Universidad de las Ciencias Informáticas

Facultad 3

Título: Implementación del módulo Dashboard versión 1.0 del Sistema de Gestión Integral de Aduana.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Félix Manuel Ruibal Berenguer

Tutores: Ing. Raymond Weeden Gamboa

Ing. Adrián Naranjo García

Junio 2015

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Félix Manuel Ruibal Berenguer Ing. Raymond Weeden Gamboa

_____

Ing. Adrián Naranjo García

DATOS DE CONTACTO

Ing. Raymond Weeden Gamboa. Ingeniero en Ciencias Informáticas. Graduado en 2009. 4 Años de experiencia en el tema. Profesor Instructor. Correo rweeden@uci.cu
Ing. Adrian Naranjo García. Ingeniero en Ciencias Informáticas. Graduado en 2010. 4 Años de experiencia en el tema. Correo angarcia@uci.cu

AGRADECIMIENTOS

A mi madre por ser mi guía y apoyo en estos años de universidad, por los ánimos que me ha dado para que pueda completar esta carrera.

A mi abuela por estar siempre al tanto de mi progreso como estudiante y persona.

A mi abuelo por ser un ejemplo de profesional y persona a seguir.

A toda mi familia por creer en mí y mi sueño de ser ingeniero.

A Jorge Ariel y Ernesto que han sido como los hermanos que nunca tuve.

A mis amigos que me apoyaron todo este tiempo, Araí, Thais, los jimas, Odalys, Negrín, Aniel y el resto de mis compañeros de grupo que de una forma y otra compartimos estos años de universidad.

A mis tutores por guiarme en la realización de este trabajo y por esas madrugadas en el laboratorio programando.

A todos los profesores que de una forma u otra han contribuido a mi formación como profesional.

DEDICATORIA

Este trabajo está dedicado a la memoria de mi padre quien siempre me alentó a que me formara profesionalmente y pudiera ser el apoyo de mi familia.

A mi mamá por todo el amor que me ha brindado y sacrificarse por que su hijo estudiara una carrera universitaria.

A mis abuelos por todo el apoyo y amor que me han brindado.

A toda mi familia por apoyarme siempre en las buenas y las malas.

RESUMEN

Llevar el control sobre las operaciones que se realizan sobre el Sistema de Gestión Integral de Aduanas por parte de los trabajadores de la Aduana General de la República de Cuba, es una tarea difícil para los funcionarios de esta. La presente investigación tiene como objetivo fundamental concebir una solución informática que permita el monitoreo de las acciones realizadas sobre el sistema.

Para el desarrollo de la solución informática propuesta anteriormente, se realizó un estudio previo de distintos Dashboard, analizando sus principales características. Posteriormente se llevó a cabo un estudio de las metodologías y herramientas que fueron usadas para concebir la solución. Con el uso del modelo de desarrollo Proceso Unificado Ágil adaptado a la Universidad de las Ciencias Informáticas en las etapas de análisis, diseño, implementación y validación, se pudo obtener un módulo perteneciente al sistema de Gestión Integral de Aduanas, el cual es capaz de monitorear el uso del sistema por parte de los usuarios y facilitar la toma de decisiones por parte de los directivos.

PALABRAS CLAVE

Dashboard, control, monitoreo, Aduana

ABSTRACT

It is a very difficult task to perform the control over the operations executed onto the Integrated Management System of Customs by the workers of the General Customs of the Republic of Cuba. That's why this paper is aimed to devise an informatics solution able to monitor such type of operations.

In order to accomplish that goal, there was performed a study about the different existing Dashboards to analyse their main features. Following to that, a study on development methodologies and tools to build the solution was held. So, with the use of the Agile Unified Process adapted to the Informatics Sciences University on the analysis, design, implementation and validation, a new module was obtained related to the Integrated Management System of Customs. This module is capable of monitoring the interaction of the users in the system to ease the managers' decision-making process.

KEYWORDS

Dashboard, control, monitoring, Customs

TABLA DE CONTENIDOS

INTRODUCCIÓN

En la actualidad, se ha vuelto indispensable el uso de las Tecnologías de la Información y las Comunicaciones (TIC). El constante progreso y la gran aceptación de estas tecnologías se deben a los múltiples y convenientes beneficios que brindan. Debido al gran volumen de datos que estas generan, el diseño de nuevas herramientas para el tratamiento de la información que tributen a la toma de decisiones en todos los niveles, se ha convertido en un objetivo primordial para la gestión de la información.

La gestión empresarial, encargada de la *planificación, organización, dirección y control* de los recursos de una organización con el fin de obtener el máximo beneficio posible, está potenciada por una gran cantidad de herramientas informáticas que facilitan el logro de sus objetivos. El software de gestión empresarial ha surgido para impulsar la incorporación de las TIC en los procesos de trabajo de las empresas, posibilitando la toma de decisiones rápida y segura, acortando los ciclos productivos e incrementando la calidad de los servicios y productos.

Los sistemas de gestión empresarial, como los de Administración de Relaciones con los Clientes (CRM[1]) y los de Planificación de Recursos Empresariales (ERP[2]), facilitan las estrategias de negocio de todo tipo de empresas. Los CRM implementan un modelo de negocios cuya estrategia está destinada a identificar y mantener las relaciones con los clientes, manteniendo su continuo valor agregado. Los ERP son soluciones informáticas cuyo objetivo es gestionar la información a través de las diferentes áreas para agilizar las tareas, mejorar los procesos de producción y reducir costos (1).

En el entorno competitivo actual, el análisis de información para predecir las tendencias del mercado y para mejorar el rendimiento de la empresa es una actividad esencial del negocio. Sin embargo, es cada vez más claro que el éxito empresarial requiere de un análisis de datos y respuestas inmediatas con el fin de cumplir con los rápidos cambios en la demanda de los clientes.

Las herramientas de análisis y reportes surgen a principios de los años 90, creándose la tendencia de generar reportes personalizados, sin depender exclusivamente de los departamentos de sistemas. La combinación de estas herramientas con las bases de datos, clamaron el arribo de la era del “autoservicio” por los proveedores de software para la Inteligencia de Negocios (2).

El contexto que enfrentaron muchos de los usuarios se caracterizó por un entorno con herramientas muy complejas, donde un solo reporte o varios presentados de forma desagregada no contribuían con el manejo de la información de una manera eficiente para la toma de decisiones estratégicas.

---

[1] *Customer Relationship Management.*

[2] *Enterprise Resource Planning*

Como solución, aparecieron las herramientas informáticas conocidas como dashboard[3]. Estas herramientas muestran la información más importante, de tal forma que permita hacer un seguimiento de lo que está ocurriendo en un instante. La mayoría de los dashboard que se utilizan en los negocios de hoy están muy por debajo de su potencial. Para servir a su propósito y maximizar sus prestaciones, los dashboard deben mostrar una densa red de información en una pequeña cantidad de espacio de manera que se comunique con claridad e inmediatez. Esto requiere un diseño que se nutra y aproveche el poder de la percepción visual para lograr el procesamiento de grandes cúmulos de información (3).

El uso de los dashboard unido a los ERP o CRM crea una ventaja visual, que facilita el acceso, extracción y análisis rápido de los datos. Los mismos otorgan una nueva respuesta a la antigua necesidad de reportes en el software de gestión empresarial por tratarse de una solución rápida y sencilla que ofrece gran atractivo visual. Los ejecutivos pueden explorar la información de manera rápida y fácil entre cantidades gigantescas de datos lo cual les permite tomar mejores decisiones, ahorrar tiempo en comparación a la ejecución de varios informes, la identificación rápida de los valores extremos, la creación de informaciones visuales, gráficas en pequeños espacios de tiempo y nuevas series analíticas como presupuestos, pronósticos y otras (4).

Por otra parte, la web en tiempo real es un paradigma basado en dar información a los usuarios tan pronto como esté disponible, en lugar de exigir que ellos o sus programas verifiquen la fuente de los datos periódicamente en busca de actualizaciones. Se puede activar de muchas maneras y pueden requerir de una arquitectura técnica variable. Los sistemas en tiempo real están siendo implementados en las redes sociales, búsqueda, noticias y otros sitios, logrando experiencias parecidas a la mensajería instantánea y facilitando innovaciones impredecibles. Los primeros beneficios incluyen una mayor participación del usuario y la disminución de las cargas del servidor; la web en tiempo real puede llegar a convertirse en omnipresente, un requisito para casi cualquier sitio web o servicio (5).

Cuba, dada la necesidad de una independencia tecnológica y la importancia que tiene el desarrollo de sistemas informáticos ha empezado a desarrollar sus propios sistemas web. Con el objetivo de cumplir esta tarea surge la Universidad de las Ciencias Informáticas (UCI), que desde sus inicios desarrolla productos con el fin de informatizar los sectores de la sociedad.

La UCI cuenta con un Centro para la Informatización de Entidades (CEIGE), donde se desarrollan productos para la gestión empresarial. El análisis de información que se realiza se exporta al usuario en

[3] Herramientas informáticas que muestran información de forma gráfica y ordenada en tiempo real.

forma de reportes electrónicos en el Formato de Documentos Portables (PDF[4]), muy pobres en información visual, poco personalizables y referidos a un estado particular del sistema.
El Sistema de Gestión Integral de Aduanas (GINA), desarrollado por el centro CEIGE, no permite controlar cuántos usuarios utilizan el sistema, desde donde acceden o en que módulo o aplicación se está trabajando. Tampoco permite conocer el comportamiento del consumo de los servicios o controlar cuando se añaden los datos de una nueva persona, lo que dificulta en alguna medida la toma de decisiones.

Según lo descrito anteriormente se define el **problema de la investigación** de la siguiente manera: ¿Cómo mostrar gráficamente y en tiempo real los procesos fundamentales en el sistema GINA de la Aduana General de la República (AGR)?

El **objeto de estudio** se enmarca en el monitoreo de sistemas de gestión, mientras que el **campo de acción** se centra en el monitoreo del sistema GINA de la AGR.

Se plantea entonces como **objetivo general del trabajo**: desarrollar el módulo Dashboard que permita la administración, configuración y visualización en tiempo real de la información generada por las aplicaciones desarrolladas sobre el GINA.
Para dar cumplimiento al objetivo general se plantean los siguientes **objetivos específicos**:

1. Elaborar el marco teórico relativo a los dashboard para obtener un estado del arte que permita establecer el precedente de la investigación.
2. Realizar la ingeniería de requisitos para garantizar el diseño del módulo Dashboard.
3. Diseñar la solución modelada de manera que cumpla con las necesidades del cliente.
4. Implementar la solución diseñada de manera que cumpla con las necesidades del cliente.
5. Validar la propuesta de solución en función de corroborar su correspondencia con los requisitos del cliente y el cumplimiento del objetivo general de la investigación.

[4] Portable Document Format

El presente trabajo tiene como **pregunta de investigación:** ¿Se facilitaría el proceso de toma de decisiones sobre el GINA si se desarrolla una herramienta para mostrar gráficamente y en tiempo real los procesos fundamentales del sistema?

Los métodos de investigación utilizados fueron:

Métodos Teóricos

- **Histórico lógico:** se utiliza para realizar una revisión histórica del desarrollo de los dashboard y constatar teóricamente cómo ha evolucionado la administración y configuración de la información que estos muestran.
- **Analítico-sintético:** permitió el procesamiento de la información y arribar a las conclusiones prácticas y teóricas de la investigación.

Métodos Empíricos

- **Experimento:** favoreciendo el desarrollo de pruebas para la verificación de las funcionalidades implementadas, con el fin de detectar errores y comprobar su correcto funcionamiento.
- **Observación:** se utiliza para realizar una evaluación de la situación problemática en cuestión.

Estructuración del trabajo.

El trabajo consta de tres capítulos que cubren la fundamentación teórica, el análisis y diseño del sistema e implementación y pruebas, además de las conclusiones, recomendaciones, bibliografía y el glosario de términos. A continuación un resumen de cada capítulo:

Capítulo1: Fundamentación teórica: en este capítulo se explican las metodologías, los lenguajes usados, las herramientas utilizadas para el desarrollo de la aplicación y una breve panorámica sobre los Dashboard.

Capítulo2: Análisis y diseño del sistema: en este capítulo se describen las características del módulo Dashboard y detalles relacionados con el diseño.

Capítulo3: Implementación y prueba: en este capítulo se describe cómo se lleva a cabo la implementación del sistema, características esenciales así como el aporte práctico del módulo Dashboard. Se explica cómo se realiza el tratamiento de errores y las pruebas de validación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN.

En el presente capítulo se definen una serie de conceptos necesarios para entender el objetivo fundamental del trabajo, se detallan las tendencias y tecnologías actuales, metodologías y herramientas para el desarrollo de la solución propuesta. Además se profundizan en las características de los dashboard actuales y los sistemas de monitoreo en tiempo real.

1.2. ESTADO DE ARTE

1.2.1. DASHBOARD

En español no existe una traducción única del término dashboard, al traducir dashboard al idioma español obtenemos una variedad de posibles sinónimos, entre los más comunes encontramos: cuadro de mando, panel o tablero de instrumentos, panel o tablero de control. Para evitar ambigüedades se opta por utilizar el término en inglés, el cual es utilizado en la mayoría de la bibliografía consultada.

Los dashboard están diseñados para ayudar a interpretar el estado instantáneo de un elemento de control. Para proveer de un medio que permita controlar con rapidez el estado de cualquier sistema. (6).

Según Stephen Few, presidente de Perceptual Edge y experto en visualización, un dashboard se puede definir como “Una presentación visual de información importante, necesaria para lograr uno o más objetivos; consolidada y arreglada en una sola pantalla, de tal manera que la información pueda monitorearse con un vistazo.” (6)

En los sistemas de gestión de la información, un dashboard es un sistema de información ejecutivo, con una interfaz de usuario diseñada para ser fácil de interpretar.

