Componente Configuración en la versión 2.0 del Sistema de Gestión Universitaria

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

Autores:

Suyin Margarita Pérez Pérez.

Adrian Segura Tristá.

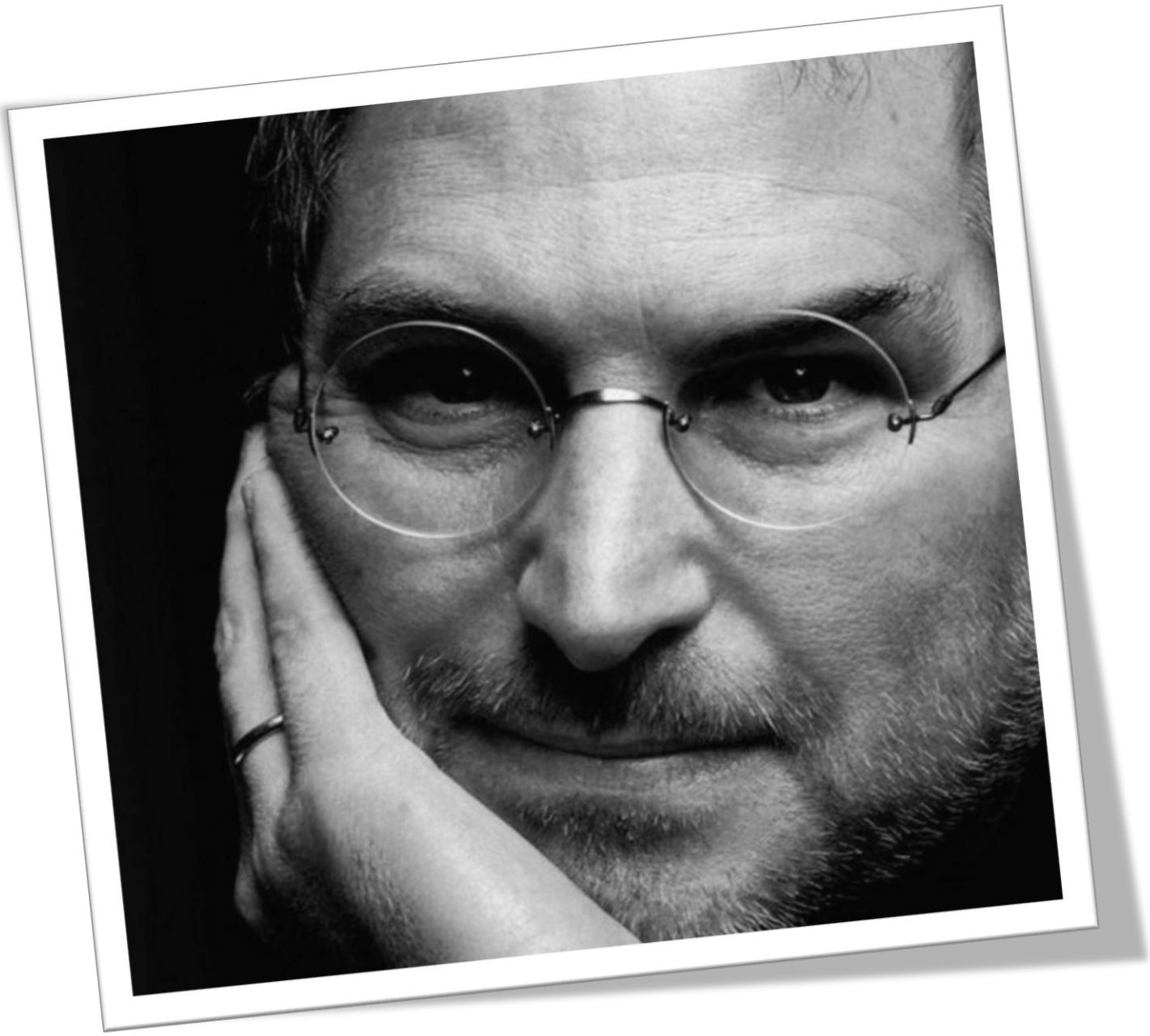
Tutores:

Ing. Yanio García Vidal.

Ing. Arlennys Susana Velázquez Hidalgo.

Ing. Nayilet Martín Soler.

La Habana, Cuba
Junio, 2015



"…la única manera de hacer un trabajo genial, es amar lo que haces…"

Steve Jobs.

Declaración de autoría

Declaramos ser los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente tesis a los ____ días del mes de _____ del año 2015.

Suyin Margarita Pérez Pérez

_____

Firma de la autor

Adrian Segura Tristá

_____

Firma del autor

Ing. Yanio García Vidal

_____

Firma del tutor

Ing. Arlennys S. Velázquez Hidalgo

_____

Firma de la tutor

Ing. Nayilet Martín Soler

_____

Firma de la tutor

Datos del contacto

- **Autores**

Nombre y apellidos: Suyin Margarita Pérez Pérez.
Dirección: General Peraza Sur #5ª Yaguajay, Sancti Spíritus.
E-mail: spperez@estudiantes.uci.cu

Nombre y apellidos: Adrian Segura Tristá.
Dirección: Calle 16 Edificio 5 (Comercio) Apto B-1 Reparto Aguilera, Las Tunas.
E-mail: asegura@estudiantes.uci.cu

- **Tutores**

Ing. Yanio García Vidal: Ingeniero en Ciencias Informáticas graduado de la Universidad de las Ciencias Informáticas, con 6 años de experiencia laboral, se desempeña como Especialista “B” en Ciencias Informáticas, ocupando el rol de asesor para el desarrollo, en el Departamento de Desarrollo en la Dirección de Informatización.

Ing. Arlennys S. Velázquez Hidalgo: Ingeniera en Ciencias Informáticas graduada de la Universidad de las Ciencias Informáticas, con 3 años de experiencia laboral, se desempeña como Especialista “B” en Ciencias Informáticas, ocupando el rol de analista de componente, en el Departamento de Desarrollo en la Dirección de Informatización.

Ing. Nayilet Martín Soler: Ingeniera en Ciencias Informáticas graduada de la Universidad de las Ciencias Informáticas, con 1 año de experiencia laboral, se desempeña como Recién Graduado en Adiestramiento, ocupando el rol de analista de componente, en el Departamento de Desarrollo en la Dirección de Informatización.

Agradecimientos

Hace 5 años llegué a esta universidad con un montón de sueños para realizar, hoy se cumplió uno de ellos, graduarme y convertirme en una profesional. Agradezco a todas las personas que de una forma u otra han hecho posible este sueño…

Especialmente a dos seres que me sería imposible hablar por separado, pues han dedicado toda su vida para darme una formación de la cual me siento orgullosa, dos seres que han sido mis pilares y ejemplo a seguir, a mis padres hermosos.

A mi gordi linda que lo es todo para mí, es el mejor regalo que la vida me ha dado.

A mis abuelitos que han sido la fortaleza para salir adelante ante los problemas que he tenido que enfrentar en la vida.

A mi tía Roxana que tanto quiero, por estar presente en todo momento, por sus palabras de firmeza cuando más lo necesité.

A alguien muy especial para mí, que a pesar de estar muy lejos hoy, siempre está a mi lado en mis pensamientos, a mi tía Martha.

A mi tío Tony y a Mileydi por estar siempre en los mejores y peores momentos de mi vida, por ayudarme a enfrentarlos, gracias por ese amor a nuestra familia y por ese apoyo incondicional.

A mi novio bello por soportarme todos los días, por estar ahí cuando más lo necesitaba, por ayudarme a enfrentar y darme sustento en cada momento difícil.

A mis suegros y mi cuñada pues a pesar que llegué a sus vidas hace poco tiempo, me han acogido como una familia, como una hija o una hermana.

A mis tutores, por guiarme, por tener tanta paciencia, por toda la ayuda prestada y el tiempo dedicado. Sin ustedes este sueño no se hubiera logrado hoy, a ustedes mis más gratos agradecimientos.

A mi compañero de tesis por ayudarme en los momentos de estudio, por su sincera amistad y el cariño que me han brindado en este duro camino.

A todos mis amigos por estar siempre en los momentos buenos y malos, por apoyarme en mis locuras y por cuidarme cuando más lo necesité.

A todos los integrantes del Sistema de Gestión Universitaria que aportaron su granito de arena para que el trabajo saliera a flote especialmente a Dianela, Yarelis, Yasmany, Alexander, Yasiel y Jorgito.

En fin quiero agradecer a cada persona que de una forma u otra han colaborado con este logro alcanzado, muchas gracias.

Suyin Margarita Pérez Pérez

Agradezco el presente trabajo de diploma:

En primer lugar a mi madre y mi padre por su ayuda y sacrificio durante toda la carrera.

A mi hermano y familia en general que tanto me aconsejan y siempre se ha preocupado por mi futuro.

A mi compañera de tesis Suyin por su apoyo y comprensión.

A mis tutores Arlennys, Nayilet y Yanio, no pude tener mejores tutores.

A todas las personas que de una forma u otra me aportaron en mi formación como profesional.

Adrian Segura Tristá

Resumen

El Sistema de Gestión Universitaria es una herramienta de apoyo que facilita la informatización de varios procesos que se desarrollan en la Universidad de las Ciencias Informáticas. Está compuesta por un conjunto de sistemas que garantizan que se cumpla este objetivo, donde el Núcleo, es la base para que exista un correcto funcionamiento en las operaciones que se realizan entre ellos. El Núcleo cuenta con un componente Configuración, que permite definir la división administrativa del país y la configuración de los nodos, esta última, agrupa determinados elementos relacionados entre sí, como: las aplicaciones y sus componentes. Las funciones que realiza el componente no son suficientes para obtener una correcta y ordenada configuración del sistema en general, debido a la gran dimensión con la que cuenta esta solución y lo cambiante que puede ser el entorno de la informática para satisfacer las necesidades del cliente. Por lo que se propone realizar una nueva versión del componente, que permita la generalización de las opciones que se establecen para el despliegue y soporte de los componentes que se van implementando e integrando a Sistema de Gestión Universitaria. Se podrán desarrollar configuraciones sin la necesidad de tener un elevado conocimiento de programación para gestionar los nodos del sistema, definir entidades de manera general, especificar la división administrativa que tendrán los países y realizar acciones automáticas mediante las tareas programadas. Se utiliza el entorno tecnológico definido por la Dirección de Informatización, para el desarrollo del componente Configuración del Sistema de Gestión Universitaria.

Palabras clave: *configuración, generalización, tareas programadas.*

Índice

Índice de figuras

Índice de tablas

Introducción

En la actualidad el conocimiento científico-técnico ha tenido una notable evolución producto a la necesidad de solucionar los problemas tanto individuales como corporativos existentes en todos los niveles de la sociedad. Uno de estos adelantos son los productos de *software* que hacen posible que la mayoría de las organizaciones puedan informatizar sus procesos para ganar competitividad y brindar servicios de calidad.

Para lograr que los productos de *software* cumplan con los objetivos específicos y sean adaptables a lo que requiere el cliente es necesario que exista una correcta configuración, debido que la misma trata de eliminar las diferentes incompatibilidades que puedan existir entre el producto creado por la empresa y lo que realmente necesita el usuario. Es importante señalar que las configuraciones pueden llevar eventualmente a errores, generalmente errores de escritura en la definición de los elementos de la configuración. Si se cuenta con una configuración defectuosa, el programa o aplicación funcionará de forma incorrecta y se solicita reconfigurar el sistema lo más rápido posible (1).

Cuba a pesar de ser un país subdesarrollado no se excluye de los adelantos generados en el campo de la informática, pues generalmente en las empresas que informatizan sus procesos son aplicadas dichas configuraciones para obtener mejores resultados en el desarrollo del *software* y así satisfacer las necesidades de la sociedad cubana. Una de las instituciones capaces de elevar el nivel de informatización en el país es la UCI[1], que cuenta con una herramienta de apoyo personalizada denominada SGU que constituye un sistema único e integrador de los procesos sustantivos de la comunidad universitaria (2).

El SGU está compuesto por un conjunto de sistemas que garantizan la informatización de estos procesos, donde el Núcleo del sistema, es la base para que exista un correcto funcionamiento en las operaciones que se realizan entre ellos. Cuenta con un componente Configuración, que permite definir la división administrativa del país y la configuración de los nodos, esta última, agrupa determinados elementos relacionados entre sí, como: las aplicaciones y sus componentes. Debido a la gran dimensión del SGU, las funciones que realiza el componente no son suficientes para obtener una correcta y ordenada configuración del sistema en general; pues se necesita que existan determinadas funcionalidades en dicho

[1] **UCI:** Universidad de las Ciencias Informáticas, cuya misión es formar profesionales comprometidos con su patria y altamente calificados en la rama de la Informática, así como producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática.

componente, para que se proporcionen las opciones que se establecen para el despliegue y soporte de los componentes que se van implementando e integrando a la aplicación.

Dentro de los inconvenientes que presenta dicho componente, se encuentra que solo personas con elevados conocimientos de programación pueden hacer uso del mismo, debido que en su mayoría las configuraciones se realizan manualmente en los archivos XML[2]. Cuando se definen los parámetros de configuración para cada uno de los sistemas, no se cuenta con una forma clara y común, pues es necesario establecerlos directamente en los archivos de configuración, trayendo consigo posibles errores por parte del responsable. No existe la visualización de los sistemas que están disponibles o los que están actualmente instalados, por tanto, cuando se realiza el despliegue, al usuario administrador se le hace difícil detectar los sistemas con los que cuentan.

No se especifican las dependencias funcionales que puedan existir cuando se realiza la acción de activar o desactivar un sistema, esto trae consigo posibles anomalías en el funcionamiento de los mismos; un ejemplo donde se evidencia es en el sistema Pregrado[3], que utiliza para su funcionamiento las estructuras administrativas definidas con anterioridad en el componente Estructura y Composición[4], por lo que se establece una relación funcional entre ambos.

Actualmente no brinda una funcionalidad donde se puedan gestionar las entidades de forma común para todos los componentes, ya que cada sistema gestiona las entidades según las necesidades del mismo de forma independiente, creando información repetida en la base de datos. En la funcionalidad División administrativa, no existe una correspondencia entre la relación que debe tener un país con sus respectivas provincias y municipios, excepto para Cuba; provocando que cuando ingrese un estudiante de otro país a la UCI existan errores a la hora de definir su país de procedencia.

En el sistema se efectúan acciones que se ejecutan en un tiempo específico y no se realizan de forma automática, muchas de estas son necesarias repetirlas cada un período de tiempo determinado, provocando que en ocasiones exista atraso por parte del responsable e incumplimiento de las mismas. Un

[2] **XML:** Por siglas en inglés *eXtensible Markup Language* ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible.
[3] **Pregrado:** Sistema del SGU, donde se gestiona la información de los estudiantes de la UCI.
[4] **Estructura y Composición:** Componente del Núcleo, que facilita la gestión de la información referente a toda la estructura administrativa y la jerarquía de una institución.

ejemplo de lo expuesto se evidencia cuando comienza un curso académico, que hay que incrementar el curso académico actual de forma manual.

Debido a los inconvenientes que presenta el componente con respecto al sistema, es preciso desarrollar nuevas funcionalidades para gestionar los nodos, definir las tareas programadas, las entidades y la división administrativa de los países en una nueva versión, para facilitar la organización estructural y la gestión de la información en el SGU.

A partir de la situación problemática existente se plantea el siguiente **problema de investigación**:

¿Cómo generalizar las configuraciones del Sistema de Gestión Universitaria a partir de las necesidades de cada uno de sus componentes?

Para dar solución al problema en cuestión se propone como **objeto de estudio** los sistemas informáticos que generalicen la configuración de sus componentes. Centrando su **campo de acción**: en la generalización de las configuraciones del Sistema de Gestión Universitaria en la Universidad de las Ciencias Informáticas.

En consecuencia se define como **objetivo general** desarrollar una solución informática que generalice la configuración en los componentes del Sistema de Gestión Universitaria de la Universidad de las Ciencias Informáticas, a partir del objetivo general planteado se derivan los siguientes **objetivos específicos**:

- Caracterizar los fundamentos teórico-metodológicos que permitan el desarrollo de la solución propuesta sobre la generalización de la configuración.
- Diseñar la propuesta de solución a partir del proceso de desarrollo de *software* utilizado.
- Desarrollar la versión 2.0 del componente de Configuración del Sistema de Gestión Universitaria.
- Evaluar la solución desarrollada a partir de las pruebas de *software*.

Para darle sustento a la investigación, se plantea la siguiente **idea a defender**: con el desarrollo de la versión 2.0 del componente Configuración, se contribuirá a mejorar la generalización de las configuraciones que se establecen para cada uno de los sistemas que componen el Sistema de Gestión Universitaria en la Universidad de las Ciencias Informáticas.

Para lograr dicho resultado se definieron las siguientes **tareas de investigación**:

- Caracterización del estado del arte de sistemas o tendencias existentes relacionadas con la generalización de la configuración de *software*, para lograr un mejor entendimiento del objeto de estudio y el campo de acción.
- Definición de los conceptos relacionados con el marco teórico de la investigación y las tecnologías necesarias para el desarrollo de la solución.
- Identificación de los requisitos según el proceso de desarrollo aplicado para elaborar la propuesta de solución.
- Validación de los requisitos del sistema aplicando las técnicas.
- Optimización del componente implementado previamente en el sistema, con el fin de incorporar funcionalidades que garanticen el cumplimiento del objetivo y así aumentar el rendimiento de la solución.
- Desarrollo de las pruebas de *software* diseñadas para medir la calidad de la solución.

Con el desarrollo de la investigación se pretende obtener como **resultado** el desarrollo de la versión 2.0 del componente Configuración, que contribuirá a generalizar las configuraciones que se establecen para cada uno de los sistemas que componen el Sistema de Gestión Universitaria en la Universidad de las Ciencias Informáticas.

