cierre.
      $request->getParameter('idVersion')correcta
      $request->getParameter('idVersion')incorrecta
    - Las funcionalidades deben estar separadas por un espacio o línea en blanco.
    - Para los archivos yml dos espacios en blanco deben ser utilizados.
- Control de flujo
    - En las declaraciones if/else deberá tener un espacio antes y después del paréntesis condicional, en la siguiente línea se abre llave y se cierra en una línea diferente.
- Declaración de variables
    - Los nombres de las variables deben relacionarse con la acción que realiza en el sistema.

Mediante el empleo de un estilo de programación para la confección de las funcionalidades del sistema es posible obtener un código más legible, permitiendo de esta forma la reducción de errores lógicos en el código y su reutilización.

4.4.2 Modelo de Despliegue

El Diagrama de Despliegue muestra la relación existente entre los componentes de hardware y software en el sistema propuesto. Muestra como un conjunto de nodos se comunican mediante los diferentes protocolos de conexiones empleados en el intercambio de información. Estos nodos pueden representar instancias de componentes de software, objetos y procesos.

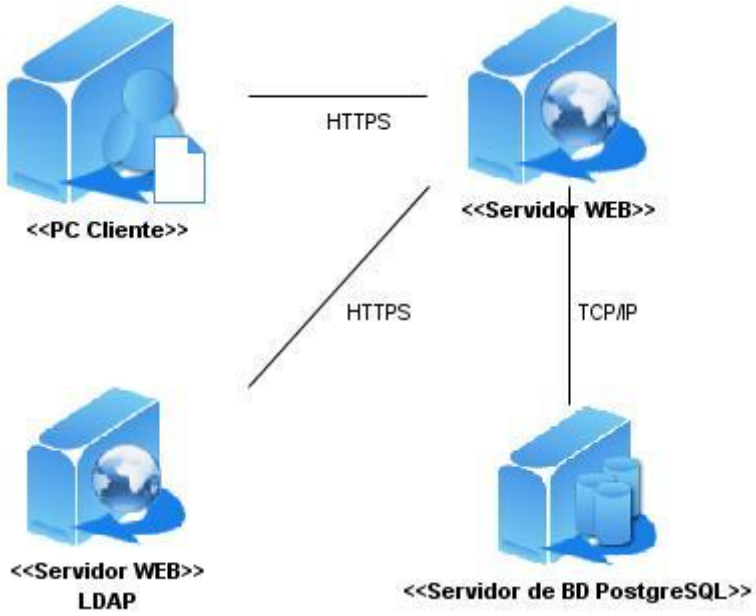


Figura 15 Diagrama de Despliegue.

A continuación se describirán los nodos físicos representados en el diagrama.

Figura 16 Nodo Cliente.

El nodo representa una PC cliente desde la cual se podrá acceder al sistema por medio de un navegador web e interactuar con todas las funcionalidades que este brinda.

Figura 17 Nodo Servidor de Aplicación Web Apache.

El nodo representa el servidor web donde estará alojado el sistema así como los componentes almacenados en el mismo.

Figura 18 Nodo Servidor de Servicios Web en la UCI (LDAP).

El nodo representa el servidor de servicios web que ofrece la universidad. Entre los servicios que ofrece se utiliza el servicio LDAP (Protocolo Ligero de Acceso a Directorios) para la autenticación de los usuarios en el sistema.

Figura 19 Nodo Servidor de Bases de Datos.

El nodo representa el servidor de Base de Datos PostgreSQL en el que estará alojada la base de datos del sistema.

Protocolos usados.

TCP/IP: Protocolo que permite la comunicación del sistema con la Base de Datos.

HTTPS: Protocolo que permite la comunicación de los usuarios con el sistema, está basado en el protocolo HTPP destinado a la copia de datos de Hiper Textos de forma segura.

4.5 Pruebas de Software

Luego que se culmina la realización de cualquier sistema, se procede a realizarles diferentes tipos de pruebas para garantizar que los posibles errores o defectos que estos posean sean corregidos antes de salir al mercado. Existen diversas técnicas de evaluación de software agrupándose en pruebas de caja blanca y pruebas de caja negra.