1.2.2. ANÁLISIS BIBLIOMÉTRICO DOCUMENTAL

Tabla 1. Análisis bibliométrico.

	Últimos 5 Años	Años Anteriores
Libros y Monografías	10	11
Tesis de Doctorado	0	0
Tesis de Maestría	0	0
Artículos de Revistas referenciadas en Web of Science, SCOPUS	0	0
Memorias de Eventos	0	0
Artículos publicados en la web	9	8
Reportes técnicos y conferencias	3	0
Entrevistas personales	0	0
Trabajo de Diploma	1	0
Total	23	19

1.2.3. SISTEMAS DE TIEMPO REAL

Existen muchas interpretaciones de la naturaleza exacta de un Sistema en Tiempo Real (STR), sin embargo, todos tienen en común la noción de tiempo de respuesta.

El Diccionario Oxford de Computación ofrece la siguiente definición de sistema en tiempo real:

Un sistema en tiempo real es cualquier sistema donde el tiempo en que se produce su salida no es significativo. Por lo general debido a que la entrada corresponde a algún instante del mundo físico y la salida tiene relación con ese mismo instante. El retraso transcurrido entre la entrada y la salida debe ser lo suficientemente pequeño para considerarse una respuesta puntual (7).

Otra definición de sistema de tiempo real dada por Stephen J. Young:

Cualquier actividad o sistema de procesamiento de información, que responda a un estímulo externo dentro de un período determinado y finito (8).

El proyecto PDCS [5] el cual tiene como objetivo contribuir al proceso de diseño y construcción de sistemas confiables de computación mucho más previsibles y rentables, brinda la siguiente definición:

Un sistema de tiempo real es un sistema que se requiere para reaccionar a los estímulos del entorno, dentro de los intervalos de tiempo determinado por el entorno (9)*.*

Las acciones de los Sistemas de Tiempo Real se producen dentro de unos intervalos de tiempo determinados por la dinámica del sistema físico que supervisan o controlan. En su sentido más general, todas estas definiciones abarcan una gama muy amplia de actividades informáticas.

Los sistemas de tiempo real pueden ser divididos en dos grupos:

Hard (Duro): Son aquellos en los que es absolutamente imperativo, que las respuestas se produzcan dentro del plazo especificado. Una respuesta tardía puede tener consecuencias fatales.
Soft (Suave): Son aquellos con restricciones de tiempo en las que una respuesta tardía no produce graves daños, el sistema seguirá funcionando correctamente (9).

En el presente trabajo de diploma cuando se usa el término “sistema de tiempo real” se refiere al sistema de tiempo real suave. Ya que es el término que se corresponde con las características del dashboard a desarrollar.

En algunas ocasiones podemos ver referencias sobre sistemas de tiempo real cuando sólo se quiere decir que el sistema es rápido. Cabe mencionar que **tiempo real** no es sinónimo de rapidez, esto significa que no es el tiempo de respuesta lo que nos enfoca en un sistema de tiempo real, sino asegurarse que el tiempo de respuesta esté dentro del plazo especificado para resolver el problema al que el sistema está dedicado (9).

1.3. SISTEMAS ESTUDIADOS

Actualmente son muchas las empresas que para lograr una mejor gestión de datos hacen uso de los dashboard, estos son diseñados con disímiles herramientas. Algunas de estas herramientas se seleccionan como objeto de estudio para analizar el tipo de información que manejan.

[5] Predictably Dependable Computer Systems.

1.3.1. WAOX v1.0

WaoX Dashboard es una aplicación para monitorizar la información generada por las diferentes aplicaciones del CEIGE desarrolladas con el marco de trabajo Sauxe. WaoX fue diseñado con el principio de brindar a los miembros de la organización una herramienta que facilite el seguimiento de la información generada en tiempo real, mejorando así la toma de decisiones (11).

Características

- Permite personalizar el ambiente de trabajo, permitiendo adicionar o eliminar monitores.
- Crear Fuentes de Datos.
- Permite adicionar hasta cuatro vistas por monitor.
- Comunicación a través del protocolo XMPP.
- Capacidad de exportar a XLS.

1.3.2. PENTAHO DASHBOARD

Conceptualmente, Pentaho Dashboard es una plataforma integrada para proporcionar información sobre datos, donde se puede ver todo tipo de informes, gráficos interactivos y los cubos creados con las herramientas de Pentaho como Pentaho ReportDesigner. Se trata de una interfaz que ofrece una vista centralizada sobre los movimientos de datos profesionales, lo que permite seguirlos y tomar decisiones (12).

Sistemas Operativos compatibles: Windows, Macintosh, Unix y Linux.

Características

- Pentaho Dashboards está licenciado bajo GNU General Public License Version 2 (GPLv2).
- Incluye componentes de navegación y el visor de informes y análisis, que pueden ser integrados en los portales o páginas web.
- Genera contenido XML.
- Totalmente personalizable mediante definiciones XML de diseño, y hojas de estilo XSL y CSS.

- Identificación de métricas clave (KPIs, Key Performance Indicators), mediante la generación de Monitoreo/Métricas.
- Realización de investigaciones de detalles subyacentes, con reportes de soportes.
- Ejecución de seguimientos de excepciones, permitiendo pre-establecer alertas basadas en reglas del negocio.
- Gran variedad gráficos, tablas y velocímetros.
- Análisis OLAP.
- Capacidad de exportar a HTML, PDF, XLS, DOC, TIFF, CSV, XML.

1.3.3. OPEN REPORTS

Open Reports es una poderosa y flexible solución de reportes web de código abierto bajo licencia GPL. Ofrece la generación de informes dinámicos y capacidades de programación de informes. Permite la creación de gráficas a través de ChartReports (13).

ChartReports utiliza JFreeCharts, un paquete abierto de gráficos, para proporcionar la capacidad de generar de forma dinámica gráficos sin la necesidad de escribir ningún código. Con el fin de crear un gráfico, ChartReport debe crear una definición de tabla y luego agregar la tabla y los parámetros de consulta de gráficos para el informe (13).

Características:

- Crear gráficos a partir de varios informes.
- Permite consultas SQL para obtener los datos a mostrar.
- Soporta 5 tipos de gráficos, Barras, Pastel, Anillo, Tiempo y Línea.
- Paneles colapsables de Alerta y Reportes.

- Soporte para una amplia variedad de formatos de exportación incluyendo PDF, HTML, CSV, XLS, RTF e imagen.

1.3.4. IBM COGNOS 8 GO! DASHBOARD

El IBM Cognos 8 es una plataforma que sustenta una amplia gama de capacidades de gestión de rendimiento, tales como informes, dashboard y planificación para proporcionar información completa y oportuna a su comunidad de usuarios (14).

IBM Cognos 8 Go! Dashboard integrado en la plataforma Cognos 8, traduce la información y datos en forma visual, presentaciones sofisticadas utilizando indicadores, mapas, gráficos y otros elementos para mostrar los resultados. Permite crear e interaccionar con dashboard complejos y confiables a través de una interfaz dinámica basada en flash. Se extiende más allá de solo medidores y gráficos al integrar IBM Cognos-Built, elementos externos como canales RSS[6], páginas web y facilidades de búsqueda. Disponible para Linux y Windows (15)(16).

Características.

- Datos y gestión del ciclo de vida: El modo Interactivo y Ensamblado permite añadir extensiones IBM Cognos y organizar las conexiones a los datos en un solo lugar para la fácil administración.
- Brinda conexión en directo con fuentes confiables de datos corporativos sujetos a la gestión habilitada por la plataforma de seguridad, así como su integración con servicios web.

IBM Cognos 8 Go! Dashboard funciona en dos modos básicos Ensamblaje e Interactivo:

Ensamblaje.

- Crear y editar gráficos dinámicos y paneles de instrumentos que son fáciles de configurar.
- Los usuarios de negocio o de tecnología de información pueden construir dashboard de partes pre-existentes.

[6] *Really Simple Syndication*

- Adicionar reportes, informes parciales o indicadores de medidas arrastrándolos desde el panel de contenido.
- Agregar filtros a partir de menús pre-definidos a los elementos del dashboard (reportes, etc.) para mostrar exactamente la información necesaria.

Interactivo.

- Cualquier usuario a cualquier nivel puede visualizar e interactuar con el dashboard.
- Modificar el diseño gráfico de un reporte o informe.
- Ordenar y filtrar los datos del dashboard para ajustarlos a los requisitos necesarios.

1.3.5. SAS BI DASHBOARD

SAS[7] BI Dashboard permite a los usuarios monitorizar los indicadores claves de rendimiento. Los dashboard pueden incluir gráficos, texto, colores e hipervínculos. Son creados, modificados y visualizados a través de una fácil interfaz basada en web. Todo el contenido se muestra en un entorno basado en roles, es seguro, personalizable y extensible (17).

Características.

SAS BI Dashboard Designer.

- Una nueva interfaz interactiva e intuitiva. Permite arrastrar y soltar objetos sobre el dashboard que se está diseñando.
- Puede cambiar el tamaño de los indicadores con el ratón. Las nuevas características ayudan a trazar el contenido del panel con precisión.
- El diseñador trabaja en tiempo real, mientras que se está diseñando, lo que se ve, es el producto final.

[7] Statistical Analysis System.

- El Dashboard Designer también ofrece ahora muchos más tipos de indicadores para su uso en un tablero de instrumentos, así como nuevos estilos de gráficos para mejorar el aspecto de los indicadores.

SAS BI Dashboard Viewer.

- Mediante la creación de dashboard, los diseñadores le proporcionan a los usuarios datos de su negocio. Los usuarios empresariales pueden ver estos dashboard en el SAS BI Dashboard Viewer.
- Permite compartir comentarios con otros usuarios acerca de los indicadores del dashboard, crear alertas personalizadas, y visualización de datos de una manera interactiva.
- Moverse fácilmente entre el Dashboard Designer y el Dashboard Viewer permite a los diseñadores crear un nuevo tablero de instrumentos, y sin esfuerzo verlo en el Dashboard Viewer, y volver rápidamente a la vez al Dashboard Designer. O bien, se puede utilizar la característica de vista previa y ver dashboard y la mayoría de sus características sin salir del Dashboard Designer en absoluto.
- Exportar a los formatos, Excel, PDF de Adobe.

1.3.6. WEBFOCUS

Information Builders ofrece dashboard robustos e innovadores que permiten compartir las misiones con toda la empresa, enlazar las operaciones directamente a los objetivos estratégicos, y monitorizar en tiempo real (18).

WebFOCUS dashboard, proporciona información sobre la organización y su funcionamiento, mejorando la toma de decisiones de los ejecutivos, y trabajadores de primera línea. WebFOCUS dashboard se utiliza para descubrir oportunidades, identificar las tendencias y encontrar vulnerabilidades antes de que se conviertan en problemas (18).

Características.

Permite a los usuarios finales en todos los niveles crear sus propios dashboard usando las tecnologías más innovadoras y modernas.

Las capacidades de personalización de WebFOCUS permiten:

- Ensamblaje de arrastrar y soltar.
- Soporte para Herramientas Web 2.0 de terceros.
- Difusión y sincronización de las preferencias hacia otros objetos del dashboard.
- Almacenamiento inteligente en cache de los datos del dashboard.
- Exportar en varios formatos, Excel, PDF de Adobe, Flash, Adobe Flex PDF y Microsoft PowerPoint.
- Calendario para entrega vía correo electrónico, un archivo de reporte, impresora y sitio FTP[8].
- Maximizar la comunicación de información a través del más completo conjunto de datos y visualizaciones de información en la industria. Más de 200 tipos de gráficos como Pareto, diagramas de dispersión, diagramas de constelación, las nubes de etiquetas, entre otros.
- Incluye mejoras de extracción, transformación y carga en tiempo real de datos desde bases de datos, y mapas de información, así como integración total con adaptadores de datos en tiempo real.

[8] File Transfer Protocol - Protocolo de Transferencia de Archivos

En la tabla 2 se muestra un resumen de algunas características de los sistemas estudiados.

Tabla 2. Resumen de características de los sistemas estudiados.

	SO	Licencia	Portable	Formatos	Fuentes de datos	Diseño en tiempo real
Waox	Windows, Macintosh, Unix y Linux.	Libre	Parte de un sistema	XLS	BD	Sí
PENTAHO Dashboard	Windows, Macintosh, Unix y Linux.	GPLv2	Parte de un sistema	HTML, PDF, XLS, DOC, TIFF, CSV, XML	BD, Cubo OLAP, Pentaho metadata, XML, BD	No
OpenReports	Windows	GPL	Independiente	PDF, HTML, CSV, XLS, RTF,imagen	Consultas SQL, BD	No
IBM GO	Windows, Linux	Propietaria	Parte de un sistema.	PDF, Excel, Word, Power Point	BD. Servicios	No
SAS BI	Windows, Linux	Propietaria	Independiente.	Excel, PDF de Adobe.	Mapas de información, BD	No
WebFocus	Windows	Propietaria	Independiente	Excel, PDF, Flash, PowerPoint.	BD, mapas de información	Sí

A parte de las características anteriores, también se analizaron los tipos de gráficos que muestran estos sistemas lo cual aportaría una base para la elaboración de la interfaz de una nueva aplicación. Entre las principales limitaciones presentadas por estos sistemas se encuentran:

- Formar parte de otro sistema o plataforma.
- No son soluciones de código abierto.
- No han sido desarrollados utilizando PHP, ExtJS y Oracle 11g. Tecnologías necesarias para poder incluirlo como una herramienta del GINA.

Por lo que se concluye que estos sistemas no pueden ser integrados con el GINA y no constituye una respuesta al problema planteado.