Para responder al desarrollo de la investigación se emplearán los **métodos investigativos**: **teóricos** que se obtienen de la interacción directa entre el sujeto y el objeto de la investigación y **empíricos** que se basan en el trabajo con conceptos, categorías, leyes, teorías, entre otros. Dentro de los métodos empíricos se aplicará la **"Entrevista"** para obtener información referente a formar la base en la definición de los requisitos del componente; así como para precisar el problema a resolver, los problemas existentes y los servicios que brinda actualmente el componente. Se aplicará también la **"Revisión documental"**, que se pondrá de manifiesto en las revisiones a la bibliografía existente y a la hora de consultar la información en sitios de interés nacional e internacional, para apoyar la realización de las tareas definidas en la investigación.

De los **métodos teóricos** se usará el **"Analítico-Sintético"** cuando a partir de condiciones específicas se llegue a ideas generales del tema, posibilitando definir los conceptos que se manejaran en la investigación

y las principales características de las herramientas, lenguajes y metodología a utilizar en el desarrollo del componente y el **"Modelado"**, que permitirá elaborar el modelo conceptual y el mapa de navegación.
El documento está estructurado por los siguientes capítulos:

Capítulo1. Fundamentación teórica: incluye el estudio del estado del arte del tema tratado, donde se exponen los principales conceptos manipulados en el transcurso de la investigación. Se realiza la descripción y análisis de los sistemas existentes y una caracterización de las herramientas, técnicas, metodología y el proceso de desarrollo a utilizar en la construcción de la propuesta de solución.

Capítulo 2. Concepción de la propuesta de solución: realiza un análisis de la propuesta de solución que permita describir las reglas del negocio, construir el modelo conceptual y definir mediante las técnicas de obtención de requisitos, los requisitos funcionales y no funcionales. Además de describir la arquitectura, patrones de diseño y de base de datos, que se utilizan en la investigación, para poder elaborar el diagrama de Despliegue y el modelo de base de datos.

Capítulo 3. Construcción y validación de la propuesta de solución: describe lo relacionado con el desarrollo de la solución, teniendo en cuenta las principales necesidades del componente Configuración. Se aplican las técnicas para validar los requisitos definidos, se realizan pruebas a la solución propuesta a través de la estrategia de pruebas seleccionada y se valoran los resultados de la investigación.

El documento, incluye las conclusiones generales, bibliografías consultadas que sirven de soporte a la investigación, así como los anexos donde se encuentran los artefactos generados a lo largo del desarrollo de la investigación.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En la sociedad se lleva a cabo un proceso sólido y creciente del desarrollo de *software*, donde la correcta configuración del mismo juega un papel fundamental para satisfacer las necesidades del cliente. El presente capítulo tiene como objetivo tratar los conceptos y aspectos más significativos que se relacionan con la configuración de los sistemas informáticos, abordadas en diferentes fuentes bibliográficas para profundizar y ampliar el estudio del arte. Se realiza un análisis a diversos sistemas informáticos referentes al tema y se detallan las tecnologías, lenguajes y herramientas utilizadas en la implementación de la propuesta de solución.

1.2 Marco conceptual

Para lograr una mejor comprensión de la investigación, se abordan un conjunto de conceptos necesarios y están estrechamente relacionados con el dominio del problema.

1.2.1 Configuración

El concepto configuración en los últimos tiempos ocupa un espacio importante en las agendas de los investigadores, por lo que se considera relevante exponer aquellos que más tributan a la construcción teórica del componente a desarrollar. Según el Diccionario de la Real Academia Española (DRAE)[5] define que configurar es darle una determinada forma a algo, es un conjunto de los aparatos y programas que constituyen un sistema informático (3).

Existen dos tipos principales de configuración: la predeterminada y la personalizada. La configuración personalizada es aquella realizada por el usuario con un objetivo específico, haciendo posible transformar al elemento en cuestión en algo más útil y a la vez seguro; mientras que la predeterminada se define automáticamente, salvo algunos casos esta configuración no es recomendada y además de no seguir los intereses o necesidades del usuario también puede ser fácilmente alterada por agentes externos como virus o *hackers*[6] (1).

[5] **DRAE**: Es el diccionario normativo en idioma español o castellano editado y elaborado por la Real Academia Española (RAE).
[6] ***Hackers*:** Experto informático que tiene vastos conocimientos sobre las computadoras, redes, bases de datos, programas de computadoras y sobretodo de programación, su objetivo es poner la información al servicio de la comunidad en general.

Figura 1: Tipos de configuración

Con lo antes mencionado se puede concluir que la configuración es un conjunto de elementos que determinan la composición de un programa o sistema informático, son opciones que generalmente se establecen cuando se instala un *software*, pero también pueden implantarse en cualquier otro momento. Elementos que pueden ser alterados si así se considera necesario tanto para corregir un error, como para dar nuevas funciones o redefinir el elemento en diferentes modos.

1.2.2 Tarea programada

Una **tarea** es una obra, una labor, un quehacer o un trabajo que debe hacerse en un tiempo limitado (4). Mientras que **programar** es elaborar o formar programas para la resolución de problemas mediante ordenadores, también se puede decir que es preparar los datos previos indispensables para obtener la solución de un problema, así como crear programas con previa declaración de lo que se piensa hacer (5).

Al igual que el desarrollo de la humanidad está basado en obras o trabajos que requieren un esfuerzo y que deben hacerse en un determinado límite de tiempo, los sistemas informáticos están basados en la simultaneidad de tareas para que exista un funcionamiento correcto en los mismos. Estas tareas pueden realizarse de forma automáticas o no, siendo la primera opción más factible ya que el programa es capaz de ejecutar varias acciones o eventos predefinidos en determinadas fechas establecidos por el administrador, a esta opción se le denomina tarea programada.

Figura 2: Tareas programadas

1.2.3 Generalización

La Real Academia Española concreta que para generalizar es necesario abstraer o hacer lo que es común y esencial a muchas cosas, para formar un concepto general que comprenda todas (6).

La **generalización** implicará la extensión o propagación de algo, ya sea positivo o negativo, ejemplo de ello es la generalización de un conocimiento, como la informática que ha tenido una repercusión altamente positiva, gracias a que las personas logran mantenerse informadas y comunicadas permanentemente, en tanto, la generalización de una enfermedad implicará un escenario altamente negativo en la sociedad o comunidad en la cual se desarrolle (7).

Cuando se **generaliza** un componente de un programa informático, se hace con el objetivo de que sus funcionalidades puedan ser usadas por otros componentes de forma común para su beneficio, o sea se abstrae de detalles particulares para que exista una forma general para todos. La generalización facilita la construcción y la gestión de la información en un programa o una aplicación. Un ejemplo se muestra en el SGU, puesto que alguno de sus sistemas definen las entidades de forma independiente para la gestión de la información que maneja cada uno. Con la generalización de las entidades, se podrá tener una funcionalidad donde se encuentren todas las entidades definidas de forma común para todos los sistemas, con ello se podrá mejorar la calidad del trabajo y no existirá información repetida en las base de datos.

1.2.4 Dependencias funcionales

Una **dependencia** es una subordinación a un poder mayor, es la relación de origen o conexión (8), mientras que lo **funcional** es todo aquello en cuyo diseño u organización se ha atendido, sobre todo, a la facilidad, utilidad y comodidad de su empleo (9).

Se puede definir que las **dependencias funcionales** son aplicables en diferentes ramas de las ciencias, como la informática, pues para que exista un correcto funcionamiento de los sistemas informáticos estas son empleadas para lograr obtener conocimiento de los componentes que dependen de otros componentes. Lo antes expuesto se evidencia en el Núcleo del SGU, pues de él dependen para un adecuado funcionamiento los sistemas Cooperación, Pregrado, Postgrado, entre otros.

Figura 3: Dependencias funcionales del Núcleo

1.3 Análisis de las soluciones existentes

Con el objetivo de aumentar el nivel funcional del componente Configuración, se realizó el análisis de los sistemas nacionales e internacionales, que permitan apoyar el desarrollo de la investigación para lograr adaptar el SGU a las necesidades de los clientes.

1.3.1 XAVIA PACS-RIS

El XAVIA PACS-RIS es una aplicación diseñada para ofrecer al personal médico que labora en los Departamentos de Diagnóstico por Imágenes una gama de herramientas de propósito general. El objetivo fundamental de esta aplicación es la visualización y procesamiento de imágenes médicas, así como la edición de los informes que son emitidos. Con ello se facilita el acceso e intercambio de las imágenes desde cualquier punto de la institución de salud y la creación de las listas de trabajo para los equipos de adquisición de imágenes. El sistema está conformado por tres módulos principales: Visor, Bandeja de

casos y el Reportador, cada uno tiene una configuración específica para su funcionamiento, donde se utilizan XML y DLL[7] para representar estos parámetros de configuración. Cuando se procede a desplegar la aplicación de escritorio estas configuraciones son exportadas a un módulo configuración, con el cual se pueden realizar los cambios necesarios para los tres módulos mencionados anteriormente.

1.3.3 Inter-nos 2

Inter-nos 2 permite la gestión, publicación y transmisión de contenido multimedia sobre una red de datos a través de la web[8] mediante la tecnología *streaming*[9]. La plataforma constituye una herramienta de gran utilidad para emisoras de radio y televisión que deseen publicar sus producciones audiovisuales, colocar su señal en vivo en una red interna o transmitirla vía internet para hacerla accesible a un gran número de usuarios. Esta plataforma cuenta con un módulo configuración que contiene un panel de funcionalidades que permiten administrar las distintas acciones que se desarrollan en los sistemas. Dentro de las funcionalidades con las que cuenta se encuentra configurar los archivos de multimedia, donde se definen las extensiones de los formatos de los archivos, también se puede administrar las imágenes, las programaciones, las publicaciones en vivo así como configurar el módulo almacén. Para representar los parámetros de configuración se utiliza una tabla en la base de datos con dos columnas, donde se identifican la clave y el valor de la configuración para evitar que se combine la lógica del negocio con el acceso a datos.

1.3.4 Gestor de paquetes Synaptic de GNU/Linux

Synaptic es una interfaz gráfica aplicada a los sistemas operativos libres basada en paquetes. La aplicación permite instalar y gestionar *software* en el equipo, mediante un menú interactivo. Una aplicación individual puede incluso tener varios paquetes. Para evitar duplicidades, la mayoría de las aplicaciones reutiliza la funcionalidad de otras aplicaciones o bibliotecas. Las bibliotecas sólo proporcionan funciones a otras bibliotecas o aplicaciones y no son aplicaciones por sí mismas. De esta manera, la mayoría de los paquetes dependen de otros paquetes. El Gestor de Paquetes Synaptic resuelve las dependencias funcionales automáticamente, mostrando una advertencia de los demás paquetes que se necesitan instalar o desinstalar para utilizar el paquete deseado.

---

[7] **DLL:** Por sus siglas en inglés biblioteca de enlace dinámico (*'dynamic-link library'*), se refiere a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo.
[8] ***Web* o Red:** Se utiliza para nombrar a una red informática y en general a Internet.
[9] ***Streaming* o corriente contínua:** Es la distribución digital de multimedia a través de una red de computadoras de manera que el usuario consume el producto, generalmente archivo de video o audio, en paralelo mientras se descarga.

1.3.5 Drupal

Drupal es un marco de gestión de contenidos, libre, muy configurable, que sirve para administrar recursos web; es un sistema dinámico ya que en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno web.

El marco utiliza el concepto de nodo como un término genérico para cada pieza del contenido del sitio, o sea cualquier recurso que se ingrese al sistema pasa a ser un nodo. Este nuevo concepto permite estandarizar la información asignándoles las mismas características a distintos tipos de objetos y la posibilidad de tener toda la información centralizada y a la vez catalogada.

Drupal es una plataforma extremadamente flexible por lo que permite muchas opciones para configurar el aspecto del sitio, organizando la información en distintas funcionalidades agrupadas por módulos.

El sistema dispone de un Panel de Administración que dentro de sus categorías está el administrador de contenidos que maneja la configuración de cada módulo que se disponga; otras de las categorías es Temas de Administración que configura el aspecto visual, título del sitio y otros parámetros globales del sitio web.

1.3.6 Resultado del análisis de las soluciones existentes

Al realizar el estudio y análisis de los sistemas mencionados anteriormente, se puede concluir que a pesar de no cumplir todas las expectativas requeridas, aportaron amplios conocimientos acerca de las características, implementación y principios de funcionamiento, que sirve como base para guiar el desarrollo de la solución propuesta. Destacar que dichas aplicaciones adaptan las configuraciones a sus propias necesidades, por lo que los vuelve sistemas más competentes a la hora de satisfacer las expectativas de sus clientes. Inter-nos2 y Xavia Pacs-Ris aportaron un conjunto de conocimientos sobre cómo gestionar y manipular las configuraciones dinámicas. El Synaptic de Linux fue un apoyo para el estudio y manejo de las dependencias funcionales, contribuyendo a la consistencia entre los nodos y una correcta configuración sin la necesidad de escribir código directo en los archivos de configuración y Drupal contribuyó a la comprensión como ejemplo de CMS[10] para la gestión de módulos dinámicos.

[10] **CMS**: Por sus siglas en inglés significan Sistema Gestor de Contenido, es una herramienta que permite a un editor crear, clasificar y publicar cualquier tipo de información en una página web.

1.4 Metodología a utilizar para la implementación de la solución

Para estructurar, planear y controlar el proceso de desarrollo de la versión 2.0 del componente Configuración del SGU, el Departamento de Desarrollo de la DIN[11] de la UCI define la metodología DAC[12].

1.4.1 Proceso de desarrollo de *software* DAC

Según la necesidad que existe en la actualidad de hacer los *software* cada vez más rápidos, utilizando menos recursos y con mayor calidad, surge la metodología DAC cuyas siglas significan Desarrollo Ágil con Calidad, este proceso de *software* combina las buenas prácticas del nivel 2 de madurez de CMMI[13] y los conceptos vinculados al enfoque ágil, estableciéndose un marco de trabajo del proceso común, adaptable para los proyectos del departamento de desarrollo de la DIN, a partir de la integración de los modelos: lineal, desarrollo concurrente y programación extrema. Está enfocado a proyectos pequeños o proyectos grandes divididos en sub-proyectos que desarrollan *software* de gestión basado en componentes.

DAC plantea que el problema una vez identificado y definido debe ser descompuesto en problemas más pequeños y si es necesario, realizar con estos la misma operación. Cada sub-problema será resuelto mediante un componente y el problema resuelto será el *software* o producto final. En la siguiente figura se muestra el ciclo de vida que tendrá la propuesta de solución (10).

[11] **DIN**: Por sus siglas significan Dirección de Informatización, se encarga de informatizar varios procesos que se desarrollan en la Universidad de las Ciencias Informáticas.
[12] **DAC**: Por sus siglas Desarrollo Ágil con Calidad, metodología diseñada por el Departamento de Desarrollo de la Dirección de Informatización de la UCI.
[13] **CMMI:** *Capability Maturity Model Integration* por sus siglas en inglés - Integración de modelos de madurez de capacidades - es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de *software*.

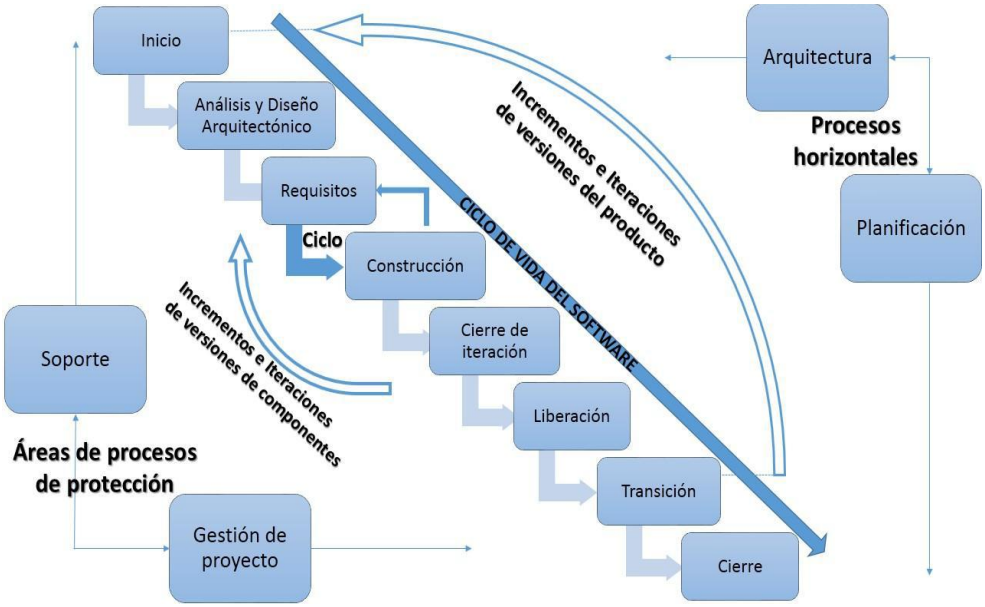


Figura 4: Modelo del proceso DAC *(10)*

1.5 Marco de trabajo a utilizar para el desarrollo de la solución

El marco de trabajo constituye un elemento importante a la hora de desarrollar una solución, pues es capaz de proveer una arquitectura sólida y más extensible para la implementación de los diferentes componentes en un sistema informático. Con la utilización del marco de trabajo se obtendrá un rápido desarrollo y una mayor calidad del producto, pues permite a los programadores, analistas y todo el conjunto de desarrolladores del *software* compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación que se desea construir.

1.5.1 GUUD 2.0

El Departamento de Desarrollo de la DIN define para el desarrollo de la propuesta de solución como marco de trabajo GUUD; que permite la creación de aplicaciones web escritas en PHP. Constituye un híbrido entre el marco de trabajo de PHP[14] CodeIgniter en su versión 1.7.3 y la librería de JavaScript, jQuery en su versión 1.9.2. Para la selección del CodeIgniter se tuvo en cuenta que es un producto de *software* libre, por lo que su uso es ilimitado; también se tuvo en cuenta algunas de sus principales características como: versatilidad, compatibilidad con la mayoría de los sistemas operativos, facilidad de instalación, flexibilidad, ligereza y documentación amplia y específica. Por otra parte se une la librería

[14] **PHP**: Es un lenguaje de programación de uso general, de código del lado del servidor, originalmente diseñado para el desarrollo web de contenido dinámico.