Las pruebas de caja blanca o estructurales, se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.

Las pruebas de caja negra o funcionales, realizan pruebas sobre la interfaz del programa. No necesariamente se necesita conocer la lógica del sistema si no su funcionamiento.

A continuación se muestra el diseño de casos de prueba de caja negra de la funcionalidad Autenticar Usuario. Los diseños de caso de pruebas correspondientes a los demás Casos de Uso pueden ser consultados en los Anexos.

4.6 Pruebas de Software para la funcionalidad autenticar usuario

- ✓ **Descripción General.**

  El caso de uso se inicia cuando el usuario desea acceder al sistema.

- ✓ **Condiciones de Ejecución.**

  Ninguna.

- ✓ **Secciones a probar en el Caso de Uso.**

Nombre de la secc	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Autenticar Usuario	EC 1.1: Autenticar Usuario	El sistema brinda la posibilidad de autenticarse mediante un usuario y su respectiva contraseña.
	EC 1.2: Tratamiento de errores	El sistema muestra un mensaje de error “Usuario o contraseña. incorrectos ”

Tabla 10 Diseño de caso de prueba del caso de uso Autenticar.

- ✓ **Descripción de variable.**

No	Nombre de campo	Clasificación	Valor Nulo	Descripción

1	Usuario	Campo de texto	No	Se tiene que escribir un usuario.
2	Contraseña	Campo de texto	No	Se tiene que escribir la contraseña.

Tabla 11 Descripción de variables del caso de uso Autenticar

- ✓ **Matriz de Datos.**

Escenario	**Variable 1 Usuario**	**Variable 2 Contraseña**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
EC 1.1: Autenticar usuario.	V	V	El sistema permite autenticarse de forma correcta.	Quedó autenticado el usuario.	1- Autenticar usuario.
EC 1.2: Tratamiento de errores	V	I	El sistema muestra un mensaje de error “Usuario o contraseña incorrectos ”	No pudo acceder al sistema.	No existe coincidencia entre el nombre de usuario y la contraseña.
EC 1.2: Tratamiento de errores	I	I	El sistema muestra un mensaje de error “Usuario o contraseña *incorrectos ”*	No pudo acceder al sistema.	No existe coincidencia entre el nombre de usuario y la contraseña.

Tabla 12 Matriz de Datos del caso de uso Autenticar.

Luego de realizar todos los casos de pruebas a las funcionalidades del sistema, se obtuvieron las siguientes no conformidades las que fueron corregidas y comprobadas en una segunda iteración de las pruebas:

- ✓ La funcionalidad eliminar componente podía ser efectuada por los usuarios con el rol de editores y administrador cuando solo podía ser efectuada por el administrador
- ✓ En la funcionalidad permisos, cuando el administrador buscaba algún usuario y no era exitosa la búsqueda el sistema no mostraba un mensaje notificando que la búsqueda no arrojó resultados

4.7 Conclusiones

En este capítulo fueron expuestos los diferentes diagramas de clases de diseño, los que contribuyeron a realizar una mejor implementación del sistema. Se expusieron las características generales para el desarrollo del sistema, teniendo en cuenta los componentes utilizados y el diseño de la Base de datos, viéndose representada la relación entre estos mediante el diagrama de componente. Se tuvo además una visión de cómo quedaría distribuido físicamente el sistema mediante el diagrama de despliegue y se desarrollaron pruebas mediante la técnica de caja negra las que arrojaron como resultado que el sistema cumple con las funcionalidades requeridas.

CONCLUSIONES GENERALES

Los resultados alcanzados con la realización de la investigación permitieron darle cumplimiento a los objetivos propuestos además de:

- ✓ Establecer las bases teóricas y tecnológicas para la realización de la solución propuesta.
- ✓ Modelar los procesos para almacenar y gestionar componentes.
- ✓ Realizar el análisis y diseño del Repositorio de Componentes.
- ✓ Implementar el Repositorio de Componentes.

La propuesta realiza en el presente trabajo de diploma será de gran ayuda para el departamento de señales digitales pues al implementar el nuevo modelo de producción basado en componentes tendrán la herramienta para almacenar y gestionar los componentes desarrollados en el mismo, así como el control de todas las versiones de los mismos.