1.4. TECNOLOGÍAS UTILIZADAS

Las tecnologías son un conjunto de conocimientos técnicos que son utilizados para el desarrollo de software, que posibilitan la satisfacción y adaptación de las necesidades a las que se enfrentan los desarrolladores. El presente trabajo forma parte del proceso productivo del CEIGE, el cual ha tomado decisiones tecnológicas que involucran la utilización de tecnologías de código abierto para el desarrollo de sus productos. A continuación se describen brevemente estas tecnologías.

1.4.1. METODOLOGÍA DE DESARROLLO DE SOFTWARE

Las Metodologías de Desarrollo de Software tienen como objetivo presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desarrollar software de calidad, incluyendo heurísticas de construcción y criterios de comparación de modelos de sistemas.

Para el desarrollo de esta solución se aplicó la metodología definida por la UCI, la cual es una variación del Proceso Unificado Ágil (AUP) de Scott Ambler ya que esta se adapta al ciclo de vida productivo de la UCI. Esta variación cuenta con tres fases:

- **Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto (19).
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al

ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software (19).

- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (19).

Esta variación propone ocho disciplinas:

1. **Modelado de Negocio (Opcional):** el Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito (19).
2. **Requisitos:** el esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto (19).
3. **Análisis y Diseño:** en esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis (19).
4. **Implementación:** en la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema (19).
5. **Pruebas Internas:** en esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas (19).
6. **Pruebas de Liberación:** pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (19).
7. **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (19).

8. **Despliegue (Opcional):** Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente (19).

Productos de trabajo

Entre las técnicas ágiles que utiliza AUP se encuentra el Modelado ágil, se hará uso de está técnica para los proyectos que necesiten por sus características encapsular sus requisitos funcionales en Historias de usuarios o en Descripción de requisitos por procesos. La otra forma de encapsular los requisitos se mantiene por Casos de Uso (CU). De forma general se siguen utilizando los productos de trabajos definidos en el Expediente de proyecto, algunos son obligatorios independientemente del tipo de proyecto y otros son opcionales en base a las particularidades del mismo. Todos los productos de trabajos se encuentran en los documentos públicos ubicados en excriba.prod.uci.cu y en mejoras.prod.uci.cu. (19).

1.4.2. MARCOS DE TRABAJO Y BIBLIOTECAS

Symfony 1.2

Symfony es un marco de trabajo[9] (*framework)* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (20).

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft y se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. A continuación se muestran algunas de sus características (20):

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.

[9] Estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales, y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con bibliotecas desarrolladas por terceros (20).

Ext JS 3.0

Ext JS es una biblioteca JavaScript ligera y de alto rendimiento, que nos permite crear páginas e interfaces web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Es extensible para la gran mayoría de los navegadores, evitando el tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, la carga de procesamiento se distribuye permitiendo que el servidor, al tener menor carga, pueda manejar los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de estar sujeta a una acción del usuario, dando la libertad de cargar la información sin que este lo note (21).

Propel

Propel es un servicio de objeto persistente y de consulta, lo que significa que Propel provee un sistema para almacenar objetos en una base de datos y un sistema para la búsqueda y restauración de objetos desde una base de datos. Propel le permite realizar consultas complejas y manipulación de bases de datos sin escribir una sola cláusula SQL. Propel hace más fácil la escritura de aplicaciones, más fácil de desplegar, y mucho más fácil para migrar si alguna vez la situación lo amerita (22).

Propel puede ser descrito como un mapeado objeto-relacional, una capa DAO, o una capa objeto persistente. Propel es un puerto de Apache torque basado en acercamientos probados, desarrollado por el

proyecto Torque y optimizado para PHP. Propel espera proporcionar un inteligente y comprensivo servicio de manejo de datos con un mínimo costo de realización para su aplicación en PHP (22).

Para esos familiares con patrones O/R, Propel inicialmente implementa el patrón entrada de datos en fila (22).

Sin embargo, Propel también genera las clases para cada tabla que exhibe algunas de las propiedades de la tabla del patrón datos de entrada:

Una tabla de entrada de datos almacena todo el SQL para acceder a una sola tabla o vista: selecciones, inserciones, actualizaciones, y eliminaciones. Otro código llama los métodos para todas las interacciones con la base de datos (22).

En Propel las clases de tabla de entrada de datos son llamadas clases Peer, mientras las clases de filas de entrada de datos son llamadas entidad o clases objeto (22).

Como una aplicación, Propel tiene dos componentes principales (y ahora formalmente separados):

- Un motor generador para construir sus clases y archivos SQL (generador-propel).
- Un ambiente de ejecución que proporciona herramientas para construir consultas SQL, ejecutando consultas compiladas, y herramientas para el manejo de conexiones para múltiples bases de datos simultáneamente (propel) (22).

El ambiente de ejecución proporciona una capa de abstracciones y encapsulación de bases de datos, reglas y lógicas de negocios. Las clases Propel representan la capa modelo del tradicional MVC, diseñado para encapsular cualquier nivel de validación de dato necesitado por su aplicación (22).

1.4.3. LENGUAJES DE PROGRAMACIÓN

PHP 5.3

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje interpretado de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Sus sintaxis son muy similares a lenguajes como C y PERL. Es multiplataforma permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requisitos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores (23).

Dispone de una conexión nativa a los principales sistemas de base de datos utilizados actualmente tales como Postgres, MySQL, Oracle, Microsoft SQL Server, lo cual permite la creación de aplicaciones web robustas. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de contar con una comunidad de desarrolladores que intercambian experiencias lo que facilita la rápida solución de problemas sin costo alguno. Quizás una de sus mayores desventajas radica en que promueve la creación de código desordenado, por lo que lo hace muy complejo de mantener (23).

JavaScript

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Este es un lenguaje Case Sensitive[10]y al contrario que Java (con el cual no existe ninguna relación), no es exactamente un lenguaje orientado a objetos, puesto que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base que serían los prototipos y extendiendo su funcionalidad. Su función es ampliar las funcionalidades de HTML, permitiendo interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Mozilla Firefox, Netscape, Opera (24).

CSS

Cascade Style Sheet (CSS[11]) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas (25).

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (25).

[10] Sensible a la diferencia entre minúsculas y mayúsculas.
[11] Hojas de estilo cascada, CSS siglas en inglés.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc (25).
Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc (25).

JSON

JSON [12] es un formato ligero de intercambio de datos, un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. Es un formato de texto que es completamente independiente del lenguaje, estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos (26).

UML 2.0

El Lenguaje Unificado de Desarrollo (UML) representa un número de modelos de desarrollo basados en componentes que han sido propuestos en la industria. El proceso unificado define los componentes que se utilizarán para construir el sistema y las interfaces que conectarán los componentes. Utilizando una combinación del desarrollo incremental e iterativo, el proceso unificado define la función del sistema aplicando un enfoque basado en escenarios (desde el punto de vista del usuario) y acopla la función con un marco de trabajo arquitectónico que identifica la forma que tomará el software (27).

1.4.4. HERRAMIENTAS

A continuación se describen las herramientas utilizadas para lograr el cumplimiento del objetivo general.

Visual Paradigm 8

Visual Paradigm (VP) es una herramienta CASE[13] para el diseño UML y se encuentra diseñada para ayudar al desarrollo de software. Utiliza UML como lenguaje de modelado y soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. Ofrece un

[12] JavaScript ObjectNotation
[13] Ingeniería de Software Asistida por Computadora, CASE siglas en inglés.

completo conjunto de herramientas para equipos de desarrollo de software, necesarias para la captura de requisitos, planificación de software, el modelado de clases, modelado de datos, etc.
Permite el modelado colaborativo con Subversion y la integración de aplicaciones empresariales a las bases de datos. Es capaz de realizar ingeniería tanto directa como inversa, posibilita generar código para lenguajes como PHP. Posee un generador de informes automático para la documentación del proyecto en varios formatos como HTML, PDF. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto.

NetBeans IDE 7.4

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos. Al día de hoy hay disponibles dos productos: el NetBeans IDE y NetBeans Platform (28).

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso (28).

Subversion

Subversion es un sistema de control de versiones libre y de código fuente abierto. Es decir, Subversion maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. En este aspecto, mucha gente piensa en los sistemas de versiones como en una especie de “máquina del tiempo” (29).

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para

temer porque la calidad del mismo vaya a verse afectada por la pérdida de ese conducto único—si se ha hecho un cambio incorrecto a los datos, simplemente deshaga ese cambio (29).

Algunos sistemas de control de versiones son también sistemas de administración de configuración de software. Estos sistemas son diseñados específicamente para la administración de árboles de código fuente, y tienen muchas características que son específicas del desarrollo de software— tales como el entendimiento nativo de lenguajes de programación, o el suministro de herramientas para la construcción de software. Sin embargo, Subversion no es uno de estos sistemas. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros. Para usted, esos ficheros pueden ser código fuente— para otros, cualquier cosa desde la lista de la compra de comestibles hasta combinaciones de vídeo digital y más allá (29).

Características

- Versionado de directorios: Implementa un sistema de ficheros versionado “virtual” que sigue los cambios sobre árboles de directorios completos a través del tiempo. Ambos, ficheros y directorios, se encuentran bajo el control de versiones.
- Verdadero historial de versiones: Permite añadir, borrar, copiar, y renombrar ficheros y directorios. Y cada fichero nuevo añadido comienza con un historial nuevo, limpio y completamente suyo.
- Envíos atómicos: Una colección cualquiera de modificaciones o bien entra por completo al repositorio, o bien no lo hace en absoluto. Esto permite a los desarrolladores construir y enviar los cambios como fragmentos lógicos e impide que ocurran problemas cuando sólo una parte de los cambios enviados lo hace con éxito.
- Versionado de metadatos: Cada fichero y directorio tiene un conjunto de propiedades —claves y sus valores —asociado a él. Usted puede crear y almacenar cualquier par arbitrario de clave/valor que desee. Las propiedades son versionadas a través del tiempo, al igual que el contenido de los ficheros.
- Elección de las capas de red: Subversion tiene una noción abstracta del acceso al repositorio, facilitando a las personas implementar nuevos mecanismos de red. Subversion puede conectarse al servidor HTTP Apache como un módulo de extensión. Esto proporciona a Subversion una gran ventaja en estabilidad e interoperabilidad, y acceso instantáneo a las características existentes que

ofrece este servidor—autenticación, autorización, compresión de la conexión, etcétera. También tiene disponible un servidor de Subversion independiente, y más ligero. Este servidor utiliza un protocolo propio, el cual puede ser encaminado fácilmente a través de un túnel SSH.

- Manipulación consistente de datos: Subversion expresa las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de texto (legibles para humanos) y ficheros binarios (ilegibles para humanos). Ambos tipos de ficheros son almacenados igualmente comprimidos en el repositorio, y las diferencias son transmitidas en ambas direcciones a través de la red.

- Ramificación y etiquetado eficientes: El coste de ramificación y etiquetado no necesita ser proporcional al tamaño del proyecto. Subversion crea ramas y etiquetas simplemente copiando el proyecto, usando un mecanismo similar al enlace duro. De este modo estas operaciones toman solamente una cantidad de tiempo pequeña y constante.

- Hackability: Subversion no tiene un equipaje histórico; está implementado como una colección de bibliotecas compartidas en C con APIs bien definidas. Esto hace a Subversion extremadamente fácil de mantener y reutilizable por otras aplicaciones y lenguajes (29).

Apache 2.2

Apache es un servidor web de código libre para la transferencia de hipertextos (Hypertext Transfer Protocol, HTTP por sus siglas en inglés) para plataformas Unix, Windows y Macintosh. Su implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo que, usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada (30).

Principales características:

- Es un servidor de web conforme al protocolo HTTP/1.1.
- Soporta tanto host basados en IP como host virtuales.
- Apache soporta autentificación básica basada en la Web.

- Modular: puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.
- Personalización de las respuestas ante los posibles errores que se puedan dar en el servidor (30).

Oracle Database 11g

Oracle Database 11g Enterprise Edition ofrece confiabilidad, escalabilidad y desempeño de primer nivel para configuraciones en clúster y en un solo servidor. Ofrece las más completas características para soportar el procesamiento de transacciones más exigente, inteligencia de negocios, y aplicaciones para la administración de contenido. Protección ante las fallas del servidor, fallas del sitio, errores humanos, y reducción del tiempo de baja programado. Protección de datos con seguridad única en el nivel de filas, auditorías detalladas, y encriptación transparente de datos. Incluye data warehousing de alto desempeño, procesamiento analítico online, y características de extracción de datos. Constituye un sistema gestor de bases de datos con características objeto-relacionales. Sus principales características son las siguientes:
Entorno cliente/servidor.

- Gestión de grandes bases de datos.
- Usuarios concurrentes.
- Alto rendimiento en transacciones.
- Sistemas de alta disponibilidad.
- Disponibilidad controlada de los datos de las aplicaciones.
- Adaptación a estándares de la industria, como SQL-92.
- Gestión de la seguridad.
- Autogestión de la integridad de los datos.
- Opción distribuida.
- Portabilidad.
- Compatibilidad.
- Replicación de entornos.

- Provee un control de accesos discrecional, es decir, acceso restringido a la información basado en privilegios.

Gestiona la seguridad de la base de datos usando:

- Usuarios y esquemas de la base de datos.
- Privilegios.
- Roles.
- Ajustes de rendimiento y cuotas.
- Límites sobre los recursos.
- Auditoría.

Cada usuario posee un dominio de seguridad, que determina:

- Acciones (privilegios y roles) disponibles para el usuario.
- Cuotas sobre tablespaces.
- Límites en los recursos del sistema.

Posee varias estructuras y mecanismos de software para proveer:

- Recuperación de la base de datos ante distintos tipos de fallos.
- Operaciones de recuperación flexibles.
- Disponibilidad de los datos durante las operaciones de backup y recovery.

Utiliza varias estructuras para proveer la recuperación completa de la instancia:

- Redo Log.
- Segmentos de rollback.
- Fichero de control.
- Copias necesarias de la base de datos (31).

CONCLUSIONES PARCIALES

En el capítulo se demuestra la necesidad e importancia que tiene para la Aduana contar con un módulo en el GINA que permita monitorear a este sistema, se realiza un análisis de distintos Dashboards, los cuales debido a sus características no pueden ser adaptados al GINA. Se analizaron diferentes temáticas importantes para sentar las bases para el desarrollo de un software como son: metodologías, lenguajes y tecnologías a utilizar.

Como metodología de desarrollo de software se seleccionó la metodología AUP adaptada a la UCI. Se determinó emplear Ext JS en su versión 3.0 como marco de trabajo en el lado del cliente, Symfony en su versión 1.2 como marco de trabajo en el lado del servidor y como SGBD se empleó Oracle en su versión 11g, ya que son las tecnologías y herramientas que define el proyecto.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.1. INTRODUCCIÓN

En el siguiente capítulo se muestra la descripción de la herramienta propuesta. Primeramente se describen cada uno de los procesos de negocio mediante los cuales se capturan los requisitos necesarios para la realización del sistema, se enumeran y especifican cada uno de ellos, tanto los funcionales como los no funcionales. Posteriormente se muestran los artefactos correspondientes al diseño de clases y modelo de datos. Por último se realiza la descripción del diseño de la aplicación que se elaboró para darle solución a los objetivos propuestos.

2.2. DESCRIPCIÓN DEL SISTEMA

El módulo Dashboard de GINA es una aplicación para monitorear la información generada por los diferentes módulos que componen el sistema. Este fue diseñado con el principio de brindar a los directivos de la AGR un mayor control sobre lo que ocurre en el Sistema en tiempo real, facilitando la toma de decisiones.

Para complementar lo antes descrito el sistema permite personalizar el ambiente de trabajo, permitiendo adicionar o eliminar monitores, cada monitor puede mostrar varias vistas (gráficos y tablas) con información en tiempo real, brindada por las fuentes de datos. Se pretende cubrir totalmente o en su parcialidad los problemas de análisis de información en tiempo real que presentan los módulos de GINA.

El sistema brinda una interfaz compuesta por un panel situado a la izquierda de la pantalla, donde el usuario puede seleccionar diferentes monitores o vistas detalladas de cada monitor, si es seleccionado un monitor, este se mostrará en una pestaña con sus respectivas vistas. Cada monitor contará con un máximo de hasta 6 vistas. Cada vista utiliza una fuente de datos.

2.2.1. MODELO CONCEPTUAL

Un modelo conceptual explica los conceptos más significativos en un dominio del problema, identificando los atributos y las asociaciones. Es la herramienta más importante del análisis orientado a objetos. Un modelo conceptual representa cosas del mundo real, no componentes del software. En UML se representa mediante un grupo de diagramas de estructura estática donde no se define ninguna operación.

En estos diagramas se muestran conceptos (objetos), asociaciones entre conceptos (relaciones) y atributos de conceptos (atributos) (32). En la figura 1 se puede observar el modelo conceptual perteneciente al módulo Dashboard.

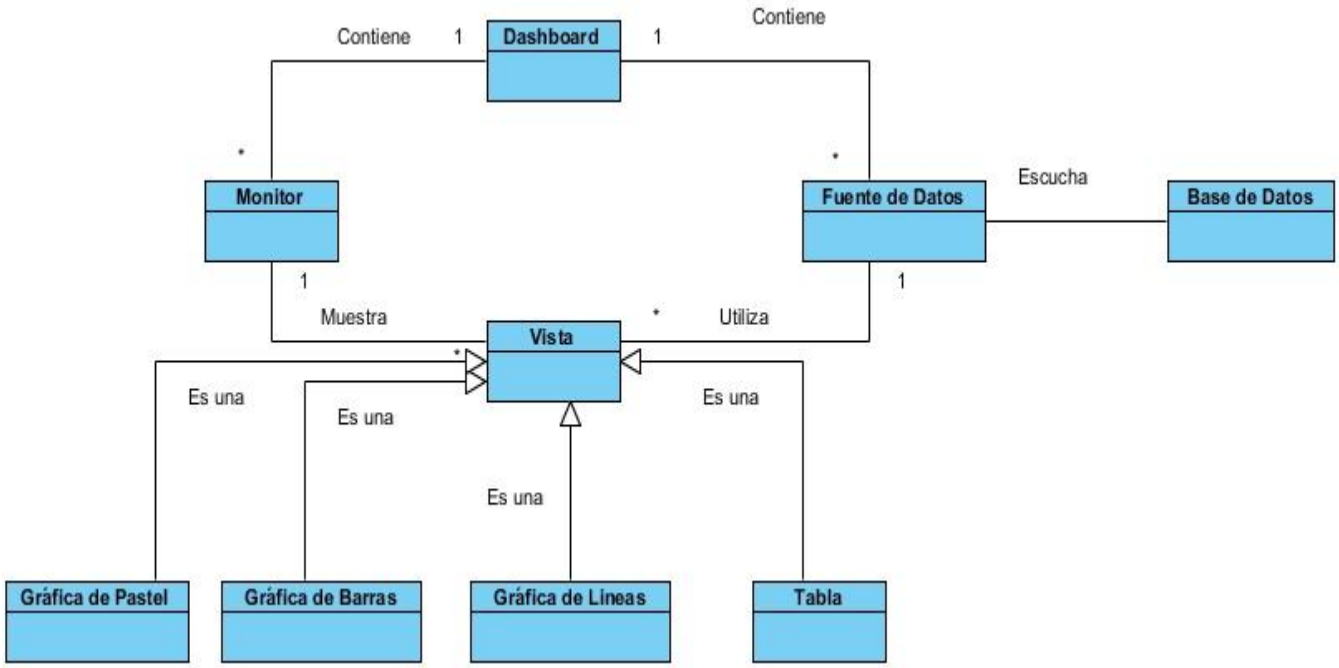

Figura 1. Modelo Conceptual

2.3. TÉCNICAS PARA LA CAPTURA DE REQUISITOS

Siguiendo los procedimientos para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana, se llevaron a cabo las siguientes técnicas para la captura de los requisitos: entrevistas y tormenta de ideas.

Entrevistas: se utilizaron para recopilar información mediante preguntas que se les realizaron a los técnicos aduaneros encargados de trabajar con los módulos de Administración, Persona y a uno de los desarrolladores de GINA encargado de los servicios. Se analizaron aquellas áreas que necesitaban ser monitorizadas, de ahí surgen todos los requisitos para lograr el correcto funcionamiento del sistema.

Tormenta de ideas: se realizó una reunión con los interesados en la que todos expresan sus ideas sobre el problema y su solución. La forma de llevarla a cabo es que cada participante diga su idea sin ser interrumpido por otro. Al finalizar la sesión se puede hacer una recolección de ideas sin duplicidad.

2.4. REQUISITOS

Los requisitos de un sistema son los aspectos que el sistema desarrollado debe cumplir. Surgen de las necesidades del cliente, de las limitaciones del entorno donde se va a implantar o de la propia gestión de la información que debe realizar el sistema. Normalmente van a representar valores que debe cumplir como mínimo o como máximo a cada uno de los aspectos desarrollados del sistema. Los requisitos sirven para acotar la funcionalidad o la construcción del sistema suponiendo límites al diseño del sistema y enumerando todas las funcionalidades que debe cubrir el sistema (33). La especificación de los requisitos de software se puede obtener en el Anexo 1.

2.4.1. PROTOTIPOS

Prototipo de sistema: es un sistema desarrollado con la finalidad de probar ideas y suposiciones relacionadas con el nuevo sistema. Los usuarios evalúan el diseño y la información generada por el sistema (34).

Razones para desarrollar prototipos de sistema: los prototipos tienen mayor utilidad bajo las siguientes condiciones (34):

1. Los encargados de diseñar e implantar sistemas nunca han desarrollado uno con las características del sistema propuesto.
2. Se conoce solo una parte de las características esenciales del sistema; las demás no son identificables a pesar de un cuidadoso análisis de requisitos.
3. La experiencia con el uso de sistema añadirá una lista significativa de requisitos que el sistema debe satisfacer.
4. Las diferentes versiones del sistema evolucionan con la experiencia, al igual que el desarrollo adicional y el refinamiento de sus características.
5. Los usuarios del sistema participan en el proceso de desarrollo.

2.4.2. REQUISITOS FUNCIONALES

Los requisitos funcionales son aquellos que afectan directamente a la funcionalidad principal del sistema. Normalmente esta funcionalidad describe los procesos de negocio a los que se destina el sistema (33). Los requisitos se encuentran descritos en el documento Dashboard_Especificacion de Requisitos de Software.

RF1. Salvar datos de Administración.

RF2.Salvar datos de Personas.

RF3.Salvar datos de Servicios.

RF4. Salvar datos de Alertas.

RF5. Mostrar monitor.

A continuación se muestra la descripción del requisito Mostrar monitor.

Especificación del requisito Mostrar monitor

Precondiciones	El Usuario se ha autenticado en el sistema. El Usuario tiene permisos para acceder al módulo Dashboard.
Flujo de eventos	
Flujo básico Mostrar monitor	
1	Seleccionar del menú una de las siguientes opciones: - Monitor Administración - VD-Usuarios Logueados - VD-Cantidad de Usuarios por Aduana - VD-Cantidad de Usuarios utilizando Aplicación - VD-Cantidad de Usuarios utilizando Módulo - Monitor API - Monitor Personas de Interés - Monitor errores en información recibida por aerolínea - Monitor Personas - VD-Personas creadas - VD-Personas creadas por Aduana - VD-Personas creadas por Aplicación - VD-Cantidad de Personas creadas por Módulo
2	Se muestra la pantalla el monitor seleccionado.
Pos-condiciones	
1	Se mostraron los gráficos en la pantalla.
Flujos alternativos N/A	
Validaciones N/A	

Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	N/A	
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

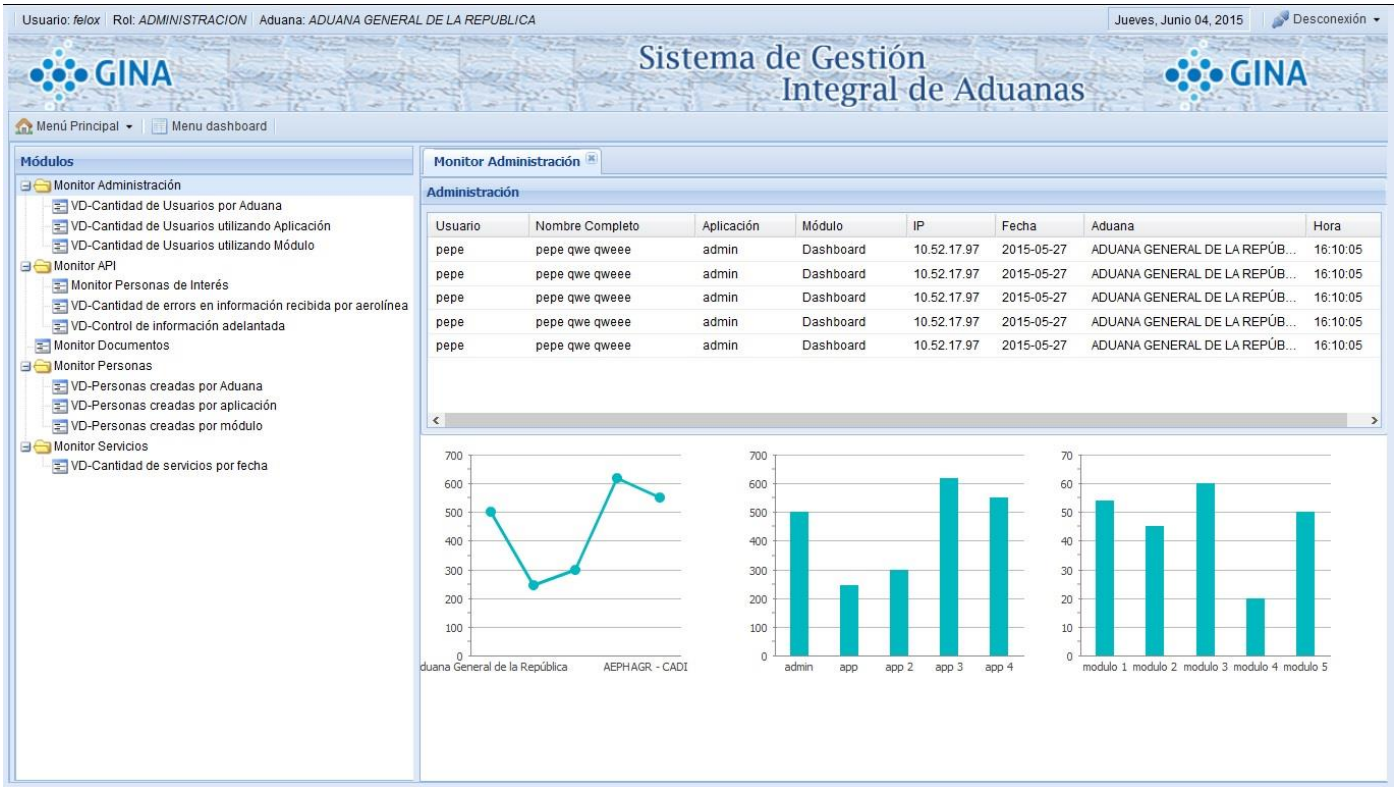

Figura 2. Prototipo de interfaz de un Monitor

2.4.3. REQUISITOS NO FUNCIONALES

Son aquellos requisitos del sistema que no representan la funcionalidad principal del sistema, sino que fijan condiciones para realizar dicha funcionalidad.

Usabilidad

Deben contar con un menú que les permita a los usuarios acceder a las principales funciones que son de su interés. La resolución de la página se adaptará a la pantalla del cliente. Podrá ser usado por cualquier persona que acceda a él que tenga algún conocimiento básico de computación y trabajo en la web.

Soporte

Las aplicaciones clientes deben ser capaz de correr sobre cualquier plataforma, para el caso de Windows se recomienda XP por la experiencia acumulada por los usuarios. Para la parte servidora se recomienda que arranque sobre plataforma Linux. Ser programado en PHP 5.3 y con un gestor de base de datos Oracle 11g.

Portabilidad

Multiplataforma. El sistema se podrá montar sobre Unix, Linux, Windows. Así mismo podrá usar una serie de gestores de base de datos, como PostgreSQL, MySQL, Oracle, aunque preferiblemente se desea la portabilidad sobre software libre.

Hardware

Cliente:

Las aplicaciones son desarrolladas para que las PC clientes de los usuarios puedan usar las aplicaciones con el menor requerimiento posible.

- Procesador Intel Pentium III de 1.4GHz de velocidad de procesamiento y 512 Mb de memoria RAM y 10Gb libres de disco duro.
- Tarjeta de red 10/100Mbits.

Servidor:

- Procesador Intel Core 2 Duo a 2.6 GHz de velocidad de procesamiento y 2Gb de memoria RAM.
- 60Gb de espacio libre en disco.
- Tarjeta de red 10/100Mbits.

2.4.4. TÉCNICAS PARA LA VALIDACIÓN DE LOS REQUISITOS

Siguiendo el procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE, el cual plantea sobre las técnicas para la validación de los requisitos lo siguiente:

En el procedimiento propuesto se recomienda el uso de varias técnicas para la correcta validación de los requisitos en el Departamento de Soluciones para la Aduana las cuales son: revisión técnica formal, construcción de prototipos, matrices de trazabilidad y generación de casos de pruebas (35).
La validación de los requisitos se realizó con la combinación de las siguientes técnicas: revisión técnica formal y construcción de prototipos. Se realizaron diversas revisiones al documento de requisitos con la participación de los analistas del proyecto, jefe del subsistema, jefe del proyecto y especialistas del Centro de Automatización para la Dirección y la Información de la Aduana (CADI) para lograr una correcta interpretación de la información transmitida, los señalamientos planteados fueron recogidos y corregidos posteriormente. Además se efectuó la construcción de prototipos, los cuales fueron presentados por cada requisito de software, aclarando con la especialista principal de Sistemas Automatizados del CADI (35).

2.5. PATRONES DE DISEÑO UTILIZADOS EN LA SOLUCIÓN

Un Patrón de Diseño es una solución repetible a un problema recurrente en el diseño de software, existen diversos tipos de patrones, entre ellos se encuentran los GOF (Gang of Four) y los GRASP (General Responsibility Assignment Software Patterns). Los patrones de diseño GOF son aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software (36). Estos están agrupados en tres grupos:

- Patrones Creacionales: Abstraen el proceso de instanciación de objetos, ayudando a que el sistema sea independiente de cómo se crean, componen y representan sus objetos.
- Patrones Estructurales: Se encargan de cómo se combinan clases y objetos parar formar estructuras más grandes.
- Patrones de Comportamiento: Tienen que ver con algoritmos y asignación de responsabilidades (36).

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones (36).

La utilización del marco de trabajo Symfony proporciona ventajas significativas para los desarrolladores de software. Este marco de trabajo es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones de diseño

arquitectónicos más utilizados en la actualidad, ya que el mismo marco de trabajo los utiliza. Se escriben a continuación los patrones usados en el desarrollo de este trabajo:

Modelo-Vista-Controlador

El Modelo-Vista-Controlador (MVC) es un patrón arquitectural aportado por SmallTalk y hoy en día muy difundido en uso en aplicaciones de entorno web. La evolución de lo que se conoce como modelo 2 de aplicaciones web (separación de responsabilidades de presentación, negocio y navegación) avanza un poco más en el reparto de tareas en la aplicación web. Pese a que hay distintos puntos de vista acerca de la forma de aplicar e implementar este patrón, en esencia las ideas principales sobre su estructura y funcionalidad son las mismas. El MVC tiene tres piezas claves que se reparten la responsabilidad de la aplicación (37).

El modelo

La lógica de negocio de las aplicaciones web depende casi siempre en su modelo de datos. El componente que se encarga por defecto de gestionar el modelo en Symfony es una capa de tipo ORM (object/relational mapping) realizada mediante el proyecto Propel. En las aplicaciones Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; de esta forma nunca se accede de forma explícita a la base de datos. Este comportamiento permite un alto nivel de abstracción y permite una fácil portabilidad (37).

La principal ventaja que aporta el ORM es la reutilización, permitiendo llamar a los métodos de un objeto de datos desde varias partes de la aplicación e incluso desde diferentes aplicaciones, también encapsula la lógica de los datos (37).

La utilización de objetos en vez de registros y de clases en vez de tablas, tiene otra ventaja: permite añadir métodos accesorios en los objetos que no tienen relación directa con una tabla (37).

La vista

La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones (37).

Los diseñadores web normalmente trabajan con las plantillas y con el layout. Estas partes están formadas por código HTML que contiene pequeños trozos de código PHP (37).

Para mejorar la reutilización de código, los programadores suelen extraer trozos de las plantillas y los transforman en componentes y elementos parciales. De esta forma, el layout se modifica para definir zonas en las que se insertan componentes externos. Los diseñadores web también pueden trabajar fácilmente con estos trozos de plantillas (37).

Los programadores normalmente centran su trabajo relativo a la vista en los archivos de configuración YAML (que permiten establecer opciones para las propiedades de la respuesta y para otros elementos de la interfaz) y en el objeto respuesta. Cuando se trabaja con variables en las plantillas, deben considerarse los posibles riesgos de seguridad de XSS (cross-site scripting) por lo que es necesario conocer las técnicas de escape de los caracteres introducidos por los usuarios. Independientemente del tipo de trabajo, existen herramientas y utilidades para simplificar y acelerar el trabajo (normalmente tedioso) de presentar los resultados de las acciones (37).

El controlador

Responsable del flujo de control, la navegabilidad y el estado de la entrada del usuario. Es el corazón del funcionamiento del patrón y responsable de (37):

1. Interceptar y recoger las peticiones http del cliente. Así, el cliente no invocará directamente ninguna página jsp o html, sino que será redireccionado adecuadamente por el controlador.

2. Traducir la petición en una operación de negocio específica.

3. Invocar la operación o bien delegar en un manejador.

4. Determinar la siguiente vista a mostrarle al cliente

5. Retornar el control al cliente.

6. El hecho de que todas las peticiones http pasen por el controlador facilita el mantenimiento de la aplicación, sobre todo en lo referente al control de la navegabilidad, sustitución de páginas, etc (37).

Patrones GOF:

Observador (Observer): clasifica como patrón de comportamiento a nivel de objeto, el problema que lo motiva es la existencia de diferentes objetos desacoplados que deben mantenerse actualizados del estado de otro objeto, o de los determinados sucesos que ocurren en él, su propósito es garantizar que los interesados en el estado del objeto observado sean notificados convenientemente (38).
El observador es ampliamente utilizado en el diseño de componentes de interfaz de usuario de la capa de presentación. Define una relación de un objeto a muchos objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros objetos (38).

Singleton (Instancia única): asegura que una determinada clase sea instanciada una y sólo una vez, proporcionando un único punto de acceso global a ella (38). Este patrón se evidencia en el controlador frontal, donde hay una llamada a la función sfContext::getInstance () que garantiza que siempre se acceda a la misma instancia.

Command: este patrón se observa en la clase sfWebFrontController, en el método dispatch(). Esta clase está por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Este patrón se aplica además en la clase sfRouting, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el marco de trabajo, la cual se puede activar o desactivar. En este método es parseada la URL con el objetivo de precisar los parámetros de la misma y de esta forma saber el Actions que debe responder a la petición (38).

Patrones GRASP:

Experto: es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Expresa que siempre se debe asignar una responsabilidad al experto en información, o sea, a la clase que cuenta con la información necesaria para llevar a cabo la funcionalidad (20).
Creador: este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica del mismo es encontrar un creador que necesite conectarse al objeto creado en alguna situación, eligiéndolo como el creador. Se favorece el bajo acoplamiento (20).

Bajo Acoplamiento: el patrón bajo acoplamiento impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a los resultados negativos que puede producir un acoplamiento alto. No soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio. El mismo no se puede considerar de manera aislada a otros patrones como el Experto o el de Alta Cohesión, sino que necesita incluirse como uno de los diferentes principios de diseño que influyen en una elección, al asignar una responsabilidad (20).

Alta Cohesión: este patrón es un principio a tener en mente durante todas las decisiones de diseño. Constituye un objetivo subyacente a tener en cuenta continuamente. Es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. Como beneficios que este aporta se pueden mencionar (20):

- Se incrementa la claridad y facilita la comprensión del diseño.
- Se simplifican el mantenimiento y las mejoras.
- Se soporta a menudo bajo acoplamiento.

Controlador: es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. La mayor parte de los sistemas reciben eventos de entrada externa, en cualquiera de los casos que puedan existir, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada (20).

2.6. DIAGRAMA DE CLASES DEL DISEÑO CON ESTEREOTIPOS WEB

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. En la figura 3 se muestra el diagrama de clases del diseño con estereotipos web del requisito funcional Mostrar monitor, donde se representan las principales clases, operaciones y relaciones que se necesitan para darle cumplimiento al requisito.

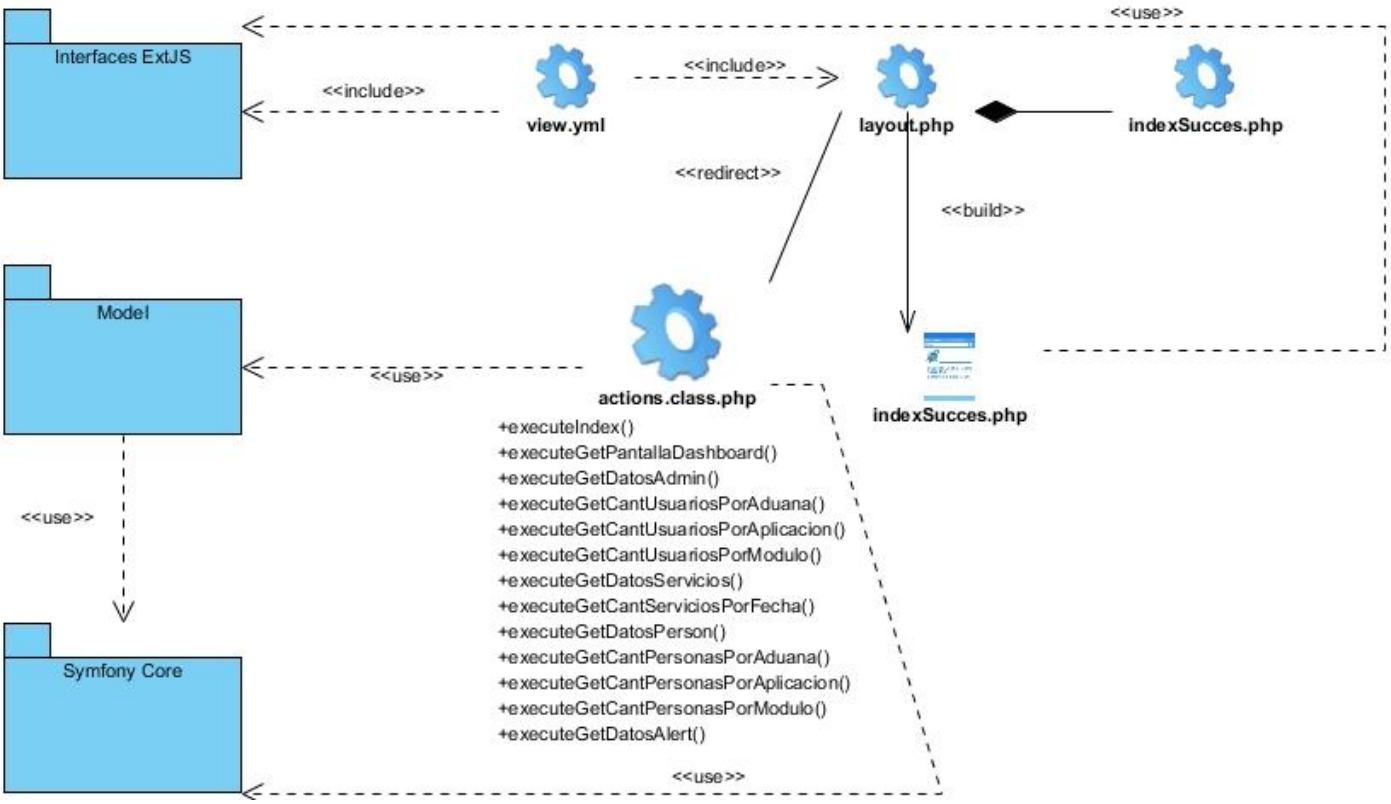

Figura 3. Diagrama de Clases con Estereotipos Web del requisito Mostrar monitor

2.7. DIAGRAMA DE SECUENCIA

El diagrama de secuencia permite modelar la interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación por orden de tiempo y se modela cada método de las clases participantes en el requisito. En la figura 4 se muestra el diagrama de secuencia del monitor de administración perteneciente al requisito funcional Mostrar monitor.

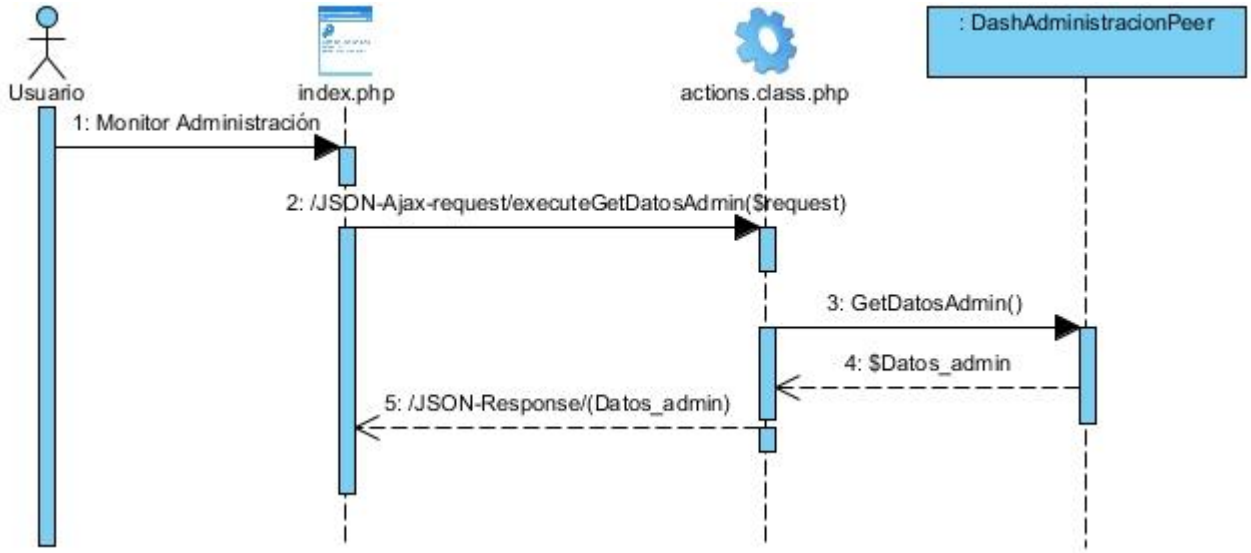

Figura 4. Diagrama de Secuencia de Mostrar monitor administración.

2.8. MÉTRICAS PARA LA VALIDACIÓN DEL DISEÑO

La clase es la unidad principal de todo sistema orientado a objetos. Esto implica que las medidas y métricas para una clase individual, la jerarquía y las colaboraciones sean sumamente valiosas para un ingeniero de software que tenga que estimar la calidad de un diseño. La métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado, los objetivos principales de las métricas son: comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel del proyecto (39).

A continuación se aplica una de las métricas de diseño orientado a objeto y se analizan los resultados para determinar la calidad del diseño planteado.

Métrica de Tamaño Operacional de Clase (TOC)

Esta métrica es propuesta por Lorenz y Kidd, los cuales dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y asuntos relacionados con el código, y las métricas orientadas a valores externos examinan el acoplamiento y la reutilización (39).

La métrica TOC es muy usada por diseñadores de software, con una amplia documentación y literatura, fácil de calcular y bastante efectiva. Se basa en el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (**Tabla 3**):

Tabla 3. Afectaciones en el diseño según la métrica TOC.

Atributos	**Afectación**
Responsabilidad	El aumento del TOC implica el aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	El aumento del TOC implica el aumento de la complejidad de implementación de la clase.
Reutilización	El aumento del TOC implica la disminución del grado de reutilización de la clase.

Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Algunos especialistas plantean umbrales para esta métrica basándose en el promedio de operaciones por clases obtenidos, estos valores fueron los aplicados en el diseño de este sistema. (**Tabla 4**).

Tabla 4. Valores de los umbrales para la métrica TOC.

	Criterio	**Categoría**	
Responsabilidad	Baja	<=Promedio	<= 9,43
	Media	>Promedio y <= 2*Promedio	>9,43<=18,86
	Alta	>2*Promedio	>18,86
Complejidad Implementación	Baja	<=Promedio	<=9,43
	Media	>Promedio y <= 2*Promedio	>9,43<=18,86
	Alta	>2*Promedio	>18,86
Reutilización	Baja	>2*Promedio	>18,86
	Media	>Promedio y <= 2*Promedio	>9,43<=18,86
	Alta	<=Promedio	<= 9,43

En las figuras de la 5 a la 8 se muestran los resultados de la evaluación de la métrica.

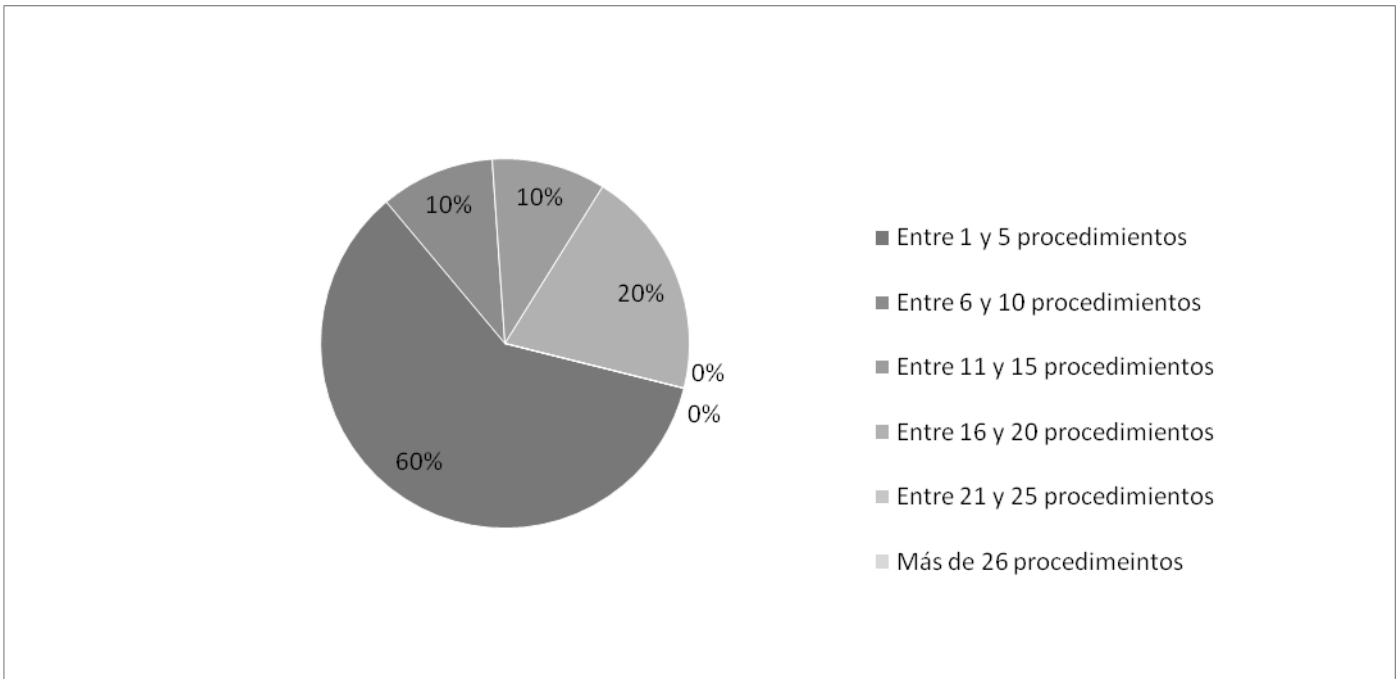

Figura 5.Representación del tamaño de las clases del módulo Dashboard

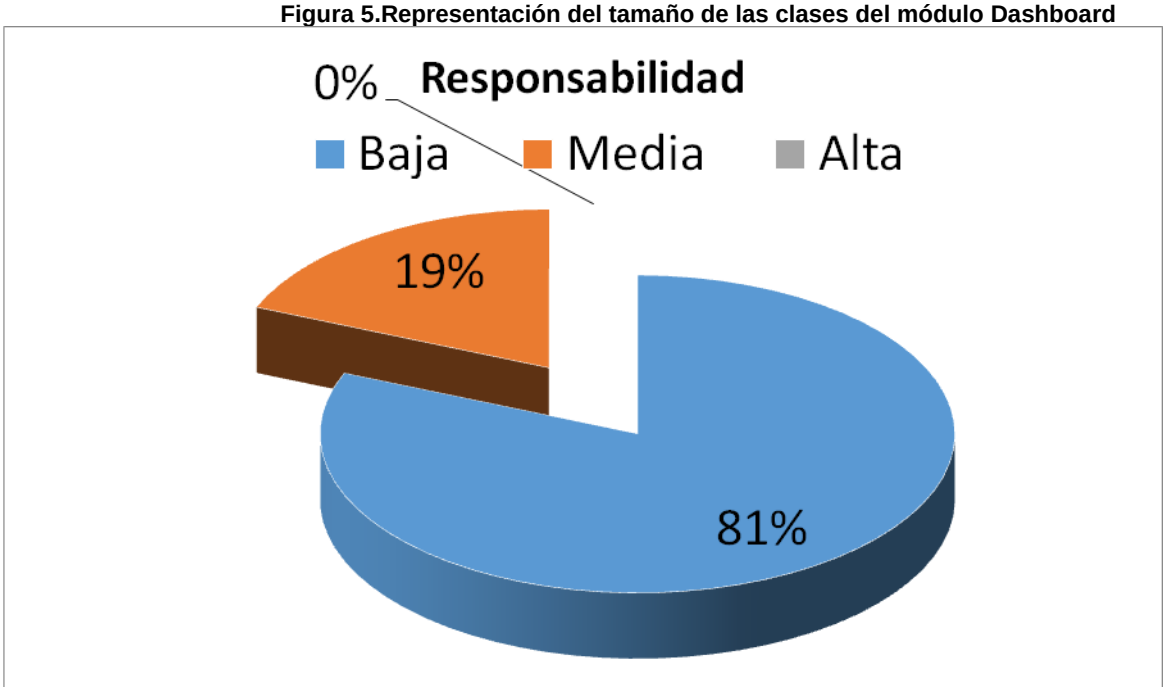

Figura 6.Representación de las clases según la responsabilidad.

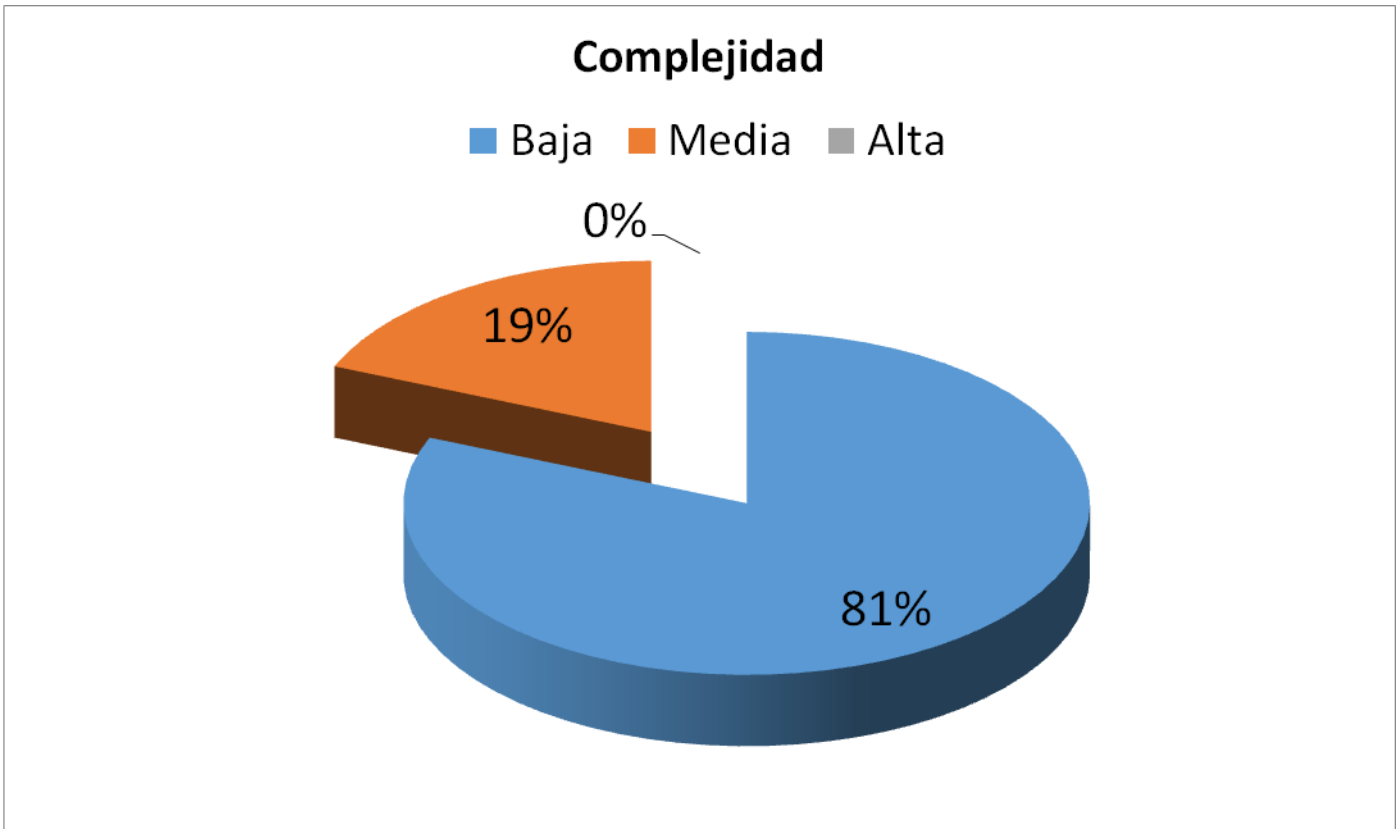

Figura 7.Representación de las clases según la complejidad de implementación.

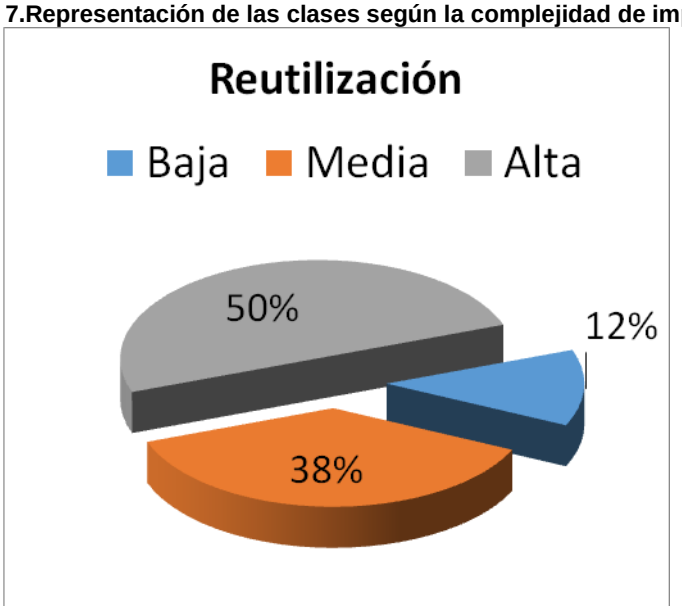

Figura 8.Representación de las clases según según su reutilización.

Tras un análisis de los resultados arrojados por la evaluación bajo los instrumentos de medición de la métrica TOC, se demuestra que se alcanzaron buenos valores para cada uno de los atributos de calidad evaluados, puesto que, como se puede observar, el 60% de las clases del software contienen un número menor que el promedio de procedimientos por clases, lo cual influye positivamente en el hecho de que predomine una responsabilidad baja de las clases en un 81%, y hace que carezcan de mucha complejidad, por lo que se tornan reutilizables. Estos valores demuestran que los indicadores de reutilización, complejidad y responsabilidad no se ven afectados.

2.9. MODELO DE DATOS

Una forma de agrupar los datos es el Modelo de Entidad-Relación que no es más que un modelo de datos basado en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos, implementándose en forma gráfica (40). En la figura 9 se muestra el modelo de datos correspondiente al módulo Dashboard.

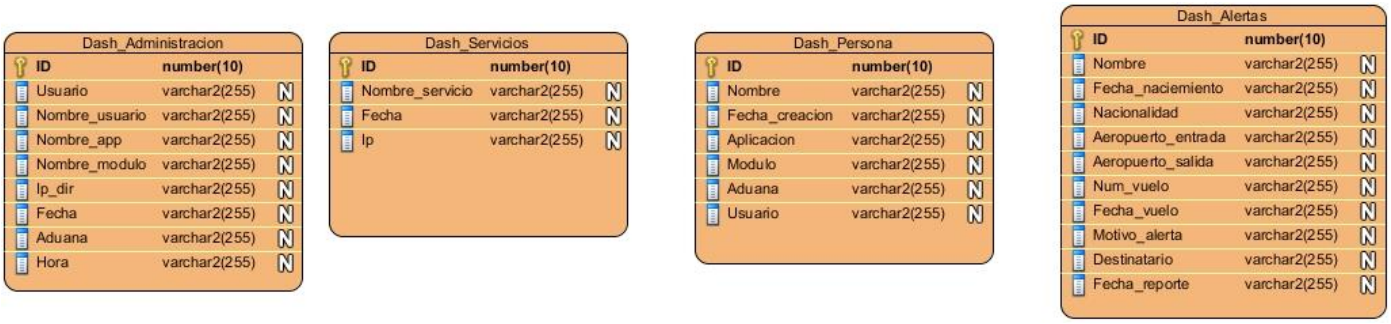

Figura 9. Modelo de datos

2.10. CONCLUSIONES PARCIALES

En el capítulo se definieron los requisitos funcionales del sistema y las diferentes técnicas empleadas para la captura y validación de los mismos. Se realizó el diseño de la solución, obteniendo como resultado productos de trabajo en los que se exponen varios diagramas tales como: el diagrama de secuencia, el de clases del diseño con estereotipos web y el modelo de datos. Además se presentaron los resultados de su validación, evidenciándose a través de la utilización de la métrica TOC, buenos valores para los atributos de calidad, predominando una baja responsabilidad, complejidad y un alto por ciento de reutilización.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1. INTRODUCCIÓN

En este capítulo se lleva a cabo la implementación del sistema, describiendo el estándar de codificación utilizado para el desarrollo del mismo y se mostrará parte del código obtenido. Además se realizará la validación del sistema implementado mediante pruebas de software, con el objetivo de medir el grado en que este cumple con los requisitos planteados por el cliente.

3.2. ESTÁNDAR DE CODIFICACIÓN

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

En la actualidad existen diferentes estándares para cada uno de los lenguajes, su utilización permite una comunicación fluida y directa entre los programadores de manera que se favorece la reutilización y mantenimiento de los sistemas.

Para la implementación del módulo Dashboard fue necesario regirse por el estándar de codificación utilizado en el proyecto GINA, el cual se muestra a continuación.

Acciones:

Todos los nombres de acciones deben estar en la nomenclatura “CamelCase” comenzando por la palabra execute.

Nombres de las clases:

- Los nombres de las clases deben estar expresados en notación “UpperCamelCase”.
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.

Nombres de los archivos de las clases:

- Los nombres de los archivos de las clases deben estar compuestos por el nombre de la clase seguido de un punto y la palabra “class” y la extensión del archivo “.php”.

Nombres de las funciones:

- Los nombres de las funciones deben dejar reflejado claramente cuál es la acción que realiza el mismo.

- Se debe apoyar en la utilización de sufijos que ayuden a identificar el resultado final de la ejecución de un método.
- Se debe apoyar en los prefijos para expresar la acción que realiza sobre un elemento determinado.

3.2.1. TRATAMIENTO DE ERRORES

Entre los aspectos más importantes a tener en cuenta durante el desarrollo de un software se encuentra el tratamiento de errores, debido a que los usuarios cometen errores a la hora de ejecutar cualquier acción sobre la aplicación. En el sistema propuesto se tratan los errores de forma tal que las interacciones con la base de datos (inserción y obtención) se realicen de forma correcta.

En la vista del sistema no fue necesario validar la entrada de datos ya que el usuario solo interactuará con el sistema seleccionando opciones en pantalla. Por otro lado se tomó la estrategia de validar los datos que van a ser almacenados en la base de datos dándose el caso de que si los datos fuesen incorrectos se lanzaría un excepción deteniendo el proceso. En la figura 10 se puede observar un fragmento del código de la clase DashboardServ en el que se realiza el tratamiento de errores.

```
try {
        $valores = array(
            'nombreServicio' => $nombre,
            'fecha' => date('Y-m-d'),
            'ip' => sfContext::getInstance()->getRequest()->getHttpHeader('addr', 'remote'),

        );

        $servicio = new DashServicios();
        $servicio->fromArray($valores, BasePeer::TYPE_STUDLYPHPNAME);
        $servicio->save();

} catch (Exception $exc) {
    sfContext::getInstance()->getLogger()->err("Error al guardar los datos de Servicios: " . $exc->getMessage());
    sfContext::getInstance()->getLogger()->err("Datos de Servicios:");
    sfContext::getInstance()->getLogger()->err(print_r($valores), true);
}
```

Figura 10. Ejemplo de tratamiento de errores en la clase DashboardServ

3.2.2. DIAGRAMA DE DESPLIEGUE

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos, donde cada nodo representa un recurso de cómputo que poseen relaciones que representan medios de comunicación entre ellos. Representa una

correspondencia entre la arquitectura de software y la arquitectura del sistema (hardware). En la figura 11 se muestra el diagrama de despliegue del proyecto.

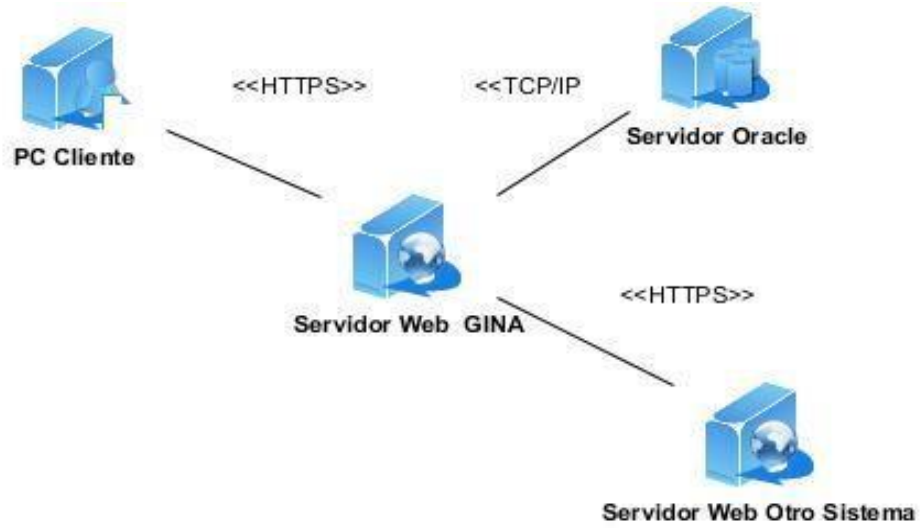

Figura 11. Diagrama de despliegue

Servidor Oracle:

Representa el servidor de bases de datos Oracle, en el cual se encuentran almacenados los datos del sistema GINA.

Nodo PC Cliente:

Representa los ordenadores personales con los cuales los usuarios del sistema accederán a la aplicación del sistema GINA.

Nodo Servidor Apache GINA:

Representa el servidor web donde se encontrará la aplicación del sistema GINA en la cual se encuentra la solución abordada en el presente trabajo.

Nodo Servidor Web Otro Sistema:

Representa otro servidor web de un sistema con el cual el sistema GINA puede tener intercambio de información.

3.3. PRUEBAS DE VALIDACIÓN DE LA SOLUCIÓN

Al desarrollar sistemas informáticos se corre un alto riesgo de que se produzcan errores, los cuales pueden ocurrir desde el comienzo del proceso, ya sea en la definición de los objetivos, el diseño, la implementación o en otras fases.

El proceso de pruebas tiene como objetivos fundamentales planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Diseñar e implementar las pruebas creando los casos de pruebas que especifican qué probar. Además de realizar las diferentes pruebas y manejar los resultados de cada una sistemáticamente de forma que las no conformidades puedan ser corregidas (40).

Para realizar la validación de la solución propuesta se aplicarán las pruebas funcionales de caja negra, haciendo uso de la técnica Partición de equivalencia para comprobar la validez en las respuestas del programa ante las acciones del usuario y la calidad de las salidas en dependencia de las entradas y pruebas de caja blanca, haciendo uso de la técnica Camino básico, la cual permite al diseñador de casos de prueba 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.

3.3.1. PRUEBAS FUNCIONALES DE CAJA NEGRA

Se realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una “Caja Negra” cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas (42).

Durante la primera iteración se encontraron un total de 9 no conformidades, sobre textos en diferentes opciones, pero estas fueron corregidas en su totalidad, ya que al efectuar la segunda iteración no se detectó ninguna no conformidad. En la figura 12 se muestran los resultados obtenidos por cada una de las iteraciones.

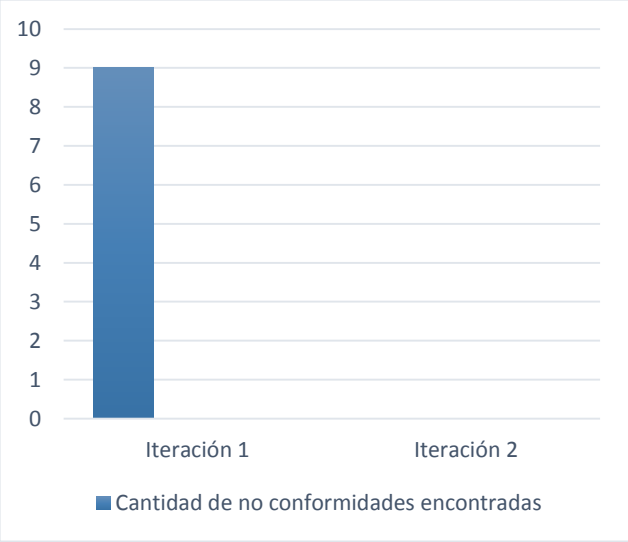

Figura 12.Cantidad de no conformidades por iteración.

A continuación se muestra el diseño del caso de prueba correspondiente al requisito funcional: Mostrar monitor.

Tabla 5.Clase equivalente válida.

Escenario	**Resultado esperado**	**Resultado obtenido**
EC 1.1 Mostrar monitor de Administración.	El sistema debe mostrar los gráficos del monitor de Administración.	Se mostró en pantalla el monitor de Admnistración.
EC 1.2 Mostrar monitor de Persona.	El sistema debe mostrar los gráficos del monitor de Persona.	Se mostró en pantalla el monitor de Persona.

Tabla 6.Clase equivalente no válida.

Escenario	Resultado esperado	Resultado obtenido
EC 1.1 Mostrar monitor de Administración.	El sistema no debe mostrar los gráficos del monitor de Administración.	Se mostró en pantalla un mensaje de alerta indicando que no hay conexión con la base de datos.
EC 1.2 Mostrar monitor de Persona.	El sistema no debe mostrar los gráficos del monitor de Persona.	Se mostró en pantalla un mensaje de alerta indicando que no hay conexión con la base de datos.

3.3.2. PRUEBAS UNITARIAS DE CAJA BLANCA

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de Flujo en el cual (11):

1. Cada nodo del grafo corresponde a una o más sentencias de código fuente.
2. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.
3. Se calcula la complejidad ciclomática del grafo.

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración y comprensión, estos brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo (11):

- **Nodo**: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.
- **Aristas**: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

- **Regiones**: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Para realizar la prueba de caja blanca, específicamente la técnica del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar (11). En la figura 13 se enumeran las sentencias de código del procedimiento realizado sobre el método getDataPersona().

```
static function getDataPersona(sfEvent $event) {

    $var = $event->getParameters(); // 1

    try { //2
        $valores = array(
            'nombre' => $var->getNombreCompleto(),
            'fechaCreacion' => date('Y-m-d'),
            'aplicacion' => sfConfig::get('sf_app'),
            'modulo' => sfContext::getInstance()->getRequest()->getParameter('module'),
            'aduana' => sfContext::getInstance()->getUser()->getAttribute('aduana_descripcion'),
            'usuario' => sfContext::getInstance()->getUser()->getUsername(),
        );

        $person = new DashPersona();
        $person->fromArray($valores, BasePeer::TYPE_STUDLYPHPNAME);
        $person->save();


    } catch (Exception $exc) { //4
        sfContext::getInstance()->getLogger()->err("Error al guardar los datos de Persona: " . $exc->getMessage());
        sfContext::getInstance()->getLogger()->err("Datos de Persona:");
        sfContext::getInstance()->getLogger()->err(print_r($valores, true));
    }
} // 3
```

Figura 13. Sentencias de código

Después de este paso, es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones. En la figura 14 se presenta el grafo correspondiente al código anterior.

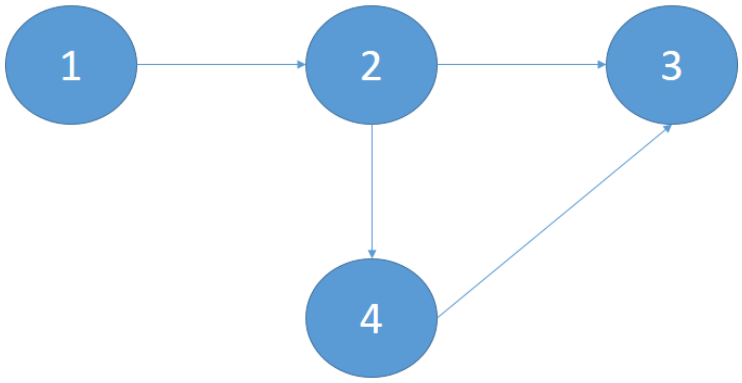

Figura 14. Grafo de flujo.

Cálculo de la complejidad ciclomática

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una edición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. (26)

Para efectuar el cálculo de la complejidad ciclomática del código, es necesario tener varios parámetros como son la cantidad total de aristas del grafo 4, cantidad total de nodos 4, cantidad total de nodos predicados 1 y la cantidad total de regiones 2, para las siguientes fórmulas (26):

1. V(G) = (A - N) + 2

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

V(G) = (4 - 4) + 2

V(G) = 2

2. V(G) = P +1

Siendo “P” la cantidad de nodos predicados (son aquellos nodos de los que parten dos o más aristas).

V(G) = 1 + 1 = 2

3. V(G) = R

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

V(G) = 2

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, dando como resultado 2, lo que indica que existen 2 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- Camino básico #1: 1-2-3
- Camino básico #2: 1-2-4-3

Tabla 7.Escenarios de casos de pruebas de cada camino básico.

Camino básico	**Descripción**	**Condición de ejecución**	**Obtenido**	**Resultado esperado**
1	Los datos son almacenados en la base de datos	El arreglo debe contener información	Un arreglo con los datos solicitados	Se almacenó en la base de datos la información contenida en el arreglo
2	Los datos son almacenados en la base de datos	El arreglo debe contener información	Un arreglo vacío	Se detiene el proceso

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el código es correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.4. VALIDACIÓN DE LA INVESTIGACIÓN

Este epígrafe tiene como objetivo analizar las ventajas que supone la utilización del módulo Dashboard. Para ello se elabora una propuesta fundamentada en los antecedentes de cómo se realizaba el control del sistema GINA en la AGR, teniendo en cuenta dos variables de medición.

Tabla 8.Situación del GINA antes y después de la elaboración del módulo Dashboard.

Variables de medición	Antes	Después
Información generada por el sistema.	No existe una herramienta que muestre gráficamente la información generada por el sistema.	El módulo Dashboard permite mostrar mediante gráficos y en tiempo real la información generada por el sistema.
Contribución a la toma de decisiones	No se podía saber cuántos usuarios se encontraban trabajando en el sistema, ni la carga que soportaban los servidores con el consumo de los servicios, lo cual dificultaba a los directivos conocer el comportamiento del sistema y tomar decisones en base a esto.	Se puede llevar un control de los usuarios que se encuentran trabajando en el sistema así como de cuáles son los servicios más utilizados. También se puede determinar si ocurre alguna anomalía en el comportamiento del sistema, lo cual facilita tomar decisiones para solucionar algún problema que pudiese surgir de manera oportuna.

3.5. CONCLUSIONES PARCIALES

La implementación y calidad del módulo fueron las premisas fundamentales en el desarrollo de este capítulo. Se efectuaron las pruebas de software necesarias para lograr la calidad requerida de la aplicación propuesta. Fueron validadas las funcionalidades implementadas a través de las pruebas de caja negra y caja blanca, mostrando cómo respondían adecuadamente a los requisitos funcionales y garantizando la satisfacción plena de las necesidades reales de los usuarios y demandas del cliente. Por último se validó la investigación demostrando las ventajas que trae consigo la elaboración del módulo Dashboard.

CONCLUSIONES

Al finalizar la presente investigación se lograron cumplir los objetivos planteados de manera satisfactoria, haciendo más sencilla la toma de decisiones por parte de los directivos de la AGR al obtener el módulo Dashboard.

Con la elaboración del marco teórico de la investigación a partir del estudio del estado del arte se evidenció la carencia de una solución informática capaz de responder a las necesidades y requisitos de la AGR.

A través de la ingeniería de requisitos se logró especificar las funcionalidades necesarias en el módulo.

El diseño de la aplicación permitió sentar las bases para la implementación del módulo Dashboard.

La implementación del módulo arrojó como resultado el código fuente de los diseños especificados en el modelo de diseño.

Se validó la solución a través de pruebas de software y una carta de aceptación emitida por el cliente validando el cumplimiento de los requisitos.

RECOMENDACIONES

- Continuar implementando monitores para cubrir la totalidad del Sistema.
- Implementar un método para generar los Dashboards dinámicamente.

BIBLIOGRAFÍA

1. **Marketing Electronico.** ***CRM y ERP.*** **[En línea] [Citado el: 15 de 11 de 2014.] http://www.marketingelectronico.com/blog/que-diferencia-hay-entre-erp-y-crm/.**

2. Canney Restrepo, Edward. TodoBI. [En línea] [Citado el: 25 de 11 de 2014.] http://todobi.blogspot.com/2007/08/la-respuesta-esta-en-los-dashboards.html.

3. Few, Stephen. ***Common Pitfalls in Dashboard Design.*** **2006.**

4. Malik, Shadan. ***Enterprise Dashboard:: Design and Best Practices for IT.*** **Hoboken, New Jersey : John Wiley & Sons, Inc., 2005.**

5. Kirkpatrick, Marshall. ReadWriteWeb. ***Explaining the Real-Time Web in 100 Words or Less.*** **[En línea] 22 de 9 de 2009. [Citado el: 25 de 11 de 2014.] http://www.readwriteweb.com/archives/explaining_the_real-time_web_in_100_words_or_less.php..**

6. DZ, Mark P. Dashboard Zone. [En línea] [Citado el: 26 de 11 de 2014.] http://www.dashboardzone.com/what-is-a-dashboard.

7. Tuck, Allene, Pyne, Sandra y Ashby, Michael F. ***Oxford dictionary of computing for learners of Enlish.*** **Oxford : Oxford University Press. 0194314413.**

8. Young, S.J. ***Real Time Languages: Desing and Development.*** **1982.**

9. Burns, Alan y Wellings, Andy. ***Real-time systems and programming languages.*** **Toronto : Addison Wesley. 1996.**

10. Gutiérrez, Javier J. ¿Qué es un framework web? ***Departamento de Lenguajes y Sistemas Informaticos.*** **[En línea] 2009. [Citado el: 26 de 11 de 2014.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.**

11. Pérz Herrera, Andy. ***WAOX v 1.0: DASHBOARD PARA EL MONITOREO DE INFORMACIÓN EN TIEMPO REAL.*** **La Habana : s.n., 2012.**

12. Pentaho. Pentaho. ***Powerful Analytics Made Easy.*** **[En línea] [Citado el: 26 de 11 de 2014.] http://community.pentaho.com..**

13. OpenReports. ***OpenReports Administration Guide.***

14. IBM. ***IBM Cognos 8 Platform.*** **[En línea] [Citado el: 14 de 1 de 2015.] http://www-01.ibm.com/software/data/cognos/products/cognos-8-platform/.**

15. IBM. ***International Business Machines Corporation.*** **[En línea] [Citado el: 16 de 1 de 2015.] http://www-01.ibm.com/software/analytics/cognos-8-go/dashboard/.**

16. IBM. ***IBM Cognos 8 Go! Dashboard.*** **s.l. : IBM Corporation, 2006.**

17. SAS. ***SAS(R) BI Dashboard 4.31: User's Guide.*** **[En línea] [Citado el: 16 de 1 de 2015.] http://support.sas.com/documentation/cdl/en/bidbrdug/64524/HTML/default/viewer.htm#n1t3l2n9us6eyyn1nm98l2wfedh7.htm.**

18. Builders, Information. ***WebFocus 8.*** **New York : s.n., 2010. DN7506431.0410.**

19. Sánchez, Tamara Rodríguez. ***Metodología de Desarrollo para la Actividad Productiva de la Uci.*** **La Habana : s.n., 2014.**

20. Potencier, Fabien y Zaninotto, François. ***Symfony la guía definitiva.*** **2008.**

21. Corzo, Giancarlo. Desarrollo en Web. ***Blog sobre desarrollo de aplicaciones web en Java, Python y JavaScript.*** **[En línea] 22 de 10 de 2008. [Citado el: 18 de 1 de 2015.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.**

22. Propelorm. ***Guía de Usuario de Propel.*** **[En línea] [Citado el: 18 de 1 de 2015.] http://svn.propelorm.org/tags/1.2.1/docs/es/user_guide/book/chapters/Introduction.html.**
23. PHP Documentation Group. ***Hypertext Pre-processor.*** **[En línea] 2 de 12 de 2011. [Citado el: 18 de 1 de 2015.] http://www.php.net/manual/es/index.php.**
24. Pérez, Javier Eguíluz. ***Introducción a Javascript.*** **2009.**
25. Pérez, Javier Eguíluz. Introducción a CSS. ***librosweb.*** **[En línea] Libros Web, 2008. www.librosweb.es/css.**
26. json. ***The JSON Specification.*** **[En línea] [Citado el: 18 de 1 de 2015.] http://json.org.**
27. Pressman, Roger S. ***Ingenieria de Software. Un enfoque Práctico 5ta Edición.*** **Madrid : Concepción Femández Madrid, 2002.**
28. NetBeans. ***Oracle Corporation.*** **[En línea] [Citado el: 24 de 1 de 2015.] http://netbeans.org/index_es.html.**
29. Collins-Sussman, Ben, W. Fitzpatrick, Brian y C. Michael Pilato. ***Control de versiones con Subversion: Revision 4980.*** **s.l. : O'Reilly Media.**
30. Ford, A. ***Apache 2 pocket reference.*** **s.l. : O'Reilly Media.**
31. Hernando, Roberto Velasco. [En línea] [Citado el: 15 de 2 de 2015.] http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.
32. GUERRERO, L. A. Análisis y Diseño Orientado a Objetos. [En línea] [Citado el: 28 de 5 de 2015.] http://www.dcc.uchile.cl/~luguerre/cc40b/clase4.html.
33. González Almirón, Cristóbal. ***Gestión de los Requisitos.*** **2010.**
34. Carvajal García, Evelin. ***INTRODUCCION AL DESARROLLO DE SISTEMAS DE INFORMACION.***
35. Martínez, Ing. Jenni Manso. ***Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.*** **La Habana : Universidad de las Ciencias Informáticas, 2010.**
36. Maestras, Juan Pavón. Dep. Ingeniería del Software e Inteligencia Artificial. [En línea] Universidad Complutense Madrid, 2008-09. [Citado el: 20 de 3 de 2015.] https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.
37. Gómez Baryolo, Oinier, Morejón Borbón, Yoandry y García Tejo, Darien. ***ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.*** **Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2009.**
38. Larman, Craig. ***UML y patrones.*** **México : Prentice-Hall Hispanoamericana, 1999.**
39. Lorenz, M. y Kidd, J. ***Object-Oriented Software Metrics.*** **1994.**
40. Costa, Dolors Costal. ***Introducción al diseño de bases de datos.***
41. Addison Wesley, James Jacobson, Ivar, Booch, Grady y Rumbaugh. ***El Proceso Unificado de Desarrollo de Software.*** **1999. 0-201-57169-2.**
42. Natalia Juristo, Ana M. Moreno, Sira Vegas. ***TÉCNICAS DE EVALUACIÓN DE SOFTWARE.*** **12 de octubre 2006.**

GLOSARIO

Framework: Estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Módulo: Es un componente autocontrolado de un sistema, el cual posee una interfaz bien definida hacia otros componentes; algo es modular si es construido de manera tal que se facilite su ensamblaje, acomodamiento flexible y reparación de sus componentes.

JSON: Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

AJAX: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

1