JQuery que ofrece un conjunto de funcionalidades, para el fácil manejo de JavaScript, pues de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. Algunas de sus principales características son:

- Permite la manipulación de elementos DOM[15].
- Realiza la manipulación de hojas de estilos CSS[16].
- Se pueden lograr efectos y animaciones.
- Funcionalidades para el uso de AJAX[17].

1.6 Lenguajes a utilizar para la implementación de la solución

Con el objetivo de no sólo solucionar los problemas existentes en el componente Configuración del SGU, sino además proponer nuevas funcionalidades que logren unificar calidad y eficiencia, se utilizaron los siguientes lenguajes, los cuales fueron definidos por las políticas de desarrollo de la Dirección de Informatización (DIN).

1.6.1 HTML 4

Descripción de su uso

HTML (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>), también puede describir hasta un cierto punto, la apariencia de un documento y es el lenguaje de publicación del *World Wide Web* (WWW[18]) (11).

[15] **DOM**: Por sus siglas en inglés significan Modelo de Objetos de Docemnto, es na interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, es un modelo estándar sobre cómo pueden combinarse dichos objetos y una interfaz estándar para acceder a ellos y manipularlos.

[16] **CSS**: Por sus siglas en inglés *Cascading Style Sheets* - Hojas de Estilo en Cascada - es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe cómo se va a mostrar un documento en pantalla, por impresora o por voz.

[17] **AJAX**: *Asynchronous JavaScript and XML* por sus siglas en inglés significan JavaScript Asíncrono y XML, es una técnica de desarrollo web para crear aplicaciones interactivas, compatible con la mayoría de los navegadores web.

[18] **WWW**: En informática, la *World Wide Web* (WWW) o Red informática mundial comúnmente conocida como la web, es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet. Con un navegador web, un usuario visualiza sitios web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia y navegar a través de esas páginas usando hiperenlaces.

1.6.2 PHP 5.6.7

Descripción de su uso

PHP (significado de "*PHP: Hypertext Preprocessor"*, traducido al español como Preprocesador de hipertexto) es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Con PHP no se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF (*Portable Document Format,* "Formato de Documento Portátil") y películas Flash (usando *libswf* y *Ming*) sobre la marcha. También puede presentar otros resultados, como XHTML[19] y archivos XML. Así mismo, puede autogenerar estos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla. Quizás la característica más potente y destacable es su soporte para una gran cantidad de base de datos. Escribir una interfaz vía web para una base de datos es una tarea simple con PHP (12).

1.6.3 JavaScript 1.8

Descripción de su uso

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, es utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y lenguaje C. Es orientado a objetos, al disponer de herencia, la cual se realiza siguiendo el paradigma de programación basada en prototipos y las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. JavaScript se ejecuta en el cliente al mismo tiempo que las sentencias van descargándose junto con el código HTML (13).

1.6.4 CSS 3

Descripción de su uso

Las hojas de estilo en cascada (*Cascading Style Sheets*, por sus siglas en inglés CSS) es un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML. El W3C[20] es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura del documento de su presentación (14).

[19] **XHTML:** Siglas del inglés *eXtensible HyperText Markup Language*. XHTML es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico, pero esto permite que posteriormente sea más fácil al hacer cambios o buscar errores entre otros.
[20] **W3C:** El Consorcio World Wide Web (W3C) es una comunidad internacional donde las organizaciones a tiempo completo y el público en general trabajan conjuntamente para desarrollar estándares Web, la misión del W3C es guiar la Web hacia su máximo potencial.

1.6.5 Lenguaje Unificado de Modelado (UML) 2.0

Descripción de su uso

Unified Modeling Language (UML, por sus siglas en inglés, traducido al español como Lenguaje Unificado de Modelado) es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas de *software*, así como para el modelado del negocio y otros sistemas. Un modelo UML está compuesto por tres clases de bloques de construcción; los elementos que son abstracciones de cosas reales o ficticias (objetos, acciones, etc.) de los cuales existen cuatro tipos de elementos, estructurales, ambientales, grupales y de anotación; las relaciones que son las que relacionan los elementos entre sí, ejemplo: dependencia, asociación, generalización y comprensión; y los diagramas que son las colecciones de elementos con sus relaciones (15).

1.6.6 XML 1.0 (*eXtensible Markup Lenguage*)

Descripción para su uso

Conjunto de reglas que por sus siglas significan Lenguaje Extensible de Marcas, se utiliza para definir las etiquetas semánticas que organizan un documento en diferentes partes. XML es un meta-lenguaje, que permite la definición de lenguajes concretos de representación de documentos. Este lenguaje es abierto, optimizado para su uso en la web y que permite describir el sentido o la semántica de los datos. El XML a diferencia del HTML, separa el contenido de la presentación (16).

1.6.7 SQL 2:2008 (*Structured Query Language*)

Es un lenguaje declarativo de alto nivel que por sus siglas significan lenguaje de consultas estructurado, un tipo de lenguaje vinculado con la gestión de bases de datos de carácter relacional, que permite la especificación de distintas clases de operaciones entre éstas. En esencia, el SQL es un lenguaje declarativo de alto nivel ya que, al manejar conjuntos de registros y no registros individuales, ofrece una elevada productividad en la codificación y en la orientación a objetos. Una sentencia de SQL puede resultar equivalente a más de un programa que emplee un lenguaje de bajo nivel. Se habla por tanto de un lenguaje normalizado que permite trabajar con cualquier tipo de lenguaje (PHP) en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL) (17).

1.7 Herramientas a utilizar para el desarrollo de la solución

Para realizar cada una de las tareas de la investigación se hizo uso de las herramientas informáticas, definidas por el departamento de desarrollo de la DIN logrando un mejor desempeño y rapidez para el desarrollo de solución.

1.7.1 Entorno de desarrollo integrado NetBeans IDE 8.0

Licencia del producto: Doble licencia; Licencia de desarrollo y distribución común, por sus siglas en inglés, *Common Development and Distribution License (CDDL)* y Licencia pública general de GNU versión 2 con excepción de rutas de clases, por sus siglas en inglés *GNU General Public License versión 2 with Classpath exception (GPL2).*

Descripción de su uso

NetBeans es un IDE escrito en Java, de código abierto (*OpenSource*), gratuito para desarrolladores de *software*. Ofrece todas las herramientas necesarias para crear aplicaciones profesionales, empresariales, web y móviles con el lenguaje Java, JavaFX, C/C ++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby, es fácil de instalar y se puede ejecutar tanto en *Windows*, como en Linux, Mac OS X y Solaris. NetBeans ofrece una versión del IDE hecho a la medida para el desarrollo de sitios web en PHP que comprenden una variedad de secuencias de comandos y lenguajes de marcado; se integra dinámicamente con HTML, JavaScript y CSS; cuando se configura el proyecto, se ejecuta en el servidor web incorporado a PHP; es totalmente compatible con el desarrollo iterativo, por lo que las pruebas proyectos PHP sigue los patrones clásicos familiares para los desarrolladores web (18).

1.7.2 PgAdmin III 1.14.0

Licencia del producto: Licencia BSD[21]

PgAdmin III es una herramienta de código abierto para la administración de base de datos PostgreSQL, cuenta con:

- Herramienta de consulta SQL[22].
- Interfaz administrativa gráfica.
- Editor de código procedural.

[21] **BSD**: Por sus siglas en inglés Distribución de *Software* Berkeley, es un sistema operativo derivado del sistema Unix nacido a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley.
[22] **SQL**: Por sus siglas en inglés significa Lenguaje de Consultas Estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

- Agente de planificación *SQL/shell/batch*.

Es capaz de gestionar versiones a partir de PostgreSQL 7.3 ejecutándose en cualquier plataforma. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar base de datos complejas (19).

1.7.3 Apache 2.4.7

Descripción de su uso

Apache es un servidor web altamente configurable, robusto y estable. Es un sistema de código abierto para plataformas *Windows*, Unix, Macintosh y otras que implementa el protocolo HTTP[23] (20).

Características principales:

- Es una tecnología gratuita de código fuente abierta.
- Es personalizable, la arquitectura modular de Apache permite construir un servidor hecho a la medida y posibilita la implementación de los últimos y nuevos protocolos.
- Es un servidor altamente configurable de diseño modular. Permite aumentar fácilmente su capacidad e instalar cualquier componente para cumplir una función específica.
- Permite personalizar la respuesta ante posibles errores y posibilita configurarlo para que ejecute un determinado *script* cuando ocurra un error en concreto. Permite la creación de ficheros de *log* facilitando, de este modo, el control de las acciones realizadas en el servidor.

1.7.5 Evolus Pencil 2.0.5

Descripción de su uso

Pencil es una herramienta para elaborar prototipos de código abierto, disponible para todas las plataformas, construida con el propósito de ofrecer una herramienta de creación de prototipos, libre y de código abierto de interfaz gráfica de usuario que sea fácilmente de instalar y utilizar para crear maquetas de plataformas de escritorio populares. Tiene entre sus características principales que los proyectos pueden ser exportados en los formatos HTML, PNG (*Portable Network Graphics*, “Gráficos de Red Portátiles”), documento de Word y PDF (*Portable Document Format,* “Formato de Documento Portátil”), además, permite la instalación de plantillas definidas por el usuario, las operaciones de dibujo estándar: alinear, escalar y rotar y la adición de los objetos externos (21).

[23] **HTTP**: Por sus siglas en inglés significan Protocolo de Transferencia de Hipertexto, es el protocolo usado en cada transacción de la *World Wide Web*.

1.7.6 Visual Paradigm para UML 8.0

Descripción de su uso

Visual Paradigm para UML 8.0 es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción y despliegue. Es multiplataforma, utiliza UML como lenguaje de modelado y cuenta con una versión libre para la comunidad (*Comunity Edition*), ayudando de una manera rápida a la construcción de aplicaciones de mayor calidad y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta también proporciona una mejor interfaz gráfica de usuario y una mayor base de datos de esquema de apoyo, permite generar la documentación del sistema en los formatos PDF, HTML y el formato de documentos de *Microsoft Word* y permite importar proyectos de otras herramientas de modelado como *Rational Rose, Erwin y Microsoft Visio*. Soporta la revisión ortográfica, brindando sugerencias para los idiomas: inglés, español, francés, alemán y portugués (22).

1.7.7 Apache Jmeter 2.8.1

Descripción de su uso

Apache JMeter, es una herramienta *Open Source* realizada en java que se utiliza para realizar pruebas de rendimiento y resistencia, generalmente a aplicaciones web. JMeter permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su “fuerza” y para analizar el rendimiento ante diferentes tipos de sobrecarga (23).

Las características principales de Apache JMeter son las siguientes:

- Permite cargar y realizar pruebas sobre distintos tipos de servidores.
- Multiplataforma: Unix (Solaris, Linux), *Windows* (98, NT, XP), *OpenVMS Alpha* 7.3.
- Testeo distribuido.
- Multihilo: Permite realizar pruebas de forma concurrente.
- Interfaz gráfica: Permite realizar las operaciones más rápido.
- Permite comprobar cómo se comportará una aplicación web ante multitud de usuarios y la velocidad de carga que tendrá (23).

1.7.8 PostgreSQL 9.4.1

Licencia del producto: Distribuida bajo la licencia de postgresql, licencia de código abierto, similar a las licencias BSD o MIT[24].

Descripción de su uso

Es un Sistema Gestor de Bases de Datos Objeto-Relacional (de sus siglas en inglés ORDBMS) basado en el proyecto Postgres, de la Universidad de Berkeley. Es una derivación libre de este proyecto. Debido a la licencia libre, PostgreSQL puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita para cualquier propósito, sea comercial, privado, o académico. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. (24).

1.8 Conclusiones parciales

Los conceptos asociados al objeto de estudio sirvieron de base y guía en el desarrollo de la investigación, adquiriendo y fomentando conocimientos necesarios, que contribuyeron en gran medida al logro de la propuesta de solución. Se realizó un análisis a sistemas que permitieran generalizar configuraciones a sus componentes, aportando nuevas ideas y características relevantes. Se utilizó la metodología DAC para estructurar, planear y guiar el proceso de desarrollo y se emplearon tecnologías, lenguajes y herramientas establecidas por la DIN que tuvieron como punto en común, la característica de ser código abierto.

[24] **MIT**: Es una licencias de *software* que ha empleado el Instituto Tecnológico de Massachusetts que permite su modificación.

Capítulo 2: Concepción de la propuesta de solución

2.1 Introducción

En el capítulo se presentan, las características de la propuesta de solución, iniciándose con los elementos relacionados al modelado conceptual, referente a la configuración de los componentes. Se definen las reglas de negocio y las técnicas de obtención de requisitos, permitiendo precisar los requisitos funcionales y no funcionales, se describen los primeros de forma detallada, para especificar las acciones que realizan en cada uno. Se describe la propuesta de solución, la arquitectura con el patrón arquitectónico utilizado, así como los patrones de diseño y de base de datos. Se representa el modelo de base de datos y el diagrama de despliegue.

2.2 Modelo conceptual

Un modelo conceptual se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos del *software*, es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción. En la construcción del modelo conceptual, se representa como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de *software* sino de la propia realidad física (25).

A continuación se muestra el modelo conceptual realizado al componente Configuración con el objetivo de descomponer el espacio del problema, en unidades comprensibles (conceptos):

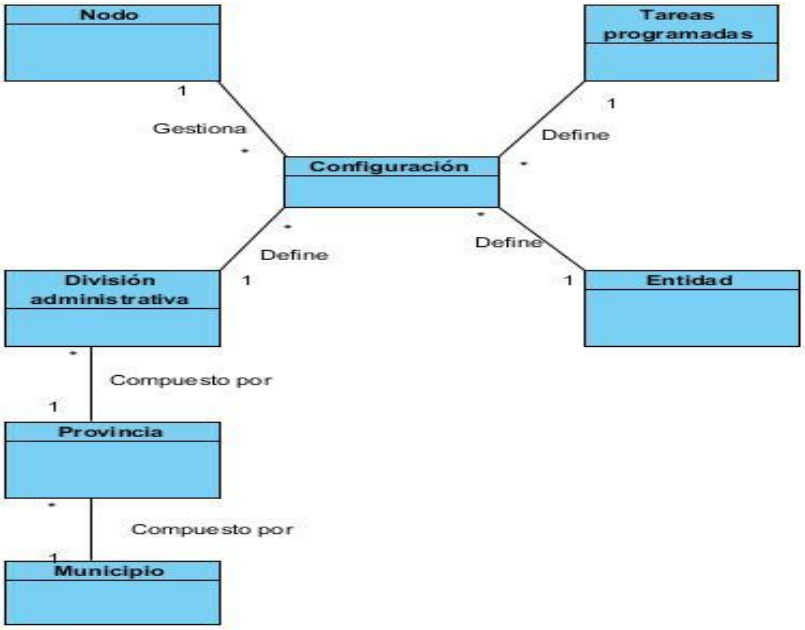


Figura 5: Modelo conceptual del componente Configuración

Para una mejor comprensión, a continuación se describen cada uno de los conceptos que intervienen en el modelo conceptual mostrado.

Configuración: Dar determinada forma a algo en específico, según las necesidades que existan.

Nodo: Unidad sobre la cual se construye un árbol, puede tener cero o más nodos hijos conectados a él. Cada nodo tiene sus propias características y pueden ser tanto módulos como aplicaciones.

Tarea programada: Acción que realiza el sistema de forma automática, en un período determinado de tiempo.

División administrativa: Fraccionamiento del país u ordenamiento interno del territorio, está compuesto por provincias y estas a su vez por municipios.

Provincia: División administrativa del territorio de un país, que forma parte de la estructura organizativa del territorio.

Municipio: División administrativa menor de un país.

Entidad: Organización o institución formada por un grupo de personas, bienes materiales y financieros.

2.3 Descripción de las reglas del negocio

Las reglas del negocio son un eslabón fundamental en la descripción de las políticas, normas, operaciones, definiciones y restricciones de una organización, pues hacen que exista un nivel detallado de

lo que debe hacer una organización. En el componente Configuración del SGU se definen un conjunto de reglas para el funcionamiento y organización del mismo:

- Solo existen dos tipos de nodos: aplicación y módulo; con la jerarquía correspondiente.
- Para crear un nodo que sea de tipo módulo, debe definirse el nodo padre.
- La etiqueta del nodo es el nombre que tendrá para representarlo al usuario.
- Existen dos tipos de parámetros de configuración para los nodos: los comunes y los específicos.
- Los parámetros comunes son asignados por defecto según el tipo de nodo.
- Para crear una entidad debe estar previamente creada al menos una categoría de entidad.
- Para crear una categoría de entidad debe estar previamente creado al menos un atributo de entidad.
- Para crear los atributos de entidad existen cuatro tipos de componente:
    - Texto corto: Es un componente *Input*.
    - Texto largo: Es un componente *Textarea*.
    - Selección inclusiva: Se utiliza para seleccionar más de un valor, cuando son hasta 5 posibles valores se muestra con *Check Box*, en caso contrario con *Multiselect*.
    - Selección exclusiva: Se utiliza para seleccionar solamente un valor de la lista, cuando son hasta 5 valores se muestran *Radio button*, en caso contrario *Combo Box*.
- La División administrativa del país está compuesta por provincias y municipios, donde estos últimos se subordinan a las provincias.
- La definición de las tareas programadas se pueden realizar por cuatro vías o acciones: Insertar, Modificar, Eliminar o A partir de una consulta.
- Los períodos de tiempo que se pueden definir en una tarea programada son: cada “n” minutos, cada una hora, cada un día, cada una semana, cada un mes, cada un año.
- Para definir las operaciones que se realizan en una tarea programada depende del tipo de dato de la columna:
    - Tipo cadena: igual, contiene, diferente.
    - Tipo entero: diferente, mayor, mayor que, menor, menor que.
    - Tipo booleano: igual, diferente.