RECOMENDACIONES

- ✓ Se recomienda que la propuesta resultante de dicha investigación se aplique en el departamento y se extienda al resto del centro.

- ✓ Agregar nuevas funcionalidades al sistema.
  - Control de cambios.
  - Notificación de cambios.
  - Control de acceso.

BIBLIOGRAFÍA

1. La Nación. [Online] 1999. http://wvw.nacion.com/ln_ee/especiales/siglo/siglo6/siglo3.html.

2. **GARZÁS PARRA, JAVIER / PIATTINI VELTHUIS, MARIO G.** *FABRICAS DEL SOFTWARE: EXPERIENCIAS, TECNOLOGÍAS Y ORGANIZACIÓN.* s.l. : RA-MA EDITORIAL, 2007.

3. **Pressman, R. S. (s.f.).** *Ingeniería de software, Un enfoque práctico.*

4. **Instituto Nacional de Estadística y Geografía de Mexico.** Ciberhabitat. [Online] http://www.ciberhabitat.gob.mx/biblioteca/extensiones/).

5. **EDUARDO ARAGÓN MONTES, JULIÁN CAMILO PÉREZ PARRA.** *SISTEMA PARA EL ALMACENAMIENTO VIRTUAL Y GESTION DE ARCHIVOS.* Cali, Colombia : s.n., 2009.

6. **Facultad de Ciencias Exactas y Naturales y Agrimesura.** Facultad de Ciencias Exactas y Naturales y Agrimesura. [Online] Universidad Nacional del Noreste. http://exa.exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/GESTAR02.htm.

7. **Felix Bachmann, Len Bass,Charles Buhman,Santiago Comella-D,Fred Long,John Robert,Robert Seacord.** *Technical Concepts of Component-Based Software Engineering,2nd Edition.* 2000.

8. **Caroline Batista Spencer Holanda, Clarissa Angélica de Almeida de Souza, Walcélio L. Melo.** *Um Repositório de Componentes para Web Dirigido.*

9. **Maribel Ariza Rojas, Juan carlos Molina García.** *Jerarquia y granularidad de componentes de software para Pymes en Bogota.* Bogotá DC : s.n., 2005.

10. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación.

11. **Departamento de Sistemas Informáticos y Computación Universidad de Valencia.** *Rational Unified Process.* Valencia : s.n.

12. **Beck, Kent.** *Extreme Programming Explained.* s.l. : Addison Wesley, 2000.

13. **Wells, Don.** XP Extreme Programming. *XP Extreme Programming.* [Online] 2009. http://www.extremeprogramming.org/.

14. **José H. Canós, Patricio Letelier yMª Carmen Penadés.** [Online] http://www.willydev.net/descargas/prev/TodoAgil.pdf.

15. **Orallo, Enrique Hernández.** *El Lenguaje Unificado de Modelado (UML).*

16. —. *El Lenguaje Unificado de Modelado (UML).*

17. **Scribd.** Scribd. [Online] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

18. **Innova, Grupo Soluciones.** Grupo Soluciones Innova. [Online] http://www.rational.com.ar/herramientas/roseenterprise.html.

19. **Jalon, Javier Garcia de.** *Aprenda Java como si estuviera en primero.* 2000.

20. **Pérez, Javier Eguíluz.** *Introducción a JavaScript.*

21. **Copyright 2011 Sencha.** Sencha. [Online] http://www.sencha.com/learn/Ext_Getting_Started.

22. **Pérez, Javier Eguíluz.** http://www.librosweb.es/. *http://www.librosweb.es/.* [Online] http://www.librosweb.es/ajax/capitulo1.html.

23. **Mehdi Achour, Friedhelm Betz, Antony Dovgal.** PHP. *PHP.* [Online] the PHP Documentation Group, 1997-2011. http://www.php.net/manual/es/intro-whatcando.php.

24. **Fabien Potencier, François Zaninotto.** *Symfony la guía definitiva.*

25. **Eguiluz, Javier.** symfony.es. [Online] http://www.symfony.es/2008/07/09/comparando-propel-doctrine-y-propelfinder/.

26. **Corporation, Oracle.** NETBEANS. [Online] 2011. http://netbeans.org/community/releases/69/.

27. **Corporation, Oracle.** NetBeans. [Online] 2011. http://netbeans.org/features/php/.

28. **Masip, David.** DesarrolloWeb.com. [Online] http://www.desarrolloweb.com/articulos/840.php.

29. **Oracle Corporation.** MySQL. [Online] http://dev.mysql.com/doc/refman/5.0/es/introduction.html.

30. **PostgreSQL-es.** PostgreSQL-es. [Online] http://www.postgresql-es.org/sobre_postgresql.

31. **Pressman, Roger S.** Ingeniería de Software. *Ingeniería de Software. Un enfoque práctico.* s.l. : Mc Graw Hill, 2001.

32. **Autores, Colectivo de.** *Arquitectura y Patrones de diseño.* 2008-2009.

33. **Ivar Jacobson, Grady Booch y James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000.

34. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999.

35. **Colectivo.** Arquitectura y Patrones de diseño. 2008-2009.

36. **Falgueras, Benet Campderrich.** *Ingeniería de Software.* s.l. : Editorial UOC, 2003.

37. **Gracia, Joaquin.** Patrones de diseño.Anális y Diseño. Ingeniería de Software. [Online] 5 27, 2005. http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

38. **Deciencias.** Introducción al lenguaje HTML. [Online] 2010. [Cited: diciembre 2, 2010.] http://www.deciencias.net/disenoweb/elaborardw/paginas/intro_html.htm.

39. **Sanchez, María A. Mendoza.** [Online] 2004. [Cited: noviembre 25, 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

40. **Alma, Enríquez Toledo.** [Online] [Cited: noviembre 26, 2010.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/MySQL.pdf.

41. **Desarrolloweb.** Qué es Oracle. [Online] 2002. [Cited: noviembre 28, 2010.] http://www.desarrolloweb.com/articulos/840.php.

42. **Alvarez, Sara.** Sistemas gestores de bases de datos. [Online] 2007. [Cited: noviembre 27, 2010.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

43. **Sunsite.** tutorial de Java. Arquitectura MVC. [Online] [Cited: diciembre 1, 2010.] http://sunsite.dcc.uchile.cl/java/docs/JavaTut/Apendice/arq_mvc.html.

44. **Software, Colectivo de Autores Asignatura Ingeniería de.** Entorno Virtual de Aprendizaje. [Online] 2011. [Cited: 2 15, 2011.] http://eva.uci.cu/mod/resource/view.php?id=35381.

45. **Universidad de San Carlos de Guatemala.** UML. [Online] 2010. http://es.scribd.com/doc/2080534/UML.

BIBLIOGRAFÍA CITADA

1. La Nación. [Online] 1999. http://wvw.nacion.com/ln_ee/especiales/siglo/siglo6/siglo3.html.

2. **GARZÁS PARRA, JAVIER / PIATTINI VELTHUIS, MARIO G.** *FABRICAS DEL SOFTWARE: EXPERIENCIAS, TECNOLOGÍAS Y ORGANIZACIÓN.* s.l. : RA-MA EDITORIAL, 2007.

3. **Pressman, R. S. (s.f.).** *Ingeniería de software, Un enfoque práctico.*

4. **Instituto Nacional de Estadística y Geografía de Mexico.** Ciberhabitat. [Online] http://www.ciberhabitat.gob.mx/biblioteca/extensiones/).

5. **EDUARDO ARAGÓN MONTES, JULIÁN CAMILO PÉREZ PARRA.** *SISTEMA PARA EL ALMACENAMIENTO VIRTUAL Y GESTION DE ARCHIVOS.* Cali, Colombia : s.n., 2009.

6. **Facultad de Ciencias Exactas y Naturales y Agrimesura.** Facultad de Ciencias Exactas y Naturales y Agrimesura. [Online] Universidad Nacional del Noreste. http://exa.exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/GESTAR02.htm.