- Para definir la condición de la construcción de una consulta en la base de datos para las tareas programadas por las vías –Modificar o Eliminar- los signos de agrupación son: paréntesis “()”, conectores “Y, O” y el operador unitario “Negación”.
- Para definir la construcción de una consulta en la base de datos para las tareas programadas por vía -A partir de una consulta-; tendrá la misma sintaxis que el *postgresql* a partir de la condición.

2.4 Requisitos del sistema

La captura de requisitos es un paso fundamental para saber exactamente lo que debe hacer y las cualidades o propiedades que deben tener los componentes definidos. Para el desarrollo de la propuesta de solución se identificaron requisitos funcionales y no funcionales, siendo los primeros guiados mediante técnicas de obtención que permiten establecer una comunicación más específica entre el cliente y el equipo de desarrollo.

2.4.1 Definición de las técnicas de obtención de requisitos

Existen varias técnicas que permiten establecer comunicación clara y eficiente entre los interesados en el producto y el equipo de trabajo, pues facilitan indagar en lo que los usuarios realmente necesitan. Estas técnicas se centran principalmente en el descubrimiento de los requisitos del sistema. Por lo que es necesario trabajar con el cliente debido a que sus requisitos abarcan las tareas que se necesitan para llevar a cabo el desarrollo del sistema. En la identificación de los requisitos de la propuesta de solución se utilizaron las siguientes técnicas:

- **Entrevista:** Se utilizó con el fin de obtener diversas opiniones y recomendaciones en un intercambio mediante preguntas con los especialistas del SGU, para esclarecer con precisión el funcionamiento del desarrollo del componente Configuración. Ver en el anexo #1.
- **Prototipado:** Con el empleo de esta técnica se realizaron las vistas del sistema, mediante la construcción de prototipos. Provocando un conjunto de ideas entre los clientes y el equipo de desarrollo para un mejor entendimiento de la solución. Además permitió formalizar la aceptación previa de los requisitos.

2.4.2 Requisitos funcionales

Los requisitos funcionales del sistema describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de *software* que se desarrolle, de los posibles usuarios y del enfoque general tomado por la

organización al redactar el requisito (26). Para guiar el establecimiento de la complejidad y la prioridad de los requisitos funcionales del componente Configuración, se utilizó el documento del expediente de proyecto del Núcleo, denominado Evaluación de requisitos del Sistema de Gestión Universitaria. En la siguiente tabla se muestran los requisitos funcionales, para un total de 30, de los cuales se identificó que 8 tienen una complejidad alta, 9 media y 13 baja, así como también se estableció la prioridad para cada uno, clasificando 9 de prioridad alta, 15 media y 6 baja.

Tabla 1: Requisitos funcionales del componente Configuración

No.	Requisitos	Complejidad	Prioridad
RF1	Mostrar atributos de entidad.	Baja	Media
RF2	Crear atributo de entidad.	Media	Media
RF3	Modificar atributo de entidad.	Media	Media
RF4	Ver detalles de atributo de entidad.	Baja	Baja
RF5	Mostrar categorías de entidad.	Baja	Media
RF6	Crear categorías de entidad.	Media	Media
RF7	Modificar categorías de entidad.	Media	Media
RF8	Ver detalles de categoría de entidad.	Baja	Baja
RF9	Mostrar entidades.	Baja	Media
RF10	Crear entidad.	Alta	Alta
RF11	Modificar entidad.	Alta	Alta
RF12	Ver detalles de entidad.	Baja	Baja
RF13	Mostrar nodos.	Baja	Media
RF14	Activar o Desactivar un nodo.	Media	Media
RF15	Dependencias de nodo.	Media	Media
RF16	Mostrar nodos activos.	Baja	Media
RF17	Crear nodo activo.	Alta	Alta
RF18	Modificar nodo activo.	Alta	Alta
RF19	Dependencia funcional de nodo activo.	Media	Media
RF20	Parámetros de configuración nodo activo.	Media	Alta
RF21	Ver detalles de nodo activo.	Baja	Baja
RF22	Mostrar tareas programadas.	Media	Media
RF23	Crear tarea programada.	Alta	Alta
RF24	Modificar tarea programada.	Alta	Alta
RF25	Eliminar tarea programada.	Baja	Media
RF26	Ver detalles de tarea programada.	Baja	Baja
RF27	Mostrar divisiones administrativas.	Baja	Media
RF28	Crear división administrativa.	Alta	Alta
RF29	Modificar división administrativa.	Alta	Alta
RF30	Ver detalles de división administrativa.	Baja	Baja

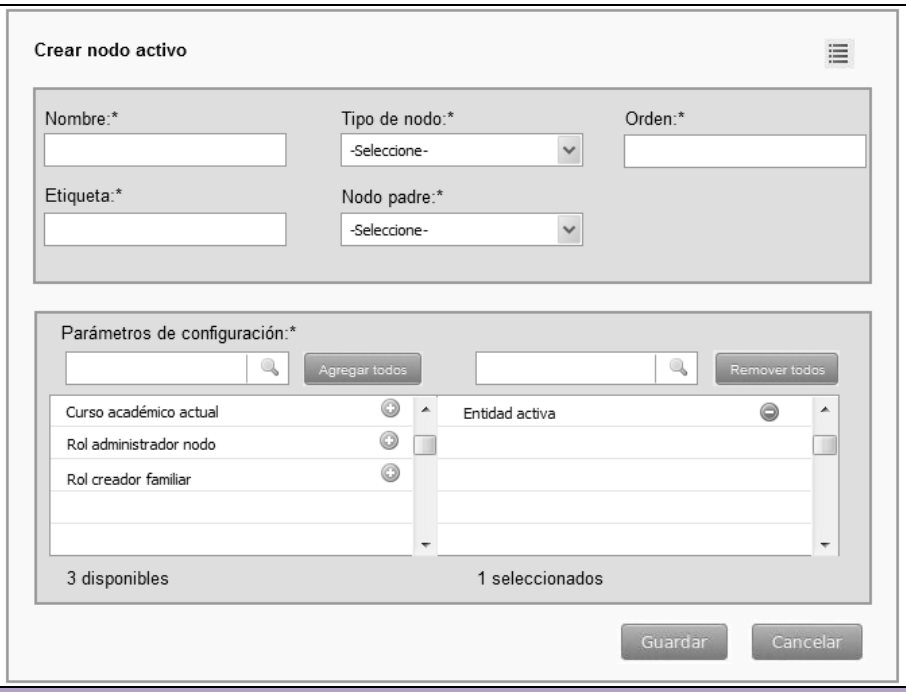
2.4.3 Descripción de requisitos

Con el objetivo de evitar errores y representar los requisitos funcionales de una manera consistente y detallada para el desarrollo del componente, se hace una descripción de requisitos. Para ello se

establecen la prioridad, la complejidad, una breve descripción de las acciones principales y el prototipo de la interfaz para el desarrollo de la funcionalidad para su posterior implementación. A continuación se muestra en la tabla 2 la descripción del requisito Crear nodo activo de la funcionalidad Nodo activo del componente Configuración, para consultar las restantes remitirse al anexo #2.

Tabla 2: Descripción de requisitos Crear nodo activo

Código	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RFConf17	Crear nodo activo	• El requisito permite crear un nodo activo. • El administrador selecciona del componente Configuración del Sistema de Gestión Universitaria (Núcleo), luego en el menú lateral la agrupación funcional General, funcionalidad Nodos activos. En el área de iconos flotantes selecciona la opción Crear nodo activo. • Se muestran los datos a llenar: Nombre, Tipo de nodo, Orden, Etiqueta, Nodo padre y Parámetros de configuración. • Además de la opción: Listar en el área de iconos flotantes.	Alta	Alta
Prototipo				



Botones e hipervínculos

Nº	Nombre	Descripción de flujo básico	Descripción de flujo alterno
1	Botón Guardar	Verificar que los campos obligatorios estén introducidos. Verificar que los datos introducidos tengan el formato correcto. Consultar si existe un nodo activo con el mismo nombre. El sistema no permitirá escribir caracteres inválidos. El sistema no permitirá escribir más de los caracteres permitidos en un campo. Insertar un nodo activo**.** Muestra el mensaje: *"El elemento ha sido creado satisfactoriamente".* Se mantiene en la misma interfaz del crear.	Si faltan campos obligatorios sin llenar muestra en rojo el mensaje: "Campo requerido" sobre el campo obligatorio. Si existe un nodo activo con el mismo nombre muestra mensaje: "*El elemento ya existe*". Si no introduce la cantidad mínima necesaria de caracteres en un campo muestra en rojo el mensaje: *"Entre al menos (cantidad de caracteres requeridos) caracteres"* sobre el campo requerido. Si introduce más palabras de las requeridas en un campo muestra el mensaje en rojo: *"Ha excedido el número de letras permitidas para una palabra" sobre el campo requerido.* Si introduce caracteres con formato incorrecto se muestra el mensaje: *"Entre solo letras, números y espacio o guión bajo entre palabras"* sobre el campo requerido.
2	Botón Cancelar	Muestra un mensaje de confirmación: "*¿Está seguro de realizar la acción?*". Si Acepta regresa a la interfaz del Mostrar nodos activos.	Si Cancela se queda en la misma interfaz.

3	Icono Listar	Se muestra en el área de iconos flotantes y regresa al listado nodos activos.	

2.4.4 Requisitos no funcionales

Los requisitos no funcionales, como su nombre sugiere, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la usabilidad, el tiempo de respuesta y la capacidad de almacenamiento (26). En la tabla 3 se muestran los requisitos no funcionales que deben cumplirse para el componente Configuración del SGU.

Tabla 3: Requisitos no funcionales del componente Configuración

Usabilidad	
RNF 1	El componente debe presentar un menú lateral y una barra de iconos flotantes que permitan el acceso rápido a la información por parte de los usuarios.
RNF 2	Las vistas del componente deben indicar en cada momento la acción que se está realizando, así como los íconos deben estar representados por una imagen acorde a la acción que se realiza.
RNF 3	Sólo se mostrarán a los usuarios aquellas acciones o informaciones del menú lateral a las que por su responsabilidad o rol dentro del negocio necesiten acceder.
Confiabilidad	
RNF 4	El tratamiento de las excepciones permitirá un seguimiento hasta guardar información, acerca del lugar dónde se produjo el error y de los parámetros utilizados en el sistema que lo provocaron. Cuando ocurre una excepción el sistema mostrará un mensaje explicativo del error ocurrido.
RNF 5	El componente puede permanecer inactivo durante 10 minutos. Al cumplirse este término se cerrará la sesión teniendo que autenticarse el usuario nuevamente.
Eficiencia	
RNF 6	El componente deberá tener por cada transacción un tiempo de respuesta promedio de 1,5 segundos y un máximo de 5 segundos.
RNF 7	El componente soportará la conexión simultánea de todos los posibles usuarios, con un promedio de 1000 y un máximo de 3000.
Soporte	
RNF 8	El componente cumplirá con las normas de codificación, conversiones para nomenclatura, bibliotecas de clase definidas para el Sistema de Gestión Universitaria en el documento SGU-NUC-010217_EC.
RNF9	El componente contará con toda la documentación definida en el expediente de proyecto asociada a su proceso de desarrollo para las actividades de soporte.
Restricciones de diseño	
RNF 10	El componente deberá ser desarrollado en su totalidad con tecnologías y componentes de código abierto.
RNF 11	El componente estará desarrollado con las herramientas definidas para el Sistema de Gestión Universitaria en el documento “Arquitectura de Software Vista de tecnología e infraestructura”, que se encuentra en el expediente de proyecto del Núcleo en la dirección: https://din-svn.uci.cu/svn/documentacion/Dpto_GU/Gestion_Universitaria/Nucleo/Ed.3.4/1ingenieria/1.2arquitectura_y_disenno/arquitectura_de_software/.
RNF 12	El componente cumplirá con la arquitectura de información definida para el Sistema de Gestión Universitaria en el documento de “Arquitectura de Software Vista de Presentación DAC”, que se encuentra en el expediente de proyecto del Núcleo en la dirección: https://din-svn.uci.cu/svn/documentacion/Dpto_GU/Gestion_Universitaria/Nucleo/Ed.3.4/1ingenieria/1.2arquitectura

	_y_disenno/arquitectura_de_software/.
Interfaz	
RNF 13	La interfaz de usuario se guiará por la vista de arquitectura de presentación definidas en el documento "Pautas de diseño XAUCE-SGU".
Software	
RNF 14	Para el despliegue del componente se debe contar en el servidor de aplicaciones web con: PHP v 5.6 con las librerías php5-ldap, php5-gd, php5-mcrypt, php5-pgsql, php5-xsl, php5-*openssl* y Apache 2.2 con el módulo *rewrite* activado.
RNF 15	Para el despliegue del componente se debe contar en el servidor de bases de datos con PostgreSQL 9.4, bajo el sistema operativo CentOS 6.6 o superior.
RNF 16	Para el uso del componente se requiere una PC cliente con cualquier sistema operativo, que se pueda instalar el navegador web Mozilla Firefox 26.0 o superior para el uso de la aplicación web.
Hardware	
RNF 17	Para la ejecución del componente se requiere que la PC cliente tenga los siguientes componentes de *hardware*: Pentium 4 o superior, 512 MB RAM y 200 MB disco duro disponible como mínimo.
RNF 18	La comunicación entre el cliente y el servidor de aplicaciones se realiza a través del protocolo HTTPS.

2.5 Descripción de la propuesta de solución

La propuesta de solución tiene entre sus principales objetivos garantizar una correcta generalización de las configuraciones del sistema tanto global, como las que son usadas por más de una línea de proceso.

El componente Configuración cuenta con dos agrupaciones funcionales: General y Entidades. General cuenta con un conjunto de funcionalidades como es Nodo, donde su principal función es visualizar la jerarquía existente entre sistemas, permitiendo realizar acciones de activar o desactivar las aplicaciones y los módulos que la integran, cuando se realizan ambas acciones se tendrán en cuenta las dependencias funcionales que puedan existir entre los mismos. Otra funcionalidad es Nodos activos, que le permite al usuario interactuar con el árbol de los nodos previamente activados, brindando la posibilidad de registrar nuevos nodos, definir el tipo de nodo que puede ser tanto sistema (o aplicación) como componente(o módulo), los parámetros de configuración y el orden que tendrá en el sistema, también permite tener un conocimiento de la relación de dependencias o dependientes que pueda tener con otros nodos. Las tareas programadas permiten realizar acciones de forma automáticas en el sistema en un tiempo definido por el administrador, en ellas se establece la base de datos en la cual tendrá lugar la tarea, la recursividad estableciendo el período de tiempo para ejecutarse y la acción a realizar ya sea -A partir de una consulta-, -Insertar-, -Modificar- o –Eliminar-; para finalmente definir la consulta en el sistema o el componente seleccionado. Otra de las funcionalidades que presenta es División administrativa, donde se registra el tipo de área administrativa: Provincia o Municipio, en esta última se define la provincia a la cual está subordinada.

En la agrupación funcional Entidad se definen las entidades, para ello se concreta previamente la categoría a la que pertenece la entidad; a una categoría de entidad se le asocian atributos de entidades estáticos y dinámicos, a estos últimos se le define el tipo de componente que tendrá en el sistema. A continuación se muestra en la figura 6 el mapa de navegación para poder hacer uso de la propuesta de solución.

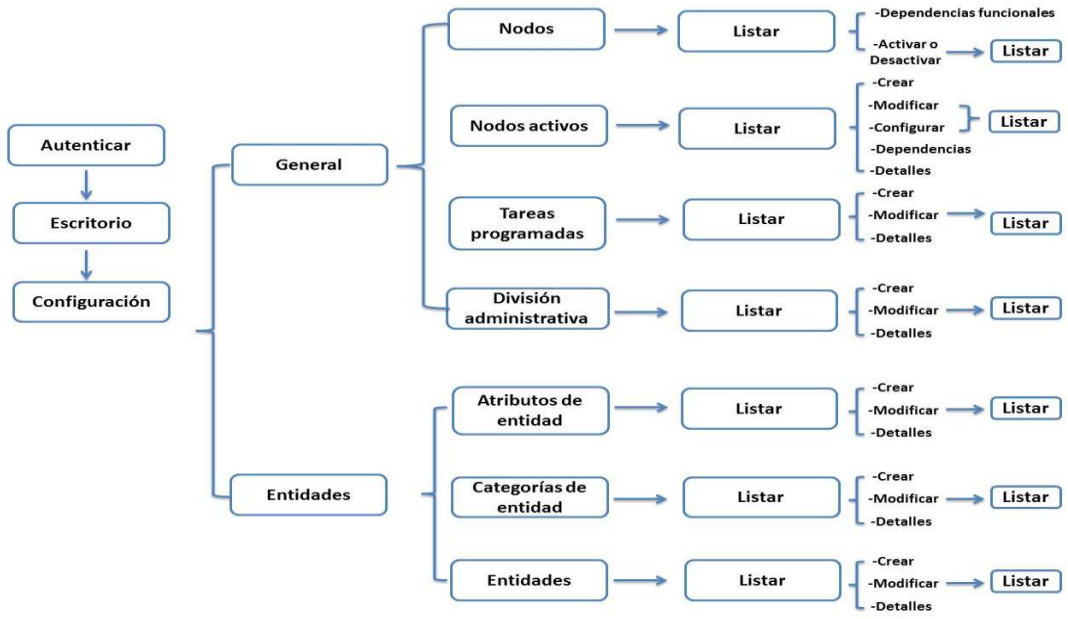


Figura 6: Mapa de navegación del componente Configuración

2.6 Descripción de la arquitectura

La arquitectura de un *software* es la estructura u organización de un sistema que incluye los componentes de este, las propiedades visibles externas de esos componentes y las relaciones que existen entre ellos (27).

La arquitectura utilizada para la propuesta de solución es Cliente-Servidor, que consiste en un cliente que envía un mensaje solicitando un determinado servicio a un servidor (hace una petición) y este envía uno o varios mensajes con la respuesta (provee el servicio). Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción Cliente-Servidor es el soporte de la mayor parte de la comunicación por redes. Ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos (28).

En la siguiente figura se muestra la representación de la arquitectura:

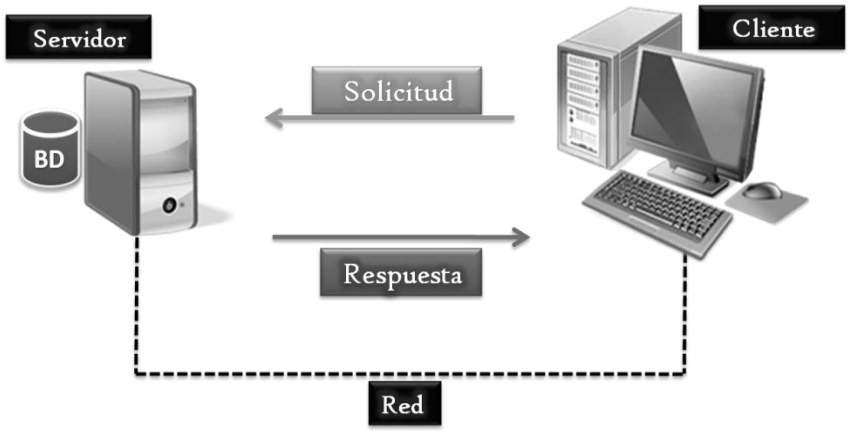


Figura 7: Arquitectura Cliente-Servidor

Cliente: Programa ejecutable que participa activamente en el establecimiento de las conexiones. Envía una petición al servidor y se queda esperando por una respuesta. Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo (28).

Servidor: Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. Al ser posible implantarlo como aplicaciones de programas, puede ejecutarse en cualquier sistema donde exista TCP/IP[25] y junto con otros programas de aplicación. El servidor empieza su ejecución antes de comenzar la interacción con el cliente. Su tiempo de vida o de interacción es "interminable" (28).

2.6.1 Patrones arquitectónicos

Los patrones arquitectónicos son patrones del *software*, que se encargan de definir la estructura de un sistema. Estos a su vez se componen de subsistemas con sus responsabilidades, también poseen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño. Un patrón arquitectónico se enfoca en dar solución a un problema en específico y abarca solo

[25] **TCP/IP**: Es una descripción de un conjunto de guías generales de diseño e implementación de protocolos de red específicos para permitir que un equipo pueda comunicarse en una red.

parte de la arquitectura. Cuando un patrón de este tipo brinda una imagen general de un sistema, él no es una arquitectura como tal. Es por esto que un patrón arquitectónico es un concepto que captura elementos esenciales de una arquitectura de *software* (29).

Como parte de la conformación de la arquitectura se encuentra el patrón arquitectónico Modelo-Vista-Controlador (MVC) que se encarga de separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Este patrón se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página (29).

Los elementos de este patrón son:

- **Modelo:** Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- **Vista:** Muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- **Controlador:** Reciben las entradas, usualmente como eventos que codifican los movimientos, pulsación de botones del ratón o pulsaciones de teclas, entre otros. Los eventos son traducidos a solicitudes de servicio ("*service requests*" en el texto original) para el modelo o la vista. El usuario interactúa con el sistema a través de los controladores (30).

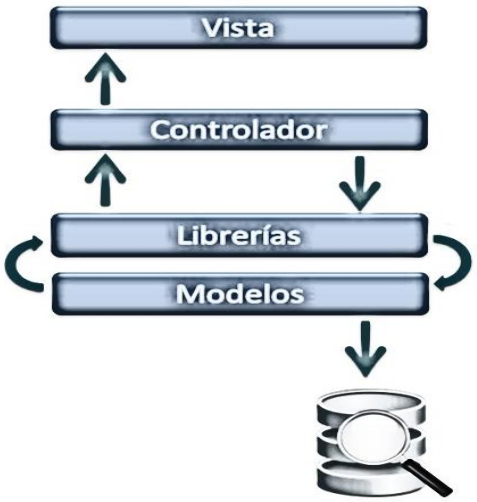


Figura 8: Patrón arquitectónico Modelo-Vista-Controlador

Implementación del MVC que realiza GUUD

En la figura 9 se muestra la representación del patrón MVC evidenciado en el marco de trabajo GUUD el cual se describe a continuación:

El marco de trabajo GUUD cuenta con un controlador frontal que cuando un cliente realiza una petición HTTP, este analiza y determina cuál controlador de aplicación (controlador de una determinada funcionalidad de un módulo) debe ser cargado para atender la petición realizada. Cada controlador de aplicación tiene asociada una o varias librerías y esta a su vez tiene asociado uno o varias modelos. Las librerías son responsables de procesar los datos e implementar la lógica del negocio y la clase modelo es encargada del acceso a los datos.

Luego de ser cargado el controlador de aplicaciones, este analiza la petición para determinar si necesita interactuar con la base de datos o solo cargar una vista determinada. En caso que necesite interactuar con la base de datos envía los datos recibidos a las librerías y estas cargan las modelos necesarias para obtener, registrar o actualizar en la base de datos la información solicitada. Cuando los datos son obtenidos, se retornan al controlador de aplicación en un proceso inverso al anterior. Posteriormente, el controlador carga estos datos a archivos escritos en HTML los cuales pueden incluir llamadas a archivos escritos en JavaScript para manejar dinámicamente su contenido. Finalmente, el resultado obtenido de todo este proceso es enviado al navegador web como respuesta a la petición inicial.

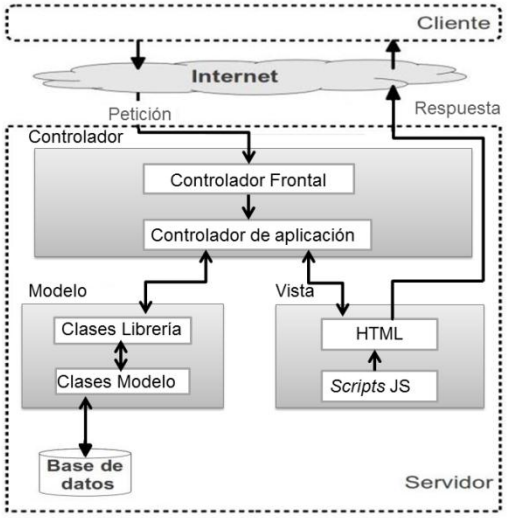


Figura 9: Funcionamiento del patrón MVC en el marco de trabajo GUUD

2.7 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. El mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Estos modelos que se presentan como parejas de problema/solución con un nombre, codifican buenos principios y sugerencias relacionados con la asignación de responsabilidades, basados en la recopilación del conocimiento de los expertos en desarrollo de *software* (31).

Dentro de los más conocidos se encuentran los patrones de Asignación de Responsabilidades (GRASP) y los patrones de la Banda de los Cuatro (GoF).

2.7.1 Patrones GRASP

Los Patrones Generales de Asignación de Responsabilidades de *software* (GRASP, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar *software* orientado a objetos (32). Los mismos no introducen ideas novedosas, sino que son la codificación de los principios básicos más usados. Los patrones GRASP están compuestos por: experto, creador, bajo acoplamiento, alta cohesión y controlador. A continuación se muestran los patrones utilizados en el desarrollo de la propuesta de solución:

- **Creador:** Este patrón se utiliza en la mayoría de las clases del sistema para permitir que una clase creadora se conecte con los objetos producidos en cualquier evento. En la figura 10 se pone de manifiesto en la clase *Loader*, pues esta es la responsable de la creación (o instanciación) de nuevos objetos o clases.

```
public function __construct() {
    parent::__construct();
    $this->load->library('tareas_programadas_lib');
    $this->load->library('nodo_activo_lib');
}

public function obtenerTareasProgramadas() {
    $this->load->helper('grid');
    $post_vars = $this->input->all_post();
    $datos_buscador = array();
    if (isset($post_vars['cadena'])) {
        $datos_buscador = json_decode($post_vars['cadena']);
    }
    echo grid_json($this->tareas_programadas_lib, 'obtenerCantidadTareasProgramadas', 'obtenerTareasProgramadas'
}
```

Figura 10: Patrón Creador

- **Alta cohesión:** Este patrón viene implementado en el propio marco de trabajo GUUD. Las librerías reflejan la alta cohesión pues la información que almacena cada una de estas clases es coherente y está (en la medida de lo posible) relacionada con ella. La librería tarea_programada_lib.php es ella y solo ella la única encargada de la lógica de esa parte del negocio.
- **Bajo acoplamiento:** Se evidencia en el hecho de que una clase no depende de muchas otras, lo cual potencia la reutilización y disminuye la dependencia entre estas. Las librerías reflejan el bajo acoplamiento. Un ejemplo es la librería tarea_programada_lib.php en su implementación solo contiene la lógica de su negocio y el acceso a datos, esta clase no tiene instancia de más ninguna clase que refleje dependencias.

- **Controlador:** Permite asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Este patrón se evidencia en las clases controladoras que se encargan de obtener datos, enviarlos a las librerías y a las vistas. En la figura 11, el menú de la izquierda define cuales son las controladoras de la propuesta de solución, dichas controladoras son las clases que recibirán las peticiones realizadas a la aplicación.

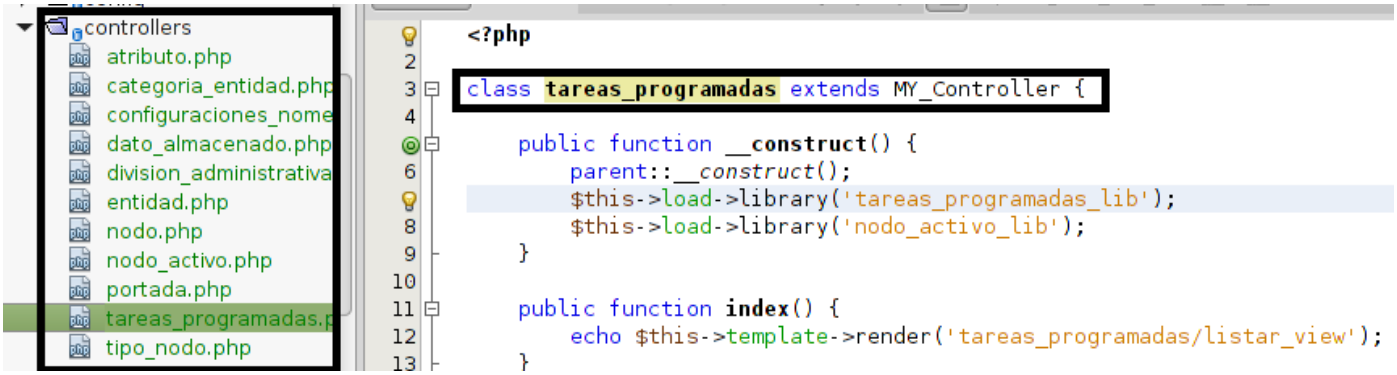


Figura 11: Patrón Controlador

2.7.2 Patrones GoF

Los patrones "Banda de los Cuatro" (*Gang-of-Four*) describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas. Existen tres tipos de patrones: de creación, estructurales y de comportamiento. Los patrones de creación abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas (15).

Los patrones GoF que se utilizaron en el desarrollo del componente son:

Patrón de comportamiento: Comprenden la asignación de responsabilidades entre objetos y algoritmos. Estos no solo conciernen a los objetos y las clases sino también la comunicación entre estas, además caracterizan flujos de control complejos que son difíciles de seguir en tiempo de ejecución (31).

- **Mediador (*Mediator*):** Este patrón se evidencia en las librerías, las cuales son mediadoras entre las clases controladoras y las modelos o acceso a datos. En la figura 12 se muestra el método registrarTareaPrograma de la clase controladora tarea_programada, que recibe las peticiones y envía las respuestas pertinentes; en la figura 13 se muestra la librería tareas_programadas_lib.php, que será una capa intermedia que implementa la lógica de registrar una tarea programada apoyándose de la clase modelo tb_dtareas_programadas_mdl para el acceso a datos.

```
public function registrarTareaProgramada() {
    $all_post = $this->input->all_post(TRUE);
    $script = $all_post['script'];
    $script_limpio = str_replace('|', ' ', $script);
    pr($script_limpio);
    $all_post['script'] = $script_limpio;


    if ($this->tareas_programadas_lib->existeTareaProgramada($all_post, false))
        throw new Exception_Error('SYS005');
    else {
        if ($this->tareas_programadas_lib->registrarTareaProgramada($all_post)) {
            $this->message('SYS001');
        } else {
            throw new Exception_Error('SYS006');
        }
    }
}
```

Figura 12: Patrón Mediador para la controladora

```php
public function registrarTareaProgramada($parametros) {
    $datosFinales = array();
    $datosFinales['nombre_tarea_programada'] = $parametros['nombre_tarea_programada'];
    $datosFinales['accion'] = $parametros['accion'];
    $datosFinales['periodo'] = $parametros['periodo'];
    $datosFinales['base_dato'] = $parametros['base_dato'];
    $datosFinales['script'] = $parametros['script'];

    if (isset($parametros['activo']))
        $datosFinales['activo'] = 't';
    else
        $datosFinales['activo'] = 'f';

    if (isset($parametros['recursiva']))
        $datosFinales['recursiva'] = 't';
    else
        $datosFinales['recursiva'] = 'f';

    $this->_ci->db->trans_start();
    $this->_ci->tb_dtareas_programadas_mdl->registrar($datosFinales);

    $this->_ci->db->trans_complete();
    return $this->_ci->db->trans_status();
}
```

Figura 13: Patrón Mediador para la librería

Patrones de creación: Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de que clase crear o cómo crearla (33).

- **Instancia única (*Singleton*):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto (34). En la figura 14 se muestra la clase librería tarea_programada_lib.php la cual tiene en su constructor la instancia de la modelo tarea_programadas_mdl.php, de esta manera no hay necesidad de instanciar esta modelo en cualquier otra parte de la clase. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor.

```php
public function __construct() {
    $this->_ci = & get_instance();
    $this->_ci->load->model('tb_dtareas_programadas_mdl');
}

public function obtenerTareasProgramadas($inicio = null, $limite = null, $elementoOrdenar = null, $dirOrde
    return $this->_ci->tb_dtareas_programadas_mdl->obtenerTareasProgramadasPorPagina($inicio, $limite, $el
}
```

Figura 14: Patrón Instancia única

2.8 Patrones de diseño de base de datos

Para el diseño y la construcción de una base de datos se requiere del mayor análisis posible pues a partir de este diseño se crea la base de datos, en la actualidad suelen ser muy grandes y a veces el trabajo con los patrones de diseño hacen que el trabajo sea más fácil además asegura un resultado correcto (35). A continuación se explican los patrones de diseño de base de datos utilizados para facilitar el almacenamiento de datos en el desarrollo del componente Configuración.

- **Patrón Entidad-Atributo-Valor:** En la versión 2.0 del componente Configuración, se utiliza en la base de datos el patrón Entidad-Atributo-Valor (EAV), para gestionar las entidades, dado que estas no tienen todos sus atributos definidos. Este patrón es flexible y utiliza entre sus variantes una tabla para gestionar los objetos, una para los atributos y otra para los valores que tiene cada atributo por cada objeto (35). La propia ventaja de ser flexible se convierte en una desventaja cuando se analiza por rendimiento, crecimiento de la tabla valores y la complejidad para las consultas. Existen diferentes alternativas para simular el patrón EAV, entre las que se encuentra el uso de los tipos de datos *json* y *hstore*. Para la versión 2.0 del componente se hace uso del tipo de dato *json* por ser un formato más ligero y sencillo para el intercambio de datos. A continuación se muestra en la figura 15 un ejemplo de este tipo de dato para definir el dominio de los atributos que tendrán lugar como valor en la tabla Entidad-Atributo-Valor.

```
FOR auxiliar in select * from sq_configuracion_sistema.tb_rentidad_atributo_valor where id_entidad = $1 LOOP

        dominio = tb_natributo.dominio from sq_configuracion_sistema.tb_natributo where id_atributo = auxiliar.id_atributo;
        long_dominio = json_array_length(dominio);
        nombre_atributo = tb_natributo.nombre_atributo from sq_configuracion_sistema.tb_natributo where id_atributo = auxiliar.id_atributo;

        if(long_dominio = 0) then
                texto = auxiliar.valor;
        else
                FOR i in 0..(long_dominio - 1) LOOP
                        if dominio->i->>'valor' = auxiliar.valor then
                                texto = dominio->i->>'texto';
                        end if;
                END LOOP;
```

Figura 15: Patrón Entidad-Atributo-Valor

- **Patrón llave subrogada:** Este patrón es muy utilizado porque facilita la interacción con la base de datos en un futuro. El mismo plantea que se genere una llave primaria única para cada entidad, en vez de usar un atributo identificador en el contexto dado. Normalmente se usan enteros en

columnas autoincrementales o GUID[26] que están demostradas que no se repiten o con una probabilidad extremadamente baja. Esto permite que las tablas sean más fáciles de consultar a partir del identificador, pues todos tienen el mismo tipo en cada una de las tablas (35). La utilización del patrón está presente en la tabla tb_dnodo, en vez de usar el campo nombre_nodo como llave se utiliza un campo numérico auto incrementable llamado id_nodo.

- **Patrón árbol simple:** El patrón árbol simple es utilizado cuando el árbol es la representación de una estructura de datos. Los elementos a almacenar son del mismo tipo, es decir, pueden ser almacenados en la misma entidad. En este patrón no pueden existir ciclos, pues un hijo no puede ser su propio padre. Cuando se habla de nodos hijos y nodos padres se refiere al nivel de los nodos, los que están arriba son los nodos padres y sus derivados, los nodos hijos (35). En la versión 2.0 del componente Configuración se ve presente en las tablas tb_dnodo y tb_darea_administrativa.

2.9 Modelo de la base de datos

Un modelo de datos es una colección de conceptos y reglas que se emplean para describir la estructura de una base de datos que incluye entidades, atributos y relaciones entre estos (36). Para diseñar la base de datos se elaboró el modelo físico de datos. Las clases persistentes que a continuación se muestran cubren las necesidades de la propuesta de solución.

[26] **GUID**: Por sus siglas en inglés significa Identificador Único Global, es un número pseudo-aleatorio empleado en aplicaciones de *software*. Aunque no se puede garantizar que cada GUID generado sea único, el número total de claves únicas es tan grande que la posibilidad de que se genere un mismo número dos veces puede considerarse nula en la práctica.

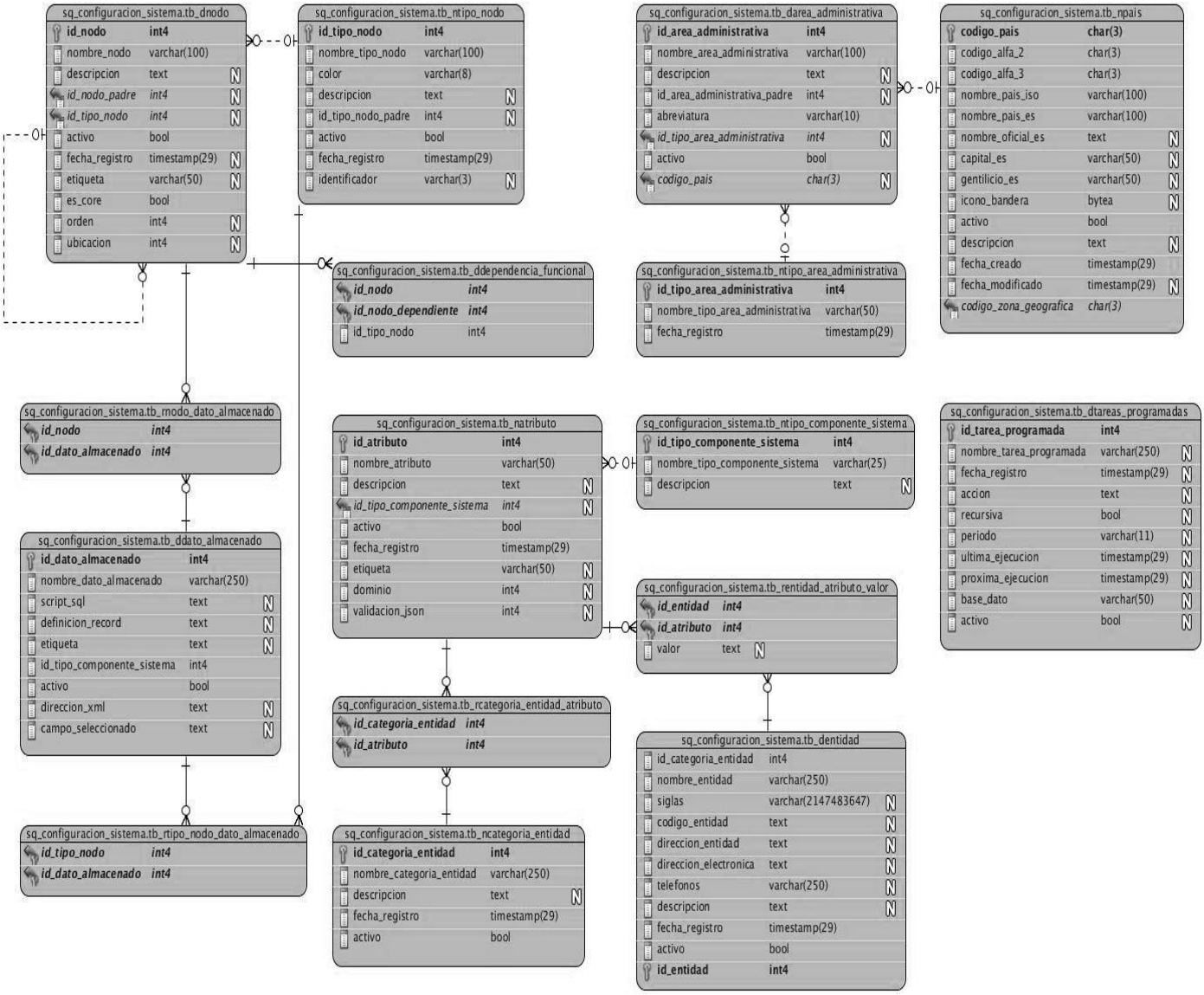


Figura 16: Modelo de la base de datos

2.10 Diagrama de despliegue

El diagrama de despliegue se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el *hardware* y el *software* que se despliega, estas relaciones son representadas por los protocolos de comunicación que se utilizan para acceder a cada uno. En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta:

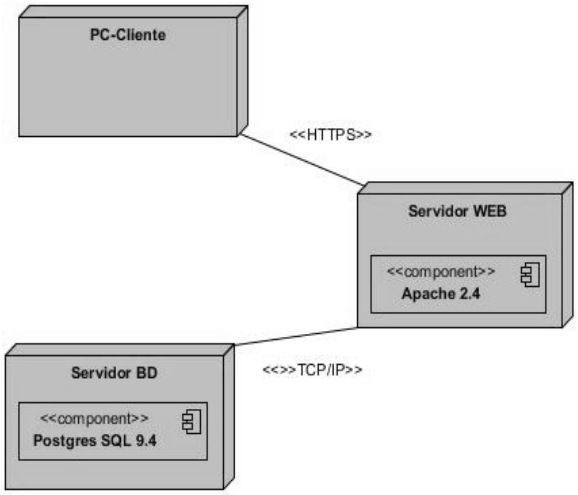


Figura 17: Diagrama de despliegue del componente Configuración

PC-Cliente: Ordenador desde donde los usuarios acceden a la aplicación. Tiene como función principal acceder al sistema e interactuar con el mismo según sus necesidades.

Servidor web: Ordenador donde se encuentra el servidor web Apache, este será el lugar en que se gestione todo el contenido de la aplicación y las máquinas clientes acceden a través de un navegador web.

Servidor BD: Es el encargado de almacenar toda la información generada por el sistema, en él se encuentra ubicada la base de datos de donde los servicios que brinda el componente se nutren de información.

HTTPS: Protocolo de transferencia de hipertexto seguro, por sus siglas en inglés, *Hypertext Transfer Secure Protocol* (HTTPS), es un protocolo de red basado en HTTP por lo que está orientado a transacciones sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores y sigue el esquema petición-respuesta entre un cliente y un servidor (37).

TCP/IP: Base de Internet y sirve para enlazar computadoras que utilizan diferentes sistemas operativos. Familia de protocolos utilizada para la conexión entre el servidor web y el servidor donde se encuentra ubicada la base de datos (37).

2.11 Conclusiones parciales

Luego de describir las características que debe cumplir el componente Configuración basado en tecnologías libres y de realizar el análisis y diseño correspondiente, se concluye que los requisitos funcionales y no funcionales identificados a partir del proceso de obtención de los requisitos y los

artefactos generados, constituyeron una guía fundamental para la construcción de la propuesta de solución. Con la realización del modelo físico de datos se hizo posible materializar la visión de la base de datos a utilizar. Además con la utilización de la arquitectura Modelo-Vista-Controlador y el uso de los patrones de diseño y de base de datos propuestos, se garantizará una mayor organización, reutilización de funciones y código más legible. Por otro lado la realización del modelo de despliegue ilustra la comunicación de los distintos componentes en los cuales se divide el sistema.

Capítulo 3: Construcción y validación de la propuesta de solución.

3.1 Introducción

En el presente capítulo se describe la implementación del *software*, fase donde finalmente se materializa el producto y cumple con los requisitos obtenidos al inicio de la investigación. Se definen los estándares de codificación que debe cumplir el equipo de desarrollo, así como los métodos y técnicas para la realización de las pruebas, con el propósito de validar la solución. El proceso de pruebas está dirigido a componentes del *software*, con el objetivo de medir el grado en que se cumplen los requisitos exigidos por el cliente y si posee la calidad requerida.

3.2 Paradigmas de programación

Los paradigmas de programación indican las diversas formas que a lo largo de la evolución de los lenguajes, han sido aceptadas como estilos para programar y para resolver los problemas por medio de una computadora (38). Indican un método para realizar un programa de cómputo y la manera en que se debe estructurar y organizar las tareas que se deben llevar a cabo en un programa (31).

Programación orientada a objetos (POO)

POO es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Los objetos son entidades que combinan estado (atributo), comportamiento (método) e identidad (39).

El marco de trabajo GUUD facilita la forma de programar en PHP y permite reutilizar el código con el paradigma de programación orientado a objetos que ya posee implementado. En él, cada objeto es responsable de inicializarse y destruirse de forma correcta, no existe la necesidad de llamar explícitamente al procedimiento de creación o de terminación debido a que el marco de trabajo se encarga de esto. Una de las principales características de esta programación es que proporciona la herencia lo que permite la reutilización del código, por lo que GUUD la utiliza entre los distintos tipos de clases como las modelos y las controladoras.

Programación dirigida por eventos (PDE)

La PDE es un paradigma de programación donde el flujo del programa está determinado por eventos o mensajes desde otros programas o hilos de ejecución. Es la base de lo que se denomina interfaz de

usuario, aunque puede emplearse también para desarrollar interfaces entre componentes de *software* o componentes de núcleos. Un programa dirigido por eventos debe haber sido creado en un lenguaje de programación orientado a objetos y cada objeto espera algún evento que realice el usuario sobre él (40).
GUUD implementa este paradigma de programación debido a que el marco de trabajo JQuery lo utiliza. Esta librería es usada para el manejo de las interfaces de usuario y utiliza los principales eventos que provee el navegador para la interacción de sus componentes. Estos eventos pueden ser desencadenados por el usuario o por otros eventos en el sistema, permitiendo una mayor interoperabilidad usuario-sistema.

3.3 Estándares de codificación

Los estándares de codificación son reglas de codificación que permiten tener una programación homogénea, comprendiendo todos los aspectos de la generalización del código, pues la aplicación debe de estar implementada como si un único programador escribiera el código de una sola vez. La usabilidad de estos permite conservar el código fuente entendible y fácil de mantener, además de mejorar la forma en la que se programa (41).
Para el desarrollo del componente se utilizaron los estándares de codificación establecidos por la DIN con el propósito de estandarizar las nomenclaturas en la implementación del mismo y obtener un producto estable.

3.3.1 Indentación, llaves de apertura y cierre, y tamaño de líneas

Usar una indentación[27] sin tabulaciones, con un equivalente a 4 espacios, para mantener integridad en las revisiones svn[28]. El uso de las llaves "{}" será en una nueva línea. La longitud de las líneas de código es aproximadamente de 75-80 caracteres. Para mantener la legibilidad del código. La indentación se ve reflejada en la función index() de la controladora Nodo:

```
public function index()
{
    echo $this->template->render('nodo_activo/listar_view');
}
```

Figura 18: Indentación

[27] **Indentación**: Es un anglicismo de la palabra inglesa *indentation*, de uso común en la informática. No es un término reconocido por la Real Academia Española. En los lenguajes de programación es un tipo de notación secundaria utilizado para mejorar la legibilidad del código fuente por parte de los programadores.
[28] **Subversion:** Abreviado frecuentemente como SVN, por el comando svn, es una herramienta de control de versiones *open source* basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es *software* libre bajo una licencia de tipo *Apache/BSD*.

3.3.2 Convención de nomenclatura

Variables: Se rigen por la nomenclatura *camelCase*[29]. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Se muestra en el método obtenerTipoNodoId() un ejemplo donde se emplean las variables:

```
public function obtenerTipoNodoDadoId($id) {
    $tipoNodo = $this->_ci->tb_ntipo_nodo_mdl->obtenerDadoId(array('id_tipo_nodo' => $id));
    return $tipoNodo;
}
```

Figura 19: Convención de nomenclatura variable

Clases: Siempre comienzan con mayúscula, en caso de nombre compuesto las palabras se separan con guión bajo "_" y el resto en minúscula.

Se muestra un ejemplo de la nomenclatura de una clase simple y una compuesta:

```
/**
 * @property atributo_lib $atributo_lib
 */
class Atributo extends MY_Controller
{
```

Figura 20: Convención de nomenclatura de clase simple

```
/**
 * @property categoria_entidad_lib $categoria_entidad_lib
 * @property atributo_lib $atributo_lib
 */
class Categoria_entidad extends MY_Controller
{
```

Figura 21: Convención de nomenclatura de clase compuesta

Funciones: Se rigen por la nomenclatura *camelCase*. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa.

Se muestra un ejemplo de la función modificarNodo(), donde se emplea la nomenclatura funciones:

[29] ***camelCase*:** Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en *camelCase* se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.

```
public function modificarNodo($datos)
{
    $this->_ci->db->trans_start();
    $nodo = array();
    $donde = array('id_nodo' => $datos['id_nodo']);
    $nodo['nombre_nodo'] = $datos['nombre_nodo'];
    $nodo['etiqueta'] = $datos['etiqueta'];
    $nodo['orden'] = $datos['orden'];
    $nodo['id_tipo_nodo'] = $datos['id_tipo_nodo'];
    $nodo['id_nodo_padre'] = $datos['id_nodo_padre'];
    $this->_ci->tb_dnodo_mdl->modificar($donde, $nodo);

    if (isset($datos['datos_almacenados']))
    {
        $datosAntes = $this->_ci->tb_rnodo_dato_almacenado_mdl->obtenerDadoAtributos($donde);
        $datosDespues = $datos['datos_almacenados'];
        $arrAntes = array();
        foreach ($datosAntes as $value)
        {
            array_push($arrAntes, $value->id_dato_almacenado);
        }
```

Figura 22: Convención de nomenclatura de función

Ficheros: Todo siempre en minúscula y en caso de nombres compuestos se usa el guión bajo, seguido del sufijo definido para cada tipo de fichero.

Vistas: Intuitivo y relacionado con el formulario y/o vista que representa. Sufijo "_view".

Ejemplo: detalles_atributo_view.php

Modelos: Mismo nombre de la tabla de la base de datos para la cual se crea. Sufijo "_mdl".

Ejemplo: tb_ddato_almacenado_mdl.php

Librerías: Mismo nombre de la clase que representa. Sufijo "_lib".

Ejemplo: entidad_atributo_valor_lib

Controladoras: Mismo nombre de la clase que representa.

Ejemplo: Categoria_entidad.

3.3.3 Estructura de control

Las estructuras de control incluyen *if, for, foreach, while, switch.* Entre las estructuras de control y los paréntesis debe de existir un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Con esto se aumenta la legibilidad del código y se disminuye la probabilidad de errores lógicos.

3.3.4 Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Se debe de cumplir con el siguiente bloque al principio de cada clase, como se muestra en la clase TareasProgramadas:

```
/*
 * Obtien las provincias
 * @author: Adrian Segura - Suyin Margarita Perez
 * @return array()
 */

public function obtenerProvincias()
{
    return $this->_ci->tb_darea_administrativa_mdl->obtenerProvincias();
}
```

Figura 23: Documentación de clase

```
/*
 * Clase controladora TareasProgramadas
 * Esta clase maneja la logica de gestionar las tareas programadas
 * @pakage Configuracion
 * @category Controller
 * @author: Adrian Segura - Suyin Margarita Perez
 */

class TareasProgramadas extends MY_Controller
{

}
```

Figura 24: Documentación de funciones

3.3.5 Buenas prácticas

Los valores booleanos y nulos siempre se escriben con mayúscula, para facilitar la legibilidad del código. Se debe usar un *enter* antes de las estructuras de control y definición de las funciones.

3.4 Validación de requisitos

La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proyecto y el producto (42).

3.4.1 Criterios para validar los requisitos

Para validar los requisitos del sistema según Pressman se pueden chequear mediante un cuestionario guiado por un conjunto de interrogantes con el objetivo de descubrir la mayor cantidad de errores posibles. A continuación se muestra las preguntas utilizadas, algunas fueron agregadas a las planteadas por Pressman por ser consideradas necesarias para la validación.

Interrogantes para la validación de requisitos:

¿Está el requisito claramente definido? ¿Puede interpretarse mal?

¿Está identificado el origen del requisito (por ejemplo: persona, norma, documento)? ¿El planteamiento final del requisito ha sido contrastado con la fuente original?

¿El requisito está delimitado en términos cuantitativos?

¿Qué otros requisitos hacen referencia al requisito estudiado?

¿El requisito incumple alguna restricción definida?

¿El requisito es verificable? Si es así, ¿podemos efectuar pruebas para verificar el requisito?

¿Se puede seguir el requisito en el modelo del sistema que hemos desarrollado?

¿Está el requisito asociado con los rendimientos del sistema o con su comportamiento?

¿El requisito está implícitamente definido?

¿El requisito es modificable?

¿El requisito está completo?

¿El requisito puede ser implementado?

¿El requisito puede ser probado?

¿El resultado de la evaluación de impacto es positivo?

Resultado de aplicar los criterios de validación

Luego de aplicar el conjunto de interrogantes para validar los requisitos definidos para el desarrollo del componente, se obtuvo el 100 % de aprobación por parte del cliente.

3.4.2 Técnicas de validación de requisitos

Con el objetivo de obtener una mayor calidad y demostrar que los requisitos definidos realmente describen al sistema que el cliente necesita; se utilizaron las técnicas para la validación de requisitos siguientes:

- **Prototipado de interfaz:** Permite al cliente entender fácilmente la propuesta de solución, al brindar la representación aproximada de la interfaz de usuario que tendrá el sistema. Existen dos tipos principales de prototipo de interfaz: -**Desechables** se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en *software*. – **Evolutivos** una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final (43). El tipo utilizado finalmente para la propuesta de solución fue Evolutivos pues de esta forma se reutiliza el prototipo diseñado en todo el proceso de desarrollo del componente.

- **Generación de casos de prueba:** Se realizaron los diseños de casos de pruebas para cada uno de los requisitos obtenidos, permitiendo verificar que todos se pudieran probar y determinar en la medida de la complejidad del diseño de caso de prueba, identificando requisitos que deberían ser reconsiderados y cuáles pueden ser los más difíciles de implementar (43).

Resultado de aplicar las técnicas de validación

Como resultado de este proceso se identificaron inconsistencias en las especificaciones tales como:

- Falta de concordancia entre la complejidad de la especificación y la registrada en el documento de Evaluación de requisitos.
- Descripciones de requisitos poco detalladas.
- Errores ortográficos.

3.5 Estrategia de pruebas

Para evaluar la calidad del componente que se está desarrollando y verificar el cumplimiento de los objetivos trazados, se aplicaron un conjunto de pruebas definidas por Pressman en su libro de Ingeniería del *software* "Un enfoque práctico", en su quinta edición. A continuación se muestra la estrategia de prueba diseñada para aplicar en la solución desarrollada:

Tabla 4: Estrategia de prueba

Prueba	**Método**	**Técnica**
Prueba de unidad	Caja blanca	Prueba de camino básico
Prueba de integración	Caja negra	Incremental
Prueba de sistema: Prueba de resistencia (*stress*) y prueba de rendimiento (carga)	Caja negra	Automático
Prueba de validación	Caja negra	Partición equivalente
Prueba de aceptación	Caja negra	Alfa y Beta.

3.5.1 Prueba de unidad

Las pruebas de unidad tienen como objetivo verificar la unidad más pequeña del diseño del *software*. Estas pruebas se concentran en la lógica del procesamiento interno y en las estructuras de datos tales como: código fuente, archivos binarios, archivos de datos, entre otros. Este tipo de prueba se puede aplicar en paralelo a varios componentes.

Las pruebas de unidad se realizan mediante el **método de caja blanca o estructural**, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de

control del diseño procedimental para obtener los casos de prueba. La técnica de prueba de caja blanca utilizada en la investigación es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (44).

Esta prueba se realiza de forma automática en el sistema, para ello el marco de trabajo GUUD posee la librería unit_test para la ejecución de pruebas estructurales. Aunque es bastante sencilla, cuenta con una sola función de evaluación y dos funciones de resultados, permite determinar con certeza si un código específico produce el tipo de dato y resultado esperado. Para ejecutar la prueba se utiliza la línea de código: *$this->unit->run*(código, resultado esperado, 'nombre de prueba'); donde “código” es el segmento de código que se desea probar, “resultado esperado” es lo que debe devolver la evaluación del código que puede ser un tipo de dato o un valor literal y “nombre de prueba” es el nombre que se le puede dar a la prueba.

A continuación se muestra el caso de prueba del método existeNodo() de la funcionalidad Crear nodo, para ver los casos de prueba en su totalidad dirigirse al anexo #3.

Tabla 5: Prueba estructural de caja blanca para la funcionalidad Crear nodo

Prueba estructural de caja blanca.	**Código de caso de prueba.**
	Crear nodo.
Probador:	Adrian Segura Tristá
Tipo de dato esperado:	*Boolean*.

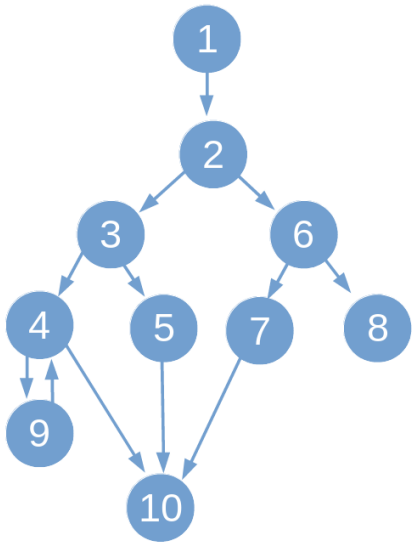
Código al que se aplica:

```
    public function existeNodo($datos, $modificar = false) {
        $bandera = false;
        $arr = array();
        $arr['id_nodo_padre'] = $datos['id_nodo_padre'];

        if ($modificar == true) {
            $arr['etiqueta '] = $datos['etiqueta'];
            $nodo = $this->_ci->tb_dnodo_mdl->obtenerDadoAtributos($arr);
            if (!empty($nodo)) {
                foreach ($nodo as $objP) {
                    if ($objP->id_nodo != $datos['id_nodo']) {
                        $bandera = true;
                    }
                }
            }
            else
                $bandera = false;
        } else {
            $arr['nombre_nodo'] = $datos['nombre_nodo'];
            $nodo = $this->_ci->tb_dnodo_mdl->obtenerDadoAtributos($arr);
            $contador = count($nodo);
            if ($contador == 0)
                $bandera = false;
            else
                return true;
        }
        return $bandera;
    }
```

Complejidad ciclomática:
V(G) = (E − N) + 2 * P
V(G) = (10 − 9) + 2 * 2 = 5
Caminos independientes:
1. 1→2→6→8
2. 1→2→6→7→10
3. 1→2→3→ 4→10
4. 1→2→3→5→10

Representación del grafo:



Caso de prueba para los caminos básicos	
Tipo de dato esperado:	*Boolean*

Función de evaluación:

```
15      public function index() {
16          $all_post = array();
17          $all_post['nombre_nodo'] = 'nodo_prueba';
18          $all_post['etiqueta'] = 'Nodo de prueba';
19          $all_post['id_nodo_padre'] = 1;
20          $all_post['id_tipo_nodo'] = 2;
            $all_post['orden'] = 50;
22          echo $this->unit->run($this->nodo_activo_lib->existeNodo($all_post), FALSE, 'Existe nodo');
23      }
```

Resultados de la prueba:

Test Name	Existe nodo
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	/var/www/akademos/base/application/sistema/configuracion/controllers/nodo_activo.php
Line Number	23

Evaluación del caso de prueba:	Satisfactoria

Se llevaron a cabo tres iteraciones para detectar no conformidades, las cuales arrojaron los siguientes resultados como se muestra en la tabla 6. Las no conformidades encontradas fueron resueltas en su totalidad.

Tabla 6: Iteración de las pruebas de caja blanca

Iteración	Cantidad de no conformidades	Asociadas a
Primera	7	Errores de validación, entradas y salidas incorrectas.
Segunda	2	Errores de validación.
Tercera	0	-

3.5.2 Prueba de integración

La prueba de integración es una técnica sistemática para construir la estructura de un programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. Existen dos tipos de integración: no incremental e incremental. En el primer caso se combinan todos los módulos y se prueba el programa en su conjunto, como es lógico pensar el resultado puede ser caótico con un gran número de fallos y la consiguiente dificultad para identificar el módulo que los provocó. Por su parte, en la integración incremental el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir. Por esta razón se escogió el enfoque incremental para la realización de las pruebas de integración de la solución (45).

Resultado

El componente Configuración se integra a la mayoría de los sistemas que componen el SGU, para ello se utilizaron el método de caja negra. A continuación se muestra un ejemplo de la prueba realizada al componente Carrera, para verlas en su totalidad dirigirse al anexo #4.

Tabla 7: Caso de prueba de integración con el componente Carrera

Caso de prueba de integración del componente Configuración
Sistema/componente al que se integra: Carrera
Condiciones de ejecución: El componente Carrera haya introducido los datos en la base de datos central y exista conexión con el componente Configuración.
Descripción de la prueba: Comprobar que el componente Carrera proporcione la información referente a las carreras y los cursos académicos.
Pasos de ejecución: Se realiza la petición a nivel de base de datos mediante la librería *plproxy* para obtener los cursos académicos así como las carreras, que se gestionan en el componente Carrera.
Resultados esperados: El componente Configuración brinda toda la información solicitada.
Evaluación: Prueba satisfactoria.

3.5.3 Pruebas de sistema

La prueba del sistema, realmente, está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas (46).

Pruebas de resistencia

Las pruebas de resistencia están diseñadas para enfrentar a los programas con situaciones anormales. Estas pruebas se ejecutan en el sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. Por ejemplo: -1- diseñar pruebas especiales que generen diez interrupciones por segundo, cuando las normales son una o dos; -2- incrementar las frecuencias de datos de entrada en un orden de magnitud con el fin de comprobar cómo responden las funciones de entrada; -3- ejecutar casos de prueba que requieran el máximo de memoria o de otros recursos; -4- diseñar casos de prueba que puedan dar problemas en un sistema operativo virtual o -5- diseñar casos de prueba que produzcan excesivas búsquedas de datos residentes en disco. Esencialmente, el responsable de la prueba intenta romper el programa (47).

Pruebas de rendimiento

La prueba de rendimiento está diseñada para probar el rendimiento del *software* en tiempo de ejecución dentro del contexto de un sistema integrado. La prueba de rendimiento se da durante todos los pasos del proceso de la prueba y a menudo, van emparejadas con las pruebas de resistencia (48).

Resultado

Se realizaron las pruebas para comprobar que el tiempo de respuesta del componente no exceda a los 5 segundos de forma general, así como el tiempo de respuesta por cada transacción promedio sea de un mínimo de 1,5 segundos y un máximo de 5 segundos. A continuación se muestran las pruebas de desempeño realizadas en el navegador web Mozilla Firefox y en el Apache JMeter.

En el navegador web Mozilla Firefox se comprobaron varias funcionalidades, donde se obtuvo un resultado satisfactorio.

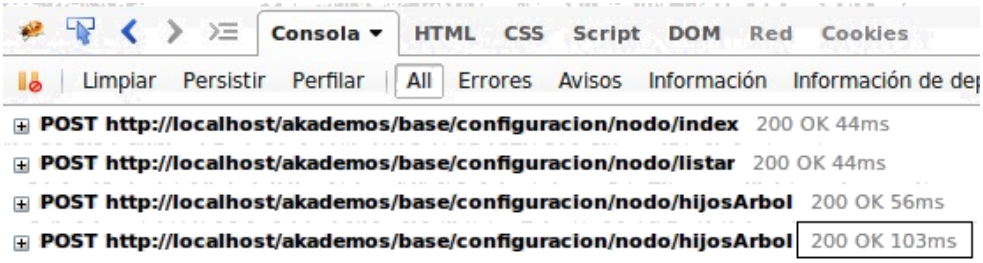


Figura 25: Prueba de desempeño a la funcionalidad Nodo

En la herramienta Apache JMeter se aplicaron las pruebas a varias funcionalidades, a continuación se muestran los resultados obtenidos de tres de estas funcionalidades.

Tabla 8: Resultados de las pruebas de sistema

Aplicado a	Usuarios(Muestras)	Tiempo de ejecución (ms)			Rendimiento		
		Mín.	Máx.	Media	% Error	Pet/seg	Kb/seg
Crear nodo	1200	4	162	348	0.0%	2.1/sec	4.36
	1300	4	170	321	0.0%	2.1/sec	4.27
	1400	5	170	298	0.0%	2.2/sec	4.43
Crear división administrativa	1000	4	153	121	0.0%	20.3/sec	41.44
	1200	4	212	128	0.0%	17.8/sec	36.30
	1400	4	212	224	0.0%	16.8/sec	34.40
Configurar nodo	1000	3	182	114	0.0%	16.7/sec	32.22
	1200	4	182	125	0.0%	13.7/sec	26.61
	1400	4	182	216	0.0%	14.6/sec	28.71

Columna mínimo (Mín.): Indica el mínimo de tiempo de ejecución invertido para una petición con n (columna Muestras) usuarios haciendo peticiones de manera concurrente.

Columna máximo (Máx.): Indica el máximo de tiempo de ejecución invertido para una petición con n (columna Muestras) usuarios haciendo peticiones de manera concurrente.

Media: Representa el tiempo de ejecución promedio de una petición con n usuarios.

Error: Indica la relación entre el total de peticiones y el número de peticiones que originaron errores.

Rendimiento (Pet./seg): Hace referencia al número de peticiones que el servidor puede procesar en un segundo.

Rendimiento (Kb./seg): Hace referencia a la cantidad de datos que el servidor puede procesar en un segundo.

3.5.4 Prueba de validación

Tras la culminación de la prueba de integración, el *software* está completamente ensamblado como un paquete, seguidamente se puede comenzar la prueba de validación. La validación se consigue cuando el *software* funciona de acuerdo con las expectativas razonables del cliente. Una vez que se procede con cada prueba de validación, puede darse una de las dos condiciones siguientes: -1- las características de funcionamiento o de rendimiento estén de acuerdo con las especificaciones y son aceptables; o -2- se descubre una desviación de las especificaciones y se crea una lista de deficiencias (49).

Para el desarrollo de las pruebas de validación se aplicó el **método de caja negra** con la **técnica de partición de equivalencia**, donde se demuestran la conformidad de los requisitos.

Los **métodos de caja negra**, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del *software*. O sea, permite al ingeniero del *software* obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a base de datos externas, errores de rendimiento y errores de inicialización y de terminación (50).

La técnica de **partición de equivalencia** consiste en ejecutar el flujo básico de las funcionalidades utilizando datos válidos e inválidos, donde se generaron los artefactos “Diseño de casos de pruebas”. Por cada requisito funcional del sistema se generó un documento donde se recogen todos los datos necesarios para probar la interfaz (51).

En el diseño de caso de prueba hay que tener en cuenta que V indica válido, I inválido y NA que no aplica. Las variables Nombre, Parámetros de configuración, Etiqueta, Tipo de nodo, Nodo padre y Orden significan los valores de entrada de datos para los casos de pruebas; Resultado del sistema es la variable de salida y en el Flujo central se especifican los pasos para lograr realizar las operaciones. La condición para ejecutar la prueba es que el usuario debe estar autenticado en el sistema y si ocurre un error durante la operación se muestra el mensaje: “Ha ocurrido un error cuando intentaba realizar la acción”.

Resultados

Seguidamente se describe el caso de prueba Crear nodo activo, para ver los diseños de casos de pruebas en su totalidad, consultar el expediente del proyecto Núcleo.

Tabla 9: Diseño de casos de prueba. Crear nodo activo. Parte I

Escenario	Descripción	Nombre	Parámetros de Configuración	Etiqueta	Tipo de nodo
EC 1.1 Insertar datos correctamente	Este escenario permite crear un nodo activo utilizando datos correctos.	V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		V	V	V	V
		"Insertar de 2 a 250 caracteres con palabras hasta 30 caracteres".	“Rol Administrador”	"Insertar de 2 a 100 caracteres con palabras hasta 30 caracteres".	“Aplicación”
		V	V	V	V
		"Pregrado5"	“Rol Administrador”	"pregrado"	“Módulo”
EC 1.2 Insertar elemento repetido	Este escenario no permite crear un nodo activo con datos repetidos.	I	V	I	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		I	V	I	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
EC 1.3 Insertar elementos incompletos.	Este escenario no permite introducir datos que son obligatorios.	I	I	I	I
		"Vacío"	“vacío”	"Vacío"	“Seleccionar”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
EC 1.4 Insertar	Este escenario no	I	V	I	V

datos incorrectos	permite introducir datos incorrectos.	"@gafsie"	“Rol Administrador”	"@gafsie"	“Módulo”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		I	V	I	V
		"Insertar más de 250 caracteres"	“Rol Administrador”	"Insertar más de 100 caracteres"	“Módulo”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		I	V	I	V
		“insertar menos de 2 caracteres”.	“Rol Administrador”	“insertar menos de 2 caracteres”.	“Módulo”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
		I	V	I	V
		“introducir más de 30 caracteres en una palabra”	“Rol Administrador”	“introducir más de 30 caracteres en una palabra”	“Módulo”
		V	V	V	V
		"Pregrado"	“Rol Administrador”	"pregrado"	“Módulo”
EC 1.5 Cancelar operación	Este escenario permite cancelar la operación realizada.	NA	NA	NA	NA

Tabla 10: Diseño de casos de prueba. Crear nodo activo. Parte II

Nodo padre	Orden	Respuesta del sistema	Flujo central
V	V	El sistema una vez creado el nodo activo,	-Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. Una

“Pregrado”	"Insertar solo números"	muestra el mensaje de información: “El elemento ha sido creado satisfactoriamente” y luego regresa al árbol de los nodos activos.	vez autenticado el usuario en el sistema, selecciona “Configuración” en la barra de íconos de los procesos horizontales. Selecciona en la agrupación funcional “General” de la derecha, la funcionalidad “Nodo activo”. El sistema muestra el árbol de nodos activos existentes hasta la fecha. Selecciona la opción “Crear” en el área de iconos flotantes. El sistema brinda la posibilidad de insertar todos los campos, además de dar la opción de Aceptar o Cancelar. -Se llenan los datos correctamente para crear un nodo activo. -Selecciona el botón Aceptar y muestra el mensaje “El elemento ha sido creado satisfactoriamente”.
V	V		
“SGU”	"Insertar de 1 a 10 caracteres"		
V	V		
“Pregrado”	"2"		
V	V	El sistema muestra un mensaje de error: “El elemento ya existe”.	Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. Una vez autenticado el usuario en el sistema, selecciona “Configuración” en la barra de íconos de los procesos horizontales. Selecciona en la agrupación funcional “General” de la derecha, la funcionalidad “Nodos activos”. El sistema muestra un árbol de los nodos activos existentes hasta la fecha. Se Selecciona la opción “Crear” en el área de iconos flotantes. El sistema brinda la posibilidad de insertar datos repetidos. Presiona el botón de Aceptar o Cancelar.
“Pregrado”	"2"		
V	V		
“Pregrado”	"5"		
I	V	El sistema muestra el mensaje “Campo requerido” y se muestra en rojo el campo que debe ser llenado obligatoriamente.	Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. Una vez autenticado el usuario en el sistema, selecciona “Configuración” en la barra de íconos de los procesos horizontales. Selecciona en la agrupación funcional “General” de la derecha, la funcionalidad “Nodos activos”. El sistema muestra un árbol de los nodos activos existentes hasta la fecha. Se selecciona la opción “Crear” en el área de iconos flotantes. El sistema brinda la posibilidad de llenar todos los campos, además de dar la opción de Aceptar y Cancelar. Se llenan los campos con datos incorrectos para crear un nodo activo. Selecciona el botón Aceptar y muestran los mensajes de errores correspondientes.
“Seleccionar”	"vacío"		
V	V		
“Pregrado”	"Vacío"		
V	V	El sistema muestra el mensaje encima del campo “Entre solo letras, números, y espacios o guión bajo entre palabras".	Al autenticarse el usuario, el sistema muestra por defecto todos los subsistemas que están contenidos dentro. Una vez autenticado el usuario en el sistema, selecciona “Configuración” en la barra de íconos de los procesos horizontales. Selecciona en la agrupación funcional “General” de la derecha, la funcionalidad “Nodos activos”. El sistema muestra un árbol de los nodos activos existentes hasta la fecha. Se selecciona la opción “Crear” en el área de iconos flotantes. El sistema brinda la posibilidad de llenar todos los campos, además de dar la opción de
“Pregrado”	"2"		
V	V		
“Pregrado”	"2"		
V	V	Si el usuario introduce más de la cantidad de caracteres permitidos, el sistema no le permitirá	
“Pregrado”	"2"		
V	V		

“Pregrado”	"2"	continuar introduciendo caracteres.	Aceptar y Cancelar. Se llenan los campos con datos incorrectos para crear un nodo activo. -Selecciona el botón Aceptar y muestran los mensajes de errores correspondientes.
V	V		
“Pregrado”	"Insertar más de 10 caracteres".		
V	V		
“Pregrado”	"2"		
V	V	El sistema muestra un mensaje de error sobre el campo requerido: “Entre al menos 2 caracteres”	
“Pregrado”	"2"		
V	I	El sistema muestra un mensaje de error sobre el campo requerido: “Entre solo números”	
“Pregrado”	"2GFA"		
V	V		
“Pregrado”	"2"		
V	V	El sistema muestra un mensaje de error sobre el campo requerido: “Ha excedido el número de letras permitidas para una palabra”.	
“Pregrado”	"2"		
V	V		
“Pregrado”	"2"		
NA	NA	El sistema muestra un mensaje de advertencia: “*¿Está seguro de realizar la acción?”,* se selecciona el botón Aceptar y regresa a la página principal donde se muestra el árbol de nodos activos.	El usuario una vez autenticado selecciona el componente “Configuración” del Sistema de Gestión Universitaria, luego selecciona de la agrupación funcional “General” la funcionalidad "Nodo activo". En el área de íconos flotantes selecciona “Crear”, introduce datos si desea o no y selecciona el botón “Cancelar”, el sistema muestra el mensaje “¿Está seguro de realizar la acción?”, se selecciona el botón Aceptar y regresa a la página principal donde se muestra el árbol de los nodos activos.

Para obtener la calidad del funcionamiento de los requisitos funcionales se llevaron a cabo tres iteraciones, las cuales arrojaron los resultados que se muestra en la tabla 11. Las no conformidades encontradas fueron resueltas en su totalidad.

Tabla 11: Iteraciones de las pruebas de pruebas de validación

Iteración	Cantidad de no conformidades	Asociadas a
Primera	18	Errores de interfaz, de validación y ortografía.
Segunda	6	Errores de interfaz y de validación
Tercera	0	-

3.5.5 Pruebas de aceptación

Las pruebas de aceptación son básicamente pruebas funcionales sobre el sistema completo, que tienen el objetivo de comprobar que se satisfacen los requisitos establecidos. Su ejecución es facultativa del cliente y en el caso de que no se realicen explícitamente, se dan por incluidas dentro de las pruebas de sistema. Las pruebas de aceptación son, a menudo, responsabilidad del usuario o del cliente, aunque cualquier persona involucrada en el negocio puede realizarlas. Estas se realizan a través de las técnicas alfa y beta. La técnica alfa es realizada por el usuario con el desarrollador como observador en un entorno controlado (simulación de un entorno de producción) y la beta es realizada por el usuario en su entorno de trabajo y sin observadores. Para realizar las pruebas de aceptación a la solución se aplicaron ambos tipos, tomando como referencia la especificación de requisitos, comprobando que el sistema cumple satisfactoriamente los requisitos del cliente y garantizando evaluar el grado de calidad del *software*.

3.5 Conclusiones parciales

La aplicación de los estándares de codificación, mejoró la forma de programar y conservar el código fuente entendible. El manejo de los paradigmas de programación establecidos en el marco de trabajo GUUD, permitió establecer un método por el cual guiarse para programar y organizar el desarrollo de la solución. Tras efectuar todas las pruebas previstas y haber alcanzado el 100% de las pruebas con resultado satisfactorio, se concluyó que la solución cumple con las especificaciones dadas por el cliente, pues se logró los objetivos propuestos. Además, se demostró que las funciones implementadas producen un resultado satisfactorio en el componente Configuración en su versión 2.0.

Conclusiones generales.

Como resultado de la investigación se arribó a las siguientes conclusiones:

- El estudio de diferentes sistemas que generalicen las configuraciones de sus componentes, permitió identificar nuevas funcionalidades que contribuyeron al desarrollo de la solución, aunque estos se ajustan a las necesidades y características de las instituciones para las que fueron desarrolladas.
- La aplicación de la metodología de desarrollo de *software* DAC, facilitó la obtención y especificación de los requisitos, así como elaboración de los artefactos propuestos y el diseño de la solución.
- Con el desarrollo de la versión 2.0 del componente Configuración del SGU, se logró gestionar los nodos, crear entidades de forma común para todos los sistemas, definir la división administrativa de un país y las acciones cada un período de tiempo de forma automática.
- Una vez realizadas las pruebas de *software* al componente Configuración 2.0 se logró garantizar el cumplimiento de los requisitos establecidos y satisfacer las necesidades del cliente.

Recomendaciones.

Tras haber finalizado el desarrollo del componente Configuración en su versión 2.0 se recomienda:

- La realización de una herramienta de instalación que automatice el proceso de despliegue del SGU a partir de la generalización de las configuraciones que brinda la solución.
- Adicionar funcionalidades a la solución para graficar la composición de los nodos que integran el sistema.

Bibliografía referenciada

1. Definiciones ABC2. *Definiciones ABC.* [En línea] http://www.definicionabc.com/tecnologia/configuracion.php.

2. DIN-DD Departamento de desarrollo. [En línea] http://gespro.din.prod.uci.cu/.

3. Diccionario de la Real Academia Española. *Diccionario de la Real Academia Española.* [En línea] 2015. http://lema.rae.es/drae/?val=configuraci%C3%B3n.

4. Diccionario de la Real Academia Española (tarea). *Diccionario de la Real Academia Española.* [En línea] 2015. http://lema.rae.es/drae/?val=tarea.

5. Diccionario de la Real Academia Española(programar). *Diccionario de la Real Academia Española.* [En línea] 2015. http://lema.rae.es/drae/?val=generalizar.

6. Diccionario de la Real Academia Española(generalizar). *Diccionario de la Real Academia Española.* [En línea] http://lema.rae.es/drae/?val=generalizar.

7. Definiciones ABC(generalización). *Definiciones ABC.* [En línea] 2007. http://www.definicionabc.com/general/generalizacion.php.

8. Diccionario de la Real Academia Española(dependencia). *Diccionario de la Real Academia Española.* [En línea] 2015. http://lema.rae.es/drae/?val=dependencia.

9. Diccionario de la Real Academia Española(funcional). *Diccionario de la Real Academia Española.* [En línea] 2015. http://lema.rae.es/drae/?val=fncional.

10. Sánchez Méndez, Alelí. *Proceso de desarrollo de software. Metodología DAC. Departamento de Desarrollo de la Dirección de Informatización, Universidad de las Ciencias Informáticas.* 2013.

11. W3C. . *W3C HTML.* [En línea] 2013. http://www.w3.org/html/..

12. The PHP Group. *The PHP Group. php manual.* [En línea] 2013. http://us.php.net/manual/en/preface.php..

13. Maestros del sitio web. *WEB ¿Qué es JavaScript?* [En línea] 2007. http://www.maestrosdel*web*.com/editorial/%C2%BFque-es-javascript/.

14. *Libross* web*. 2014 Introducción a CSS.* 2014.

15. Craig, Larman. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall, egunda edición. Segunda edición.

16. Montalvo, Marlene Melián. Biblioteca Virtual de las Ciencias en Cuba. XML el nuevo lenguaje universal. *XML el nuevo lenguaje universal.* [En línea] 2005-2008. http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASH0104/f016d031.dir/doc.pdf.

17. Definición de SQL. *Definición de SQL.* [En línea] 2008. http://definicion.de/sql/.

18. Corporation, Oracle. NetBeans IDE. *NetBeans IDE. NetBeans IDE.* [En línea] 2013. https://netbeans.org/features/php/.

19. PostgreSQL administration and management tools. [En línea] 2013. http://www.pgadmin.org/.

20. The apache *software* fundation. [En línea] 1999. http://www.apache.org/.

21. Home - Pencil Project. *Home - Pencil Project.* [En línea] 2012. http://pencil.evolus.vn/. .

22. PARADIGM, V. UML tool, business process modeler and database designer for *software* development team. [En línea] 2013. http://www.visual-paradigm.com.

23. GNU Linux Android, *software* libre, tecnología y mucho más. [En línea] 2010. www.nosolounix.com.

24. The PostgreSQL Global Development Group. *PostgreSQL.* [En línea] 2014. http://www.postgresql.org/..

25. Modelo de dominio. [En línea] 2011. http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/. .

26. Sommerville, Ian. *Ingeniería de Software. Séptima edición, Parte II, Pág 111.* 2005.

27. Pressman, Roger S. *Pressman_Cap_10_Diseno_Arquitectonico_Parte_1.* Sexta Edición.

28. EcuRed Cliente-Servidor. [En línea] http://www.ecured.cu/index.php/Cliente-Servidor.

29. Miguel Angel Alvarez. DesarrolloWeb.com. DesarrolloWeb*.com.* [En línea] 2009. http://www.desarrolloWeb.com/wiki/mvc-modelo-vista-controlador.html.

30. Welicki, León. Microsoft Developer Network. Patrones y antipatrones: una Introducción. [En línea] 2014. http://msdn.microsoft.com/es-es/library/bb972251.aspx#M19..

31. GAMMA, E, HELM, R. y JOHNSON, R. *Patrones de diseño.* 2000.

32. Grosso, Andrés. Prácticas de Software: Patrones Grasp. [En línea] 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/..

33. EcuRed. Conocimiento para todos. [En línea] http://www.ecured.cu/index.php/.

34. Microsoft. Developer Network. *Microsoft. Developer Network.* [En línea] https://msdn.microsoft.com/es-es/library/bb972272.aspx#EEAA.

35. EcuRed(patrones de Base de datos). *EcuRed.* [En línea] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

36. Alvarado, José Manuel. *Bases de datos. Unidad 2. Modelo de datos. Universidad Nacional de Ingeniería.*

37. Stallings, William. *Comunicaciones y redes de computadoras. 6ta Edición. .*

38. Paradigmas de programación. Instituto Tecnológico de Celaya. *Paradigmas de programación. Instituto Tecnológico de Celaya.* [En línea] http://itcelaya.edu.mx/.

39. Wang., Paul S. *Java con Programación orientada a Objetos.* 2000.

40. Pavón, Santiago. *Programación Orientada a Eventos.* 2012.

41. Calleja., Manuel Arias. *Estándares de codificación.*

42. Pressman, Roger S. *Infgeniería de Software. Un enfoque práctico. Capítulo 10 Ingeniería de sistema VALIDACION DE REQUISITOS, 174.* Quinta edición.

43. Marco de Desarrollo de la Junta de Andalucía. *Marco de Desarrollo de la Junta de Andalucía.* [En línea] http://www.juntadeandalucia.es/servicios/madeja/sites/default/files/historico/1.4.0/contenido-recurso-419.html.

44. Pressman, Roger S. *Ingeniería de Software. Capítulo 18 Estratégia de prueba de software. PRUEBA DE UNIDAD, 310.*

45. —. *Ingeniería de Software. Capítulo 18 Estratégia de prueba de software. Prueba de integración. Pág 312.*

46. —. *Ingeniería de Software. Capítulo 18 Estratégia de prueba de software. Prueba del sistema. Pag 317.*

47. —. *Ingeniería de Software. Capítulo 18 Estratégia de prueba de software. Pruebas de resistencia. Pag 318.*

48. —. *Ingeniería de Software. Capítulo 18 Estratégia de prueba de software. Prueba de rendimiento. Pag 318.*

49. —. *Ingeniería de Software. Capítulo 18 Estratégia de prueba de software. Pruebas de validación. Pág 316.*

50. —. *Ingeniería de Software. Capítulo17 Técnicas de pruebas de software. Prueba de caja negra. Pág 294.*

51. —. *Ingeniería de Software. Capítulo17 Técnicas de pruebas de software. Prueba de caja negra.Partición equivalente. Pág 296.*

52. *Ingeniería de Software. Capítulo17 Técnicas de pruebas de software. Prueba de caja negra.Partición equivalente. Pág 296.*

Bibliografía consultada

1. WordReference.com. [En línea] http://www.wordreference.com.
2. Definiciones ABC2. *Definiciones ABC.* [En línea] http://www.definicionabc.com/.
3. Dr. Hugo Díaz Oyarzún. Divulgación y difusión científica. [En línea] http://www.cienciateca.com/divulgacion_hugodiaz.html.
4. Solución XAVIA PACS-RIS. 2014. Solución XAVIA PACS-RIS. 2014.
5. Manual de Instalación. Internos 2.0. Centro de desarrollo. Geoinformática y señales digitales.
6. Torres Rodríguez, Frank. Marín Abreu, Ángel Dayan. Betancourt Rodríguez, Iván. Alonso Guerrero, Zorilin. Sosa Vázquez, Sisley. Fuentes Hernández, Yoendry. PLATAFORMA VIDEOWEB. TELEVISIÓN EN LA *WEB*. XVII Fórum de ciencia y técnica.
7. Drupal- Drupal™. Drupal™ https://www.drupal.org/drupal-7.0/es.
8. Apache JMeter - Apache JMeter™. 2013. Apache JMeter - Apache JMeter™. *Apache JMeter - Apache JMeter™.* [En línea] 2013. http://jmeter.apache.org/index.html.
9. Ciberaula. 2010. Ciberaula. *Una Introducción a Apache.* [En línea] 2010. Una Introducción a Apache.
10. Corporation, Oracle. 2013. NetBeans IDE. *NetBeans IDE.* [En línea] 2013.
11. Craig, Larman. 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* [ed.] Prentice Hall. Segunda edición. 2003. 970-17-0261-1.
12. Grau, Ferré, Xavier y Sánchez Segura, María Isabel. 2001. *Desarrollo orientado a objetos con UML.* [ed.] Tercera. 2001.
13. Home - Pencil Project. 2012. Home - Pencil Project. *Home - Pencil Project.* [En línea] 2012. http://pencil.evolus.vn/.
14. PostgreSQL administration and management tools. 2013. pgAdmin: PostgreSQL administration and management tools. [En línea] 2013. http://www.pgadmin.org/.
15. Roger S. Pressman. 2005. *Ingeniería del software: Un Enfoque Práctico.* 6ta. s.l.: Mc Graw Hill, 2005. pág. 980.
16. The PHP Group. 2013. The PHP Group. *Php manual.* [En línea] 2013. http://us.php.net/manual/en/preface.php.
17. The PostgreSQL Global Development Group. 2014. PostgreSQL. *PostgreSQL.* [En línea] 2014. http://www.postgresql.org/.

18. UCI. 2014. Universidad de las Ciencias Informáticas. *Misión.* [En línea] 2015. http://www.uci.cu/mision.
19. *WEB*, MAESTROS DEL. 2007. ¿Qué es JavaScript? [En línea] 2007. [Citado el: febrero 7, 2015.] http://www.maestrosdel*web*.com/editorial/%C2%BFque-es-javascript/.
20. RODRÍGUEZ, LILIANA "HACKEO ÉTICO Y SUS PRINCIPALES METODOLOGÍAS".
21. Rodríguez Gómez, David. Valldeoriola Roquet, 2009 Julio. Jordi. Métodos y técnicas de investigación en línea.
22. González Castellanos, Roberto A. 2003, Diciembre. Metodología de la investigación científica. Parte I.
23. González Castellanos, Roberto A. 2003, Diciembre. Metodología de la investigación científica. Parte II.
24. Sánchez Méndez, Alelí. *Proceso de desarrollo de software. Metodología DAC. Departamento de Desarrollo de la Dirección de Informatización, Universidad de las Ciencias Informáticas.* 2013.
25. W3C. . *W3C HTML.* [En línea] 2013. http://www.w3.org/html/.
26. The PHP Group. The PHP Group. php manual. [En línea] 2013. http://us.php.net/manual/en/preface.php.
27. Maestros del sitio web. WEB ¿Qué es JavaScript? [En línea] 2007. http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.
28. Libross web. 2014 Introducción a CSS. 2014.
29. Montalvo, Marlene Melián. Biblioteca Virtual de las Ciencias en Cuba. XML el nuevo lenguaje universal. XML el nuevo lenguaje universal. [En línea] 2005-2008. http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASH0104/f016d031.dir/doc.pdf.
30. Definición de SQL. Definición de SQL. [En línea] 2008. http://definicion.de/sql/.
31. 18. Corporation, Oracle. NetBeans IDE. NetBeans IDE. NetBeans IDE. [En línea] 2013. https://netbeans.org/features/php/.
32. PostgreSQL administration and management tools. [En línea] 2013. http://www.pgadmin.org/.
33. The apache software fundation. [En línea] 1999. http://www.apache.org/.
34. Home - Pencil Project. Home - Pencil Project. [En línea] 2012. http://pencil.evolus.vn/.
35. PARADIGM, V. UML tool, business process modeler and database designer for software development team. [En línea] 2013. http://www.visual-paradigm.com.

36. GNU Linux Android, software libre, tecnología y mucho más. [En línea] 2010. www.nosolounix.com.
37. The PostgreSQL Global Development Group. PostgreSQL. [En línea] 2014. http://www.postgresql.org/.
38. Modelo de dominio. [En línea] 2011. http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.
39. Sommerville, Ian. Ingeniería de Software. Séptima edición, Parte II, Pág 111. 2005.
40. Pressman, Roger S. Pressman_Cap_10_Diseno_Arquitectonico_Parte_1. Sexta Edición.
41. Paradigmas de programación. Instituto Tecnológico de Celaya. Paradigmas de programación. Instituto Tecnológico de Celaya. [En línea] http://itcelaya.edu.mx/.
42. Wang., Paul S. Java con Programación orientada a Objetos. 2000.
43. Pavón, Santiago. Programación Orientada a Eventos. 2012.
44. Calleja., Manuel Arias. Estándares de codificación.