7. **Felix Bachmann, Len Bass,Charles Buhman,Santiago Comella-D,Fred Long,John Robert,Robert Seacord.** *Technical Concepts of Component-Based Software Engineering,2nd Edition.* 2000.

8. **Caroline Batista Spencer Holanda, Clarissa Angélica de Almeida de Souza, Walcélio L. Melo.** *Um Repositório de Componentes para Web Dirigido.*

9. **Maribel Ariza Rojas, Juan carlos Molina García.** *Jerarquia y granularidad de componentes de software para Pymes en Bogota.* Bogotá DC : s.n., 2005.

10. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación.

11. **Departamento de Sistemas Informáticos y Computación Universidad de Valencia.** *Rational Unified Process.* Valencia : s.n.

12. **Beck, Kent.** *Extreme Programming Explained.* s.l. : Addison Wesley, 2000.

13. **Wells, Don.** XP Extreme Programming. *XP Extreme Programming.* [Online] 2009. http://www.extremeprogramming.org/.

14. **José H. Canós, Patricio Letelier yMª Carmen Penadés.** [Online] http://www.willydev.net/descargas/prev/TodoAgil.pdf.

15. **Orallo, Enrique Hernández.** *El Lenguaje Unificado de Modelado (UML).*

16. —. *El Lenguaje Unificado de Modelado (UML).*

17. **Scribd.** Scribd. [Online] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

18. **Innova, Grupo Soluciones.** Grupo Soluciones Innova. [Online] http://www.rational.com.ar/herramientas/roseenterprise.html.

19. **Jalon, Javier Garcia de.** *Aprenda Java como si estuviera en primero.* 2000.

20. **Pérez, Javier Eguíluz.** *Introducción a JavaScript.*

21. **Copyright 2011 Sencha.** Sencha. [Online] http://www.sencha.com/learn/Ext_Getting_Started.

22. **Pérez, Javier Eguíluz.** http://www.librosweb.es/. *http://www.librosweb.es/.* [Online] http://www.librosweb.es/ajax/capitulo1.html.

23. **Mehdi Achour, Friedhelm Betz, Antony Dovgal.** PHP. *PHP.* [Online] the PHP Documentation Group, 1997-2011. http://www.php.net/manual/es/intro-whatcando.php.

24. **Fabien Potencier, François Zaninotto.** *Symfony la guía definitiva.*

25. **Eguiluz, Javier.** symfony.es. [Online] http://www.symfony.es/2008/07/09/comparando-propel-doctrine-y-propelfinder/.

26. **Corporation, Oracle.** NETBEANS. [Online] 2011. http://netbeans.org/community/releases/69/.

27. **Corporation, Oracle.** NetBeans. [Online] 2011. http://netbeans.org/features/php/.

28. **Masip, David.** DesarrolloWeb.com. [Online] http://www.desarrolloweb.com/articulos/840.php.

29. **Oracle Corporation.** MySQL. [Online] http://dev.mysql.com/doc/refman/5.0/es/introduction.html.

30. **PostgreSQL-es.** PostgreSQL-es. [Online] http://www.postgresql-es.org/sobre_postgresql.

31. **Pressman, Roger S.** Ingeniería de Software. *Ingeniería de Software. Un enfoque práctico.* s.l. : Mc Graw Hill, 2001.

GLOSARIO DE TÉRMINOS

- ✓ **RAM**: la memoria de acceso aleatorio (en inglés: Random-Access Memory) es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados

- ✓ **TCP:** transmision Control Protocol (en español Protocolo de Control de Transmisión), es uno de los protocolos principales de Internet. Es utilizado para crear conexiones entre ordenadores para el envío de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron

- ✓ **RUP** o como se conoce en español Proceso Unificado de Desarrollo es una metodología que se utiliza en la ingeniería de software

- ✓ **LDAP: l**ightweight **d**irectory **a**ccess **p**rotocol (en español Protocolo Ligero de Acceso a Directorios) referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red

- ✓ **IDE:** entorno de Desarrollo Integrado, lugar donde los desarrolladores implementan los sistemas

- ✓ **UCI:** universidad de las ciencias informáticas

- ✓ **Plugin:** aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica