Universidad de las Ciencias Informáticas
Facultad 3

Título: Módulo gestión de trazas del Sistema Integral de Seguros Nacionales

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): Alexis Roberto Valle Mazorra

Tutor(es): Ing. Alian Villa Ochoa

Ing. Ivian Laobel Castellano Betancourt

28 de junio del 2018

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización de Entidades 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 _____.

Alexis Roberto Valle Mazorra

_____

Ing. Ivian Laobel Castellano Betancourt

_____

Ing. Alian Villa Ochoa

_____

DATOS DE CONTACTO

Tutor: Ing. Ivian Laobel Castellano Betancourt.
Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI).
Usuario: ilcastellano@uci.cu

Tutor: Ing. Alian Villa Ochoa.
Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI).
Usuario: alian@uci.cu

Agradecimientos

Le agradezco a mis padres, a mi abuelo, a Zirbia, a mi novia Lisandra que me dio muchas fuerzas a lo largo de la carrera, a Armando, a mis tías Danelis y Odalis, a Raykof y Dasiel por tantas horas de estudio y de football que compartimos, a mi hermanos Hermes y Marlon que siempre me apoyaron, a Liuba, a todos mis compañeros del año, a los del equipo de football sala y 11 que siempre estuve a gusto con ellos en el terreno y a todo el que tuvo que ver en algo con este trabajo como mis tutores y los trabajadores del proyecto SISEN.

Dedicatoria

Le quiero dedicar este trabajo a mi mamá porque siempre me guío por el buen camino. Ella siempre creyó en mí cuando peor lo estaba pasando en la carrera, y me dio fuerzas para seguir adelante.

A mi papá que igual que mi mamá estuvo ahí cada vez que lo necesité, y siempre estuvo pendiente a cada nota que obtuve.

A mi abuelo que siempre ha estado ahí pendiente de mi mamá cuando yo estaba en la escuela.

RESUMEN

En el Centro de Informatización de Entidades (CEIGE) en la UCI se encuentra en desarrollo el Sistema Integral de Seguros Nacionales, una aplicación web que tiene como objetivo informatizar los procesos de la Empresa de Seguros Nacionales. Sin embargo, no existe un mecanismo capaz de realizar el monitoreo y control del sistema para mejorar su seguridad. El presente trabajo de diploma tiene como objetivo desarrollar un módulo que gestione las trazas en el SISEN. Para guiar el proceso de desarrollo del módulo se caracteriza la metodología, herramientas y tecnologías empleadas. Para la estructura, diseño e implementación de las clases se utiliza el patrón arquitectónico Modelo Vista Controlador y otros patrones de diseño. Se realizan pruebas funcionales para verificar el correcto funcionamiento del módulo para la gestión de trazas.

PALABRAS CLAVE

Gestión de traza, traza, monitoreo y control, SISEN

TABLA DE CONTENIDOS

Índice de ilustraciones

Índice de tabla

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) e Internet, ha traído aparejado grandes avances para la Informática, y Cuba no está ajena a estos cambios que tienen un alto peso para el desarrollo de las tecnologías y la sociedad. Cuba aprovecha las ventajas que brindan y se ha llevado a cabo un proceso de informatización de la sociedad. La Empresa de Seguros Nacionales de Cuba (ESEN) como parte de este proyecto también está guiada a informatizar sus procesos.

La ESEN forma parte del Grupo Empresarial Caudal (OSDE Caudal ,S.A[1]), perteneciente y adscripto al Ministerio de Finanzas y Precios (MFP[2]). Fue creada mediante la Resolución No. 858/78, el 22 de diciembre de 1978, bajo la denominación de Empresa de Seguro Estatal Nacional. Se creó con el objetivo fundamental de desarrollar las distintas formas de seguros de carácter nacional que brindarán protección a los bienes propiedad de las cooperativas de producción agropecuaria y al patrimonio de la población cubana en general, permitiendo con ello, el resarcimiento de los daños y pérdidas ocurridas a las diferentes economías aseguradas. En el 2004, se modificó el objeto social y se cambió el nombre de la empresa adoptándose al nombre actual. Tiene como objeto empresarial desarrollar operaciones de seguros[3] y reaseguros[4] en moneda nacional y divisa. Las personas aseguradas pueden ser naturales y jurídicas, cubanas o extranjeras. Se realizan actividades preparatorias y complementarias al seguro, dirigidas a la evaluación de riesgos y prevención de daños en moneda nacional y divisa a personas aseguradas. También le compete ofrecer servicios de inspección, tasación y ajustes de averías, cálculos actuariales y prevención del riesgo en bienes asegurados a personas aseguradas, así como al sector agropecuario no asegurado en moneda nacional (ESEN, 2016). La ESEN cuenta con una oficina central, 17 direcciones provinciales de seguro y 169 representaciones territoriales.

La Universidad de las Ciencias Informáticas (UCI) es un centro de estudios universitarios. Nacido como un proyecto de la Revolución Cubana, se creó con los objetivos de informatizar el país y desarrollar la industria del software para contribuir al desarrollo económico. La UCI está compuesta por varios centros productivos, cada uno de ellos se especializa en el desarrollo de sistemas para las diversas esferas sociales. El Centro de Informatización de Entidades (CEIGE) desarrolla productos y servicios asociados a la gestión de entidades. Teniendo en cuenta la importancia de los procesos de la ESEN y la necesidad de resolver los problemas existentes en la gestión, en la UCI, específicamente en CEIGE, se desarrolla el Sistema Integral de Seguros Nacionales (SISEN), una aplicación web compuesta por 21 módulos, con el

propósito de informatizar los procesos que se llevan a cabo en el área comercial de la ESEN y desplegar dicho sistema en una estación central a la cual se conectarán todas las Unidades Empresariales Base (UEB) del país.

Las operaciones que se generan a diario en el SISEN, poseen un ciclo de vida que sufren modificaciones efectuadas por diversos usuarios. Sin embargo, no se registra el histórico de las acciones ejecutadas por los usuarios según sus niveles de acceso, ni las alertas o los errores emitidos por la aplicación; además no existe un control individual por cada usuario con respecto a las acciones que realizan. Esto imposibilita la realización de auditorías o conocer el estado del sistema de forma general, por lo que no se puede identificar con posterioridad cómo recuperar o revertir algún cambio realizado sobre los datos y quién fue el responsable del cambio o problema detectado, lo cual no garantiza un adecuado funcionamiento del sistema.

Por lo tanto, no existe un mecanismo capaz de registrar y analizar los principales eventos ocurridos en el SISEN, que permita a los interesados realizar el monitoreo y el control de estos eventos; y facilite:

- Llevar un control de la actividad de los usuarios dentro del SISEN.
- Detectar irregularidades en los procesos de gestión del SISEN.
- Identificar brechas de seguridad, para dar lugar a un procedimiento que permita mitigar las vulnerabilidades de red detectadas en el SISEN.

Una de las formas de conocer que estos procesos ocurren es mediante las trazas. Según (Arteaga, 2014) las trazas son un mecanismo de registro de eventos y datos, o información sobre quién, qué, cuándo y dónde un evento ocurre, ya sea para un módulo de un sistema en particular o para toda la aplicación. Son utilizadas para dejar un registro de todas las funciones que se realizan en un sistema, así como acreditar las validaciones de datos previstas, sin modificar el sistema en ningún momento. También son usadas con el fin de monitorear las acciones que se realicen (acceso a ficheros, dispositivos, empleo de los servicios, etc.), y para detectar indicios de hechos relevantes a los efectos de la seguridad que puedan afectar la estabilidad o el funcionamiento del sistema informático. Además, rastrean los caminos que siguen los datos a través del programa. Contribuyen al fortalecimiento de la seguridad en las aplicaciones web, el registro de las mismas posibilita que dichas aplicaciones sean auditables, fortaleciendo así la seguridad en las aplicaciones. Según (Martin Frías, y otros, 2014) los resultados de una traza se pueden guardar en un archivo o en una base de datos. Se pueden utilizar los archivos de traza para realizar lo siguiente:

- Auditar la base de datos.

- Realizar análisis de rendimiento.
- Llevar el control de las acciones de cada usuario.
- Optimizar las consultas.

Una vez analizada la situación anterior se formula el siguiente **problema de investigación**:

¿Cómo contribuir al monitoreo y control de las operaciones que se realizan sobre el Sistema Integral de Seguros Nacionales?

Para dar solución al problema planteado se identificó como **objeto de estudio**: El proceso de gestión de trazas.

Teniendo como **Objetivo general:** Desarrollar un módulo de gestión de trazas, que posibilite el monitoreo y control en el Sistema Integral de Seguros Nacionales.

Se enmarca en el **campo de acción:** La gestión de trazas en el SISEN.

Se tiene como **idea a defender** que: Si se desarrolla del módulo de gestión de trazas en el SISEN, entonces se contribuirá al monitoreo y control del mismo.

Para alcanzar el objetivo general se hace necesario realizar los siguientes **objetivos específicos:**

- Elaborar el marco teórico-conceptual para fundamentar la investigación.
- Diseñar la solución de software para facilitar la implementación del módulo para la gestión de trazas del Sistema Integral de Seguros Nacionales.
- Implementar el módulo para la gestión de trazas del Sistema Integral de Seguros Nacionales que permita el monitoreo y control de las operaciones registradas.
- Validar la investigación mediante técnicas y métodos científicos de investigación.
- Validar el correcto funcionamiento del sistema mediante la aplicación de pruebas de software.

Como **posibles resultados** se pretende obtener:

- Un informe detallado con toda la base teórico-práctica sobre que sustenta la solución propuesta.

- Módulo para la gestión de trazas del SISEN, haciendo uso de las tecnologías, lenguajes y herramientas establecidas. Este módulo brindará un adecuado funcionamiento del SISEN mediante las siguientes facilidades:

    - Estarán a disposición del personal autorizado reportes de registros históricos útiles que no están en funcionamiento en la actualidad; que posibilite detectar irregularidades en los procesos de gestión del sistema, identificar brechas de seguridad y llevar un control de la actividad de los usuarios.

    - Se conseguirá mantener un control específico de acuerdo a la interacción de los usuarios con la aplicación, al conocer las trazas dejadas a medida que utilizan el sistema; de modo que permita identificar con posterioridad cómo recuperar o revertir algún cambio realizado sobre los datos y quién fue el responsable del cambio o problema detectado.

La investigación se guiará a través de los siguientes métodos científicos:
Métodos teóricos:

- Analítico – sintético: a partir de un profundo análisis acerca de las teorías, documentación y herramientas existentes sobre el registro y control de las trazas en las aplicaciones de gestión, se obtendrán los elementos más importantes que hacen referencia a las herramientas.

- Inductivo – Deductivo: con la realización de un profundo análisis del trabajo general de las herramientas de registro de trazas, se podrá llegar a conclusiones de su desarrollo e integración a la solución.

- Histórico-Lógico: Permitirá conocer y comprender el estado del arte de los sistemas informáticos existentes en el mundo que poseen funcionalidades similares a la solución propuesta, tendencias actuales y proyecciones futuras, conociendo así su evolución y desarrollo.

Métodos empíricos:

- Observación: El empleo de este método permitirá hacer un análisis detallado de las principales características y funcionalidades del módulo gestión de trazas.

- Entrevista: Se empleará principalmente en la captura de requisitos. Esto constituye uno de los mayores afluentes del conocimiento para la descripción de los requisitos funcionales del módulo gestión de trazas.

El presente trabajo está estructurado de la siguiente forma:

En el capítulo 1, “Fundamentación teórica”, se hace referencia a los principales conceptos teóricos existentes dentro del objeto de estudio, se describe el entorno en el que se desarrolla la problemática, se analizan las posibles soluciones y las características de los sistemas similares. Se describe la metodología usada para el desarrollo de la solución y se exponen herramientas y tecnologías utilizadas en el transcurso de la investigación.

En el capítulo 2, “Análisis y diseño de la solución”, se identifican los requisitos y se realizan las historias de usuarios correspondientes a cada requisito. Se exponen los patrones de diseño y de arquitectura que utilizados en el desarrollo del software y se muestra el modelo de datos del sistema. Además, se valida el diseño mediante las métricas tamaño de clase y relaciones entre clase

En el capítulo 3, “Implementación y validación de la solución”, se describen los estándares de codificación utilizados en el código, además del diseño del diagrama de componentes que guíe la implementación y se prueba la propuesta de solución mediante pruebas de caja blanca y caja negra.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se explicarán un conjunto de conceptos fundamentales que engloban el dominio del problema de la presente investigación. Se realizará un estudio sobre los diferentes sistemas de software existentes para la gestión de trazas, además se describen las herramientas, las tecnologías y la metodología a usar para dar solución al problema planteado.

1.1 Conceptos asociados al dominio del problema

Según (Sacerio Martinez, 2010) un **sistema de gestión de trazas** es aquel que controla los eventos que ocurren dentro de un software determinado y resulta muy importante para su correcto funcionamiento. Un sistema informático que no posea un componente con esta funcionalidad está expuesto a grandes riesgos, entre ellos: la imposibilidad de recuperarse tras una falla del sistema, quedar impune un malhechor en caso de ser quebrantada la seguridad del sistema. Sin las trazas puede ser muy difícil, o en ocasiones imposible, averiguar el motivo del fallo de la aplicación. Por otra parte (González González, 2014) lo define como: es aquel que registra y permite auditar los eventos que ocurren dentro de un sistema determinado y es muy importante para su correcto funcionamiento. Un producto informático que no posea un componente con esta funcionalidad implica grandes riesgos como el control de las funciones informáticas de la empresa y/o institución, cumplimiento de las normativas generales de la empresa y/o institución, además de el análisis de los controles y procedimientos tanto organizativos como operativos. Un sistema de gestión de trazas es aquel que permite controlar y obtener las trazas ocurridas en el sistema, posibilitando seguridad en la información y mejor rendimiento del mismo.

Según (Corrales Martinez, 2009) las **trazas** son una colección de eventos y datos devueltos por una aplicación, que representan el historial o rastro dejado por cierto proceso cuando se ejecuta en un sistema. Así mismo, (Martin Frías, y otros, 2014) definieron que las trazas se gestionan en cualquier tipo de sistema donde se requiera hacer una revisión de los eventos, errores cometidos y las acciones realizadas por los usuarios dentro del mismo. Además permiten crear un registro de sucesos para un dispositivo o aplicación. A través de estos datos, se puede monitorear el funcionamiento de la aplicación y se ofrece una oportunidad para corregir problemas de seguridad. Según (Delgado Delgado, 2013) definió : las trazas son las huellas dejadas por los usuarios cuando interactúan con un sistema informático, o sea, son el camino que indica por donde estuvo el usuario en el sistema, a qué lugares accedió, si realizó algún pedido de datos, qué hizo con ellos. Estas se utilizan para verificar el funcionamiento del software y para detectar posibles errores que aparezcan en los sistemas. Las trazas se utilizan para comprobar las

validaciones de los datos que fluyen en el software durante su uso, estas pueden ayudar a detectar posibles ataques que pueden recibir las aplicaciones ya sea, por la red o por la interacción de algún usuario con el software. Se califican en varios tipos, entre los cuales se destaca las de error, alerta, información y debug[5]. En el desarrollo de software, las trazas permiten crear un registro de sucesos para un dispositivo o aplicación. A través de estos datos se puede monitorear el funcionamiento de la aplicación y se ofrece una oportunidad para corregir problemas de seguridad. Las trazas son el historial o registro de los eventos realizados en un sistema y tiene como objetivos brindar información importante de los eventos y poder así realizar procesos de monitoreo y control y constribuir a la seguridad del sistema.

Por su parte, la Organización Internacional para la Estandarización (ISO) la define en su *International Vocabulary of Basic and General Terms in Metrology* a la **trazabilidad** como: “La propiedad del resultado de una medida o del valor de un estándar donde este pueda estar relacionado con referencias especificadas, usualmente estándares nacionales o internacionales, a través de una cadena continua de comparaciones todas con incertidumbres especificadas”. Según la Real Academia Española, trazabilidad son: “aquellos procedimientos preestablecidos y autosuficientes que permiten conocer el histórico, la ubicación y la trayectoria de un producto o lote de productos a lo largo de la cadena de suministros en un momento dado, a través de herramientas determinadas”. Por otra parte, el numeral 7.5.3 “Identificación y Trazabilidad” de la norma ISO 9001:2008 establece que: “La organización identifica el producto por medios adecuados a través de toda la realización del producto. Se identifica el estado del producto con respecto a los requisitos de seguimiento y medición. Se controla y registra la identificación única del producto”. Se puede decir que la trazabilidad es la capacidad de recopilar información relacionada con el ciclo de vida desde el origen hasta el estado final de un producto, sean estos modelos de plantillas, datos informáticos entre otros. Dicha trazabilidad consiste en asociar sistemáticamente un flujo de datos a un flujo físico de productos de manera que se pueda relacionar en un momento dado la información requerida relativa a los productos determinados (González González, 2014). Mientras que (Martin Frías, y otros, 2014) plantea que es la capacidad que tiene una organización o sistema para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos.

La trazabilidad son los procesos que se realizan para obtener la información de las trazas que permita a los interesados llegar a una conclusión de qué ocurrió con el producto u objeto sobre el que se realizó alguna acción.

¿Por qué es importante la trazabilidad?

El seguimiento y rastro de la información supone una serie de beneficios y mejoras prácticas en el manejo de datos, trayendo consigo que la disponibilidad e integridad de la misma sea una tarea primordial. Se puede afirmar que todos los eslabones se beneficiarán del proceso de trazabilidad, ya que supone (Hernández León, y otros, 2011):

- Control individualizado de cada agente involucrado.
- Mejora de la gestión de la información.
- Permite detectar, acotar y analizar problemas con gran celeridad.

Existen aspectos fundamentales a tener en cuenta en una traza o registro de seguimiento, ellos son:

- Qué: registrar qué información fue gestionada y por qué acción.
- Quién: registrar quién gestionó la información.
- Cuándo: registrar la fecha y la hora en la que se gestionó la información.
- Dónde: registrar dónde se gestionó la información.
- Información trazabilidad: registrar cierta información que describa con facilidad de entendimiento algún evento o error en específico.

Un sistema de trazabilidad bien implantado permite, en caso de una crisis, acortar el tiempo de reacción, lo que disminuye los costos y la producción a retirar. Puede que se registre toda la información necesaria para resolver cierto problema, pero esto no basta para dar solución a la situación en cuestión. Se debe contar también con una vía de facilidad de entendimiento de este conjunto de información relevante, algo que se hace casi imposible en un volumen de cientos de registros de información a simple vista para un humano. La trazabilidad permite también la realización de auditorías y llevar un monitoreo y control de los sistemas, es por eso que se hace necesario conocer los conceptos de los mismos.

El vocablo de **auditoría** es sinónimo de examinar, verificar, investigar, consultar, revisar, comprobar y obtener evidencias sobre informaciones, registros, procesos y circuitos. Hoy en día, la palabra auditoría se encuentra relacionada con diversos procesos de revisión o verificación que, aunque todos ellos tienen en común el estar de una u otra forma vinculados a la empresa, pueden diferenciarse en función de su finalidad económica inmediata, de tal manera que según este criterio se puede establecer una primera gran clasificación de la auditoría diferenciando entre auditoría económica y auditorías especiales (De la Peña Gutierrez, 2007). Según (Martin Frías, y otros, 2014) es la capacidad de un sistema de registrar eventos y rastrear la actividad del usuario mientras accede a los recursos, soportando sistemas de trazas. A través de la auditoría se detectan irregularidades en un sistema (indicando qué ha pasado, sobre qué información y quién ha accedido; formando parte imprescindible de cualquier sistema que pretenda

proporcionar seguridad a la información). En efecto auditoría es la manera de verificar el correcto funcionamiento de algún sistema aplicando distintas técnicas, evitando irregularidades que podrían afectar el rendimiento del mismo.

Según (Marín Sánchez, y otros, 2016) el proceso de **monitoreo y control** de proyectos está compuesto por aquellas actividades requeridas para supervisar, analizar y regular el progreso y el desempeño del proyecto, para identificar áreas en las que el plan requiera cambios y para iniciar los cambios correspondientes. Brevemente, el monitoreo y control son acciones o sucesos que se realizan para llevar un seguimiento de algún proceso y comprobar que todo marcha según lo planeado.

1.2. Estudio de sistemas informáticos para la captura y gestión de trazas a nivel nacional e internacional.

Para la realización del módulo para la gestión de trazas en el SISEN se realizó un estudio de algunos de los sistemas que existen tanto a nivel nacional como internacional. A continuación, se mencionan los sistemas analizados con sus principales características:

1. Sauxe

Es un marco de trabajo desarrollado en la UCI que integra en su núcleo un conjunto de tecnologías como *Zend Framework* 1.11[6], *Doctrine* 1.2[7], *ExtJS* 2.2[8]. Este brinda a los desarrolladores una gran variedad de funcionalidades para facilitar el desarrollo de software. Entre ellas cuenta con un componente dedicado específicamente a la gestión de las trazas. Este componente brinda la posibilidad de registrar diferentes tipos de trazas, así como la visualización de las mismas (Cañete Pupo, 2010).

El estudio del componente de trazas de Sauxe permitió identificar algunos de los tipos de trazas que se registrarán, además posibilitó establecer relaciones entre las trazas que serán útiles en el diseño del modelo de datos.

2. WebSpy Vantage

WebSpy Vantage emplea un módulo web que permite a los administradores distribuir de forma segura los informes de una organización. Con este software las organizaciones pueden monitorear y dar a conocer, no sólo el uso de Internet y el comportamiento de navegación web. También el uso del correo electrónico, la mensajería instantánea, los registros de eventos, los routers, el tráfico de visitantes en el sitio web, firewalls[9], antivirus y aplicaciones anti-spam[10]. Además, puede identificar el mal uso de internet y correo electrónico, resolver problemas de la red y los problemas de descarga. El fortalecimiento de los controles de seguridad preserva el ancho de banda reduciendo al mínimo los riesgos de litigios, y la identificación del origen de la pérdida de datos o daños. La funcionalidad principal consiste en importar texto o registros

(logs[11]) desde una amplia variedad de productos de otros proveedores, o de sucesos de Windows (Webspy, 2017).

Las ventajas que tiene WebSpy Vantage es que es rápido para investigar problemas, ahorra el tiempo, genera informes rápidamente y tiene un buen rendimiento ya que el seguimiento de la productividad de los empleados y la posterior generación de informes se hará a través de un procedimiento automatizado, incrementando así la eficiencia. Otra ventaja es que se pueden crear distintos informes para cada usuario, departamento, fecha o cualquier otro dato de validación. También se pueden estimar los tiempos individuales de la sesión, los tiempos totales o promedios de navegación, el tamaño promedio de descarga, número de usuarios que accede a un sitio y las ocasiones en que un usuario visita en el día entre otras actividades. No se utilizará ya que WebSpy Vantage es un software de carácter privativo diseñado para correr sobre los sistemas operativos Microsoft Windows, Server 2003 SP1, Vista, Server 2008 y Windows 7. Protegido por la compañía de servicios WebSpy, establecida desde 1996 con oficinas estratégicamente ubicadas en los Estados Unidos, el Reino Unido y Australia.

3. Sawmill

Está especialmente diseñado para analizar cualquier tipo de registro de acceso a servidores web. Se ejecuta como un programa que regula el intercambio de información entre clientes web y las aplicaciones que se ejecutan en el servidor, o sea, lo que se conoce como un programa CGI2[12]. Este puede utilizarse desde cualquier navegador para configurar y ejecutar Sawmill o para ver estadísticas de páginas, funcionalidad que podría abrir agujeros de seguridad en el servidor, ya que una aplicación en CGI mal diseñada podría permitir acceso total o parcial al servidor. Las estadísticas son jerárquicas, atractivas y poseen enlaces que facilitan la navegación. El programa incluye una completa documentación. Este software ofrece una gran cantidad de opciones, incluida una base de datos persistente, el control sobre la apariencia de las páginas de estadísticas y diversas opciones de filtrado sobre el registro. Muestra, tras su instalación, una interfaz amigable en Windows Internet Explorer y presenta, en un cuadro de selección de opciones ubicado a la izquierda, una serie de estadísticas posibles:

1. Cantidad de visitas por hora, por día, por mes, etcétera.
2. Horas pico y horas de baja audiencia.
3. Páginas más visitadas.
4. Páginas de entrada y salida más frecuentes del sitio.
5. Utilización de buscadores, clasificación de palabras clave empleadas para buscar (Sawmill, 2017).

La desventaja principal es que el producto está registrado con licencia y protegido por las leyes de copyright[13] de Estados Unidos y las disposiciones de tratados internacionales. Por lo tanto, no se podrá utilizar el producto como solución al problema.

4. Webalizer

Es una herramienta gratuita de análisis de archivos de registro que genera resúmenes de utilización muy detallados acerca de servidores Web y FTP en formato gráfico y tabular de fácil comprensión. Los registros son un resumen estadístico del tráfico de usuarios en su servidor. Los informes de Webalizer se generan en horarios programados de acuerdo con las opciones de configuración que usted estableció para el sitio al momento de crearlo (Webalizer, 2014). Este programa fue ideado para correr sobre la plataforma libre, puede ser redistribuido y/o modificado bajo los términos de la Licencia Pública General de GNU, ya sea la versión 2 de la licencia, o cualquier versión posterior, y siempre que el aviso de copyright anterior y el permiso se incluya con todas las copias distribuidas de este software o derivados.

5. AWStats (Advanced Web Statistics o Estadística Web Avanzada)

Es una herramienta de análisis open source escrita en Perl, que permite parsear logs de un servidor web, ftp o de correo y generar estadísticas en forma de reportes HTML. Puede ejecutarse desde un navegador web o desde la línea de comandos y funciona sobre diversos sistemas operativos. Un análisis del registro completo permite mostrarles la siguiente información: el número de visitas y de visitantes únicos y la duración de las visitas, los usuarios autenticados y últimas visitas autenticadas, las páginas más vistas, los navegadores utilizados (páginas, hits, kilo-byte para cada navegador), OS (Operative System) utilizado (páginas, hits, kilo-byte para cada sistema operativo), los motores de búsqueda, frases y palabras clave utilizadas para encontrar su sitio y los errores HTTP (2018).

Después de analizar las soluciones existentes a nivel nacional e internacional ninguna daría solución a la problemática existente pues hay herramientas privadas (Sawmill, WebSpy Vantage) y esto traería mucho gasto a la empresa. Estas herramientas privadas funcionan en sistemas operativos privados otro elemento por el cual no se puede utilizar y la empresa está enfrascada en el desarrollo y uso de software libre. AWStats y Webalizer no son sistemas web y no están orientados a usuarios. Estos sistemas son adecuados para llevar estadísticas generales para empresas e instituciones, para llevar el monitoreo de sitios en específicos, sin embargo, no es funcional para mostrar informaciones de los usuarios durante su navegación. Y en Sauxe se encontraron funcionalidades que se desarrollarán en el módulo.

1.3 Metodología de desarrollo de software

Para la realización del módulo se llevó a cabo una metodología de desarrollo de software que es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo. Es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema informático (Romeu, 2015). Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, espiral entre otros). Adicionalmente debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto y guías para uso de herramientas de apoyo (Miranda, 2015). En otras palabras, una metodología es un marco de trabajo que sirve de guía para el desarrollo del software siguiendo una estructura.

La metodología que se va a utilizar en el módulo de gestión de trazas para el SISEN es la metodología Proceso Unificado Ágil de *Scott Ambler* o Agile *Unified Process* (AUP) para la UCI, esta describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en (RUP[14]). El AUP según (Rodríguez Sánchez, 2014) aplica técnicas ágiles incluyendo:

- Desarrollo dirigido por Pruebas.
- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce por tal motivo AUP-UCI propone las siguientes fases según (Rodríguez Sánchez, 2014):

- Inicio: 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.
- Ejecución: Durante esta fase se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Según (Rodríguez Sánchez, 2014) AUP-UCI propone las 7 disciplinas siguientes:

1. Modelado del negocio: Es la disciplina destinada a comprender los procesos de negocio de una organización.
2. Requisitos: Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
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. Además, 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 específico que el de análisis.
4. Implementación: En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
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.
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.
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.

En la descripción de disciplina existen cuatro escenarios para modelar el sistema de los proyectos de los cuales el que se va a utilizar en el desarrollo de la solución es el cuatro, puesto que se ajusta para proyectos que no modelan un negocio, sino modelan el sistema con las HU, siendo éste el caso que ocupa. Además, el módulo no es muy extenso y las HU permiten agilizar el desarrollo del software debido a que generalmente se escriben desde la perspectiva del usuario y el contenido de estas debe ser concreto y sencillo. También se utilizan para hacer estimaciones de tiempo de la implementación de cada requisito identificado, de manera que brinda con mayor precisión y objetividad el tiempo de ejecución de un sistema de software en pequeñas porciones.

1.4 Herramientas y tecnologías

Visual Paradigm 8.0

Visual Paradigm es una herramienta de diseño UML[15] libre y profesional, diseñada para contribuir al desarrollo de software. Soporta los principales estándares como UML, BPMN[16] y XML[17]. Ofrece un completo conjunto de herramientas a los equipos de desarrollo de software, necesarios para la captura de requisitos, la planificación de software, la planificación de pruebas, el modelado de clases y el modelado de datos (Morales, y otros). Según (Ferrer Oduardo, 2013) Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computadoras (CASE por sus siglas en inglés). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista, que fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque orientado a objetos. Se utiliza Visual Paradigm porque es una herramienta gratuita que facilita el modelado diagramas como Entidad-Relación, realizar prototipos de interfaz de usuario y el diseño de artefactos. Permite exportar los diagramas como imágenes en varios formatos y posee una interfaz amigable.

Java JDK 7

Para desarrollar el módulo se utilizó como lenguaje de programación **Java JDK 7** porque es un software que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red. En la unidad de red se pueden tener las herramientas distribuidas en varias computadoras y trabajar como una sola aplicación.

Las características de Java resultan muy interesantes, es en un programa Java poco voluminoso, su descarga desde internet requiere poco tiempo. Un programa Java es portable y se puede utilizar sin modificaciones en cualquier plataforma.

Características de Java:

- Java es a la vez un lenguaje y una plataforma de desarrollo.
- Es robusto, portable, eficaz, multitarea y dinámico (Groussar, 2012).

Es un lenguaje de programación muy utilizado en cuanto al desarrollo de herramientas y aplicaciones de negocio. Es rápido, seguro, fiable, su distribución es gratuita, brinda soporte a las técnicas de desarrollo de la Programación Orientado a Objetos (POO) y a la reutilización de componentes de software. Otro de los factores más importantes de Java es que no se quiebra fácilmente ante errores de programación, no es posible escribir en áreas arbitrarias de memoria ni realizar operaciones que corrompan el código. Puede aplicarse a la realización de aplicaciones en las que ocurra más de una cosa a la vez lo que se denomina multihilos (Fonseca Vega, y otros, 2015).

Se utiliza Java porque funciona en cualquier sistema operativo que tenga instalada la máquina virtual para este lenguaje, es una plataforma de desarrollo, está orientado a objeto. La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor y es capaz de conectar dos o más computadoras u ordenadores ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar.

Framework Spring v 4.3.6

El Spring Framework (también conocido como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. Se va a utilizar porque se ha popularizado en la comunidad de programadores en Java al ser considerado como una alternativa y sustituto del modelo de *Enterprise Java Bean*[18]. Su diseño ofrece mucha libertad a los desarrolladores en Java y posee soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria (Fuentes Torrubia, 2011). Spring está diseñado para no ser intrusivo, lo que significa que su código de lógica de dominio generalmente no tiene dependencias en el marco mismo. En su capa de integración (como la capa de acceso a datos), algunas dependencias en la tecnología de acceso a datos y las bibliotecas de Spring existirán. Sin embargo, debería ser fácil de aislar estas dependencias del resto de su base de código (Johnson, 2013).

Se utiliza Spring porque es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. Permite instanciar, inicializar y conectar objetos de la aplicación y el contenedor de Spring le proporciona un mecanismo consistente para configurar las aplicaciones, y se integra con casi todos los entornos Java, desde aplicaciones pequeñas a grandes aplicaciones empresariales.

ORM Hibernate v 5.2.8

Es una herramienta de Mapeo Objeto-Relacional (ORM) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos

(XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones (Fuentes Torrubia, 2011). Según (Finogeev, y otros, 2017) la biblioteca de Hibernate resuelve los problemas de las clases de Mapeo relacional de objetos para comunicarse con las tablas de la base de datos, los tipos de datos de Java a los tipos de datos de SQL. Proporciona herramientas para generar y actualizar un conjunto de tablas en el modelo de datos, creación de consultas SQL y procesamiento de respuestas. El uso de esta herramienta permite reducir el tiempo de configuración típica del sistema y proporciona la interfaz de la aplicación a cualquier base de datos SQL.

Se utiliza Hibernate porque es flexible en cuanto al esquema de tablas utilizado y puede adaptarse a su uso sobre una base de datos ya existente. Permite detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la programación orientada a objetos.

ExtJS 6.5

Es una librería Java Script[19] open-source[20] de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. 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[21], distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note (Sencha, 2018). Se va a utilizar ya que permite realizar interfaces de usuario, fáciles de usar, muy parecidas a las conocidas aplicaciones de escritorio, posibilitando concentrarse en la funcionalidad de las aplicaciones en vez de en las advertencias técnicas.

Sencha CMD 6.5.0

Sencha Cmd (Sencha, 2018) es una herramienta de línea de comandos multiplataforma que proporciona muchas tareas automatizadas en todo el ciclo de vida de sus aplicaciones, desde la generación de un nuevo proyecto hasta la implementación de una aplicación en la producción. Ofrece una colección de características potentes que ahorran tiempo y que trabajan juntas y en conjunto con los frameworks Sencha Ext JS y Sencha Touch. Sencha Cmd proporciona las siguientes capacidades:

1. Herramientas de generación de código.

2. Compilador JS.
3. Servidor web.
4. Integración de Sencha Web Application Manager.
5. Gestión del espacio de trabajo.
6. Herramientas de sintonización.
7. Sistema de configuración flexible.
8. Software de terceros.
9. Ganchos de generación de código.

PostgreSQL 9.2

PostgreSQL es un sistema de gestión de bases de datos Objeto-Relacional, distribuido bajo licencia BSD[22] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Incluye características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos. PostgreSQL permite realizar múltiples conexiones desde procesos clientes, existiendo un proceso maestro en el servidor que siempre se ejecuta y está a la espera de nuevas conexiones clientes, de forma tal que cuando alguien se conecta, se inicia un nuevo proceso, asegurando que la nueva conexión no necesite del proceso postgres original, por lo que una de sus principales características es la alta concurrencia (Vukotic, y otros). Según (Ribiaux, 2013) es un servidor de base de datos relacional libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados y almacenamiento de objetos de gran tamaño. Se destaca por ejecutar consultas complejas, consultas sobre vistas y subconsultas. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Cuenta con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y es multiplataforma.

Se utiliza por su código fuente que es gratuito y es multiplataforma, utiliza un modelo Cliente- Servidor, incluye características como la orientación a objetos, como puede ser la herencia, tipo de datos, funciones, restricciones y disparadores. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo.

Apache Tomcat 8.5.11

El servidor Apache Tomcat es de código abierto, basado en Java contenedor de aplicaciones web que se creó para ejecutar aplicaciones web. Fue creado bajo el Apache-Jakarta subproyecto. Sin embargo, debido a su popularidad, ahora se aloja como un proyecto de Apache separado, apoyado y mejorado por un grupo de voluntarios de la comunidad Java de código abierto. Apache Tomcat es muy estable y tiene todas las características de una aplicación web comercial, sin embargo, viene bajo la licencia de Apache de código abierto. Tomcat también proporciona una adición, toda la funcionalidad que hace es una gran opción para desarrollar una solución de aplicación web completa (Vukotic, y otros). Según (Oliva Perez, y otros, 2013) es un servidor gratuito de código abierto, ofrece un servicio estable y sencillo de mantener y configurar. Se ejecuta en varios sistemas operativos, haciéndolo esta característica prácticamente universal. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es altamente configurable en la creación y gestión de logs.

Se utiliza porque es un servidor de código abierto. Corre en una gran cantidad de sistemas operativos. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.

Mozilla Firefox 36.0.1

Mozilla Firefox es un navegador web libre y de código abierto desarrollado por Mozilla, una comunidad global que trabaja junta para mantener una Web Abierta, pública y accesible para todos. Su primera versión vio la luz el 9 de noviembre de 2004 y hasta la fecha, Firefox ha revolucionado la forma de pensar y mantenido la innovación en la web para llevar a sus usuarios una mejor experiencia. Entre sus méritos recae el orgullo de ser uno de los proyectos de Software Libre más importantes del mundo.

Cuenta con varias características entre las que se encuentran: navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas, navegación privada, aceleración mediante GPU[23] y muchas otras más. También mediante complementos se puede extender añadiéndole funcionalidades que no trae. Además, es 100% compatible con estándares web, manteniendo así la elección e innovación en la web (Mozilla, 2014). Se utiliza porque presenta una forma rápida y eficiente de navegar por la web, que permite abrir varias páginas en una misma ventana mediante el empleo de pestañas separadas. Contiene funcionalidades en las herramientas de desarrollo como el registro del servidor en la consola web, la búsqueda rápida de la regla sobre escrita en la declaración del CSS y la opción estricta para filtrado en las reglas de la vista y barra lateral UI (interfaz de usuario).

1.4 Conclusiones parciales

Luego de haber realizado un estudio sobre los principales conceptos del tema, así como de las herramientas existentes a nivel nacional e internacional, la metodología a utilizar, además de las herramientas y tecnologías para facilitar el desarrollo del módulo, se arriban a las siguientes conclusiones:

1. La gestión de trazas permite conocer datos del sistema que pueden ser utilizados en el monitoreo y control, contribuye a la seguridad del sistema y mejora la gestión de información.
2. Debido a sus características las soluciones existentes no pueden ser integradas al proyecto SISEN, pero existen funcionalidades de las mismas que pueden ser utilizadas en el módulo de gestión de trazas que se implementará.
3. Para el desarrollo del módulo de gestión de trazas para el SISEN se utilizó la metodología AUP – UCI en el escenario 4 ya que las HU permiten agilizar el desarrollo de software por su contenido concreto y sencillo, esto con el objetivo de aumentar la calidad del software.
4. El módulo se desarrollará con tecnologías y herramientas de código abierto, multiplataforma y encaminadas a la programación orientada a objetos garantizando la soberanía tecnológica.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción al capítulo

En el presente capítulo se realiza una descripción detallada de las características de sistema. Se describen brevemente los requisitos funcionales y no funcionales identificados, también como se validan los mismos. Se muestra también los artefactos generados y los patrones de diseño teniendo en cuenta la metodología seleccionada. Se define un modelo de datos con su respectivo diccionario de datos para comprender el entorno en el que trabaja el módulo y se expone la arquitectura del sistema. Para validar el diseño se utilizó las métricas tamaño operacional y relación entre clases.

2.1 Requisitos del sistema

Los requisitos del sistema establecen con detalle las funciones, servicios y restricciones operativas del sistema. El documento de requisitos del sistema (algunas veces denominado especificación funcional) debe ser preciso. Debe definir exactamente qué es lo que se va a implementar. Puede ser parte del contrato entre el comprador del sistema y los desarrolladores del software (Somerville, 2005). Según (García Peñalvo, 2018) un requisito es una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal. Los requisitos del sistema son funcionalidades, servicios o cualidades que debe poseer el sistema con el fin de cumplir un objetivo o resolver un problema.

2.1.1 Técnicas de obtención de requisitos

Entrevistas al cliente

Según (Somerville, 2005) las entrevistas formales o informales con el cliente del sistema son parte de la mayoría de los procesos de la ingeniería de requisitos. En estas entrevistas, el equipo de la ingeniería de requisitos hace preguntas al cliente sobre el sistema que utilizan y sobre el sistema a desarrollar. Los requisitos provienen de las respuestas a estas preguntas. Las entrevistas pueden ser de dos tipos:

1. Entrevistas cerradas donde el cliente responde a un conjunto predefinido de preguntas.
2. Entrevistas abiertas donde no hay un programa predefinido. El equipo de la ingeniería de requisitos examina una seria de cuestiones con el cliente del sistema y, por lo tanto, desarrolla una mejor comprensión de sus necesidades.

Las entrevistas sirven para obtener una comprensión general de lo que hace el cliente, cómo podría interactuar con el sistema y las dificultades a las que se enfrentan con los sistemas actuales. A la gente le

gusta hablar sobre su trabajo y normalmente se alegran de verse implicados en las entrevistas (Somerville, 2005).

Se aplicó una entrevista para obtener los requisitos en la cual se le hizo preguntas al cliente sobre el sistema que utilizan y sobre el sistema que se va a desarrollar. Se utilizó una mezcla de entrevista cerrada con entrevista abierta donde el cliente respondió a un conjunto predefinido de preguntas, pero también se examinó una serie de cuestiones como funcionalidades y servicios del sistema con el cliente, para desarrollar una mejor solución con respecto a sus necesidades. La encuesta aplicada se encuentra en el anexo.

Se realizaron sesiones de tormentas de ideas y participaron analistas, programadores, arquitectos e integrantes del proyecto. La tormenta de ideas consiste en realizar reuniones en grupo con el objetivo de generar una gran variedad de vistas del problema y formularlo de diferentes formas.

2.1.2 Requisitos funcionales

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requisitos. Cuando se expresan como requisitos del usuario, habitualmente se describen de una forma bastante abstracta. Sin embargo, los requisitos funcionales del sistema describen con detalle la función de éste, sus excepciones y, además, sus entradas y salidas (Somerville, 2005). Brevemente, los requisitos funcionales son funcionalidades ya sean cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone que debe tener el sistema.

Tabla 1 Requisitos funcionales

No.	Requisitos funcionales	Agrupaciones
1	Adicionar Tipo de Objetos	Gestionar Tipo de Objetos
2	Modificar Tipo de Objetos	
3	Eliminar Tipo de Objetos	
4	Listar Tipos de Objetos	
5	Buscar Tipos de Objetos	
6	Adicionar Operación	Gestionar Operación
7	Modificar Operación	

8	Eliminar Operación	
9	Listar Operaciones	
10	Buscar Operación	
11	Adicionar módulo	Gestionar módulo
12	Modificar módulo	
13	Eliminar módulo	
14	Listar módulo	
15	Buscar módulo	
16	Adicionar configuración	Gestionar configuración
17	Modificar configuración	
18	Eliminar configuración	
19	Listar configuración	
20	Buscar configuración	
21	Listar traza	Traza
22	Buscar traza	

2.1.3 Requisitos no funcionales

Los requisitos no funcionales, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema (Somerville, 2005). Son requisitos no relacionados directamente con la funcionalidad del sistema, pueden estar relacionados con propiedades emergentes del sistema, pueden describir restricciones al producto a desarrollar y describir restricciones externas del sistema. Definen las cualidades globales que el sistema ha de exhibir, suelen hacer referencia al sistema considerado de forma global, suelen ser requisitos más críticos que los requisitos funcionales y difíciles de verificar (García Peñalvo, 2018). Los requisitos no funcionales son todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento. Son las restricciones o condiciones que impone el cliente al programa que necesita.

Tabla 2 Requisitos no funcionales

No.	**Requisitos no funcionales**	**Atributo de calidad**

1	El módulo está creado para brindar una interfaz de poca complejidad, permitiendo dar la menor posibilidad de dudas en cuanto al cómo usarlo.	Usabilidad
2	El módulo podrá ser usado por personas capacitadas con los conocimientos necesarios para interactuar con el mismo.	Usabilidad
3	El módulo debe estar disponible las 24 horas, se desconectará en caso de mantenimiento.	Fiabilidad
4	El usuario deberá autenticarse para entrar al sistema mediante usuario y contraseña válidos.	Seguridad
5	El sistema debe mostrar de forma organizada las funcionalidades del sistema.	Interfaz
6	Los botones utilizados en la aplicación deben tener un tamaño moderado y su nombre entendible para el usuario.	Interfaz
7	El módulo será multiplataforma.	Portabilidad
8	La información que se maneje en el módulo estará protegida de acceso no autorizado y divulgación, a partir de los diferentes roles de los usuarios que utilicen el mismo.	Confidencialidad
9	La información manejada por el componente será objeto de protección contra corrupción, de igual manera el origen y autoridad de los datos debido a la importancia que tienen los mismos.	Integridad

2.1.4 Validación de requisitos

La validación de requisitos trata de mostrar que estos realmente definen el sistema que el cliente desea. Coincide parcialmente con el análisis, este implica encontrar problemas con los requisitos. La validación de requisitos es importante debido a que los errores en el documento de requisitos pueden conducir a importantes costes al repetir el trabajo, cuando son descubiertos, durante el desarrollo o después de que el sistema esté en uso. El costo de arreglar un problema en los requisitos haciendo un cambio en el sistema es mucho mayor que reparar los errores de diseño o los de codificación. Un cambio en los requisitos normalmente significa que el diseño y la implementación del sistema también deben cambiar y que este debe probarse nuevamente (Somerville, 2005). Según (García Peñalvo, 2018) intentan asegurar

que los requisitos verificados reflejan realmente las necesidades de clientes y usuarios. Existen técnicas para la validación de requisitos como la revisión técnica formal y el prototipado de interfaz de usuario. Según (Fernandez Rabilero, y otros, 2016) el prototipado de interfaz: permite al cliente entender fácilmente la propuesta de solución, al brindar la representación aproximada de la interfaz de usuario que tendrá el sistema.

Existen dos tipos principales de prototipo de interfaz:

Desechables: se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en software.

Evolutivos: una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final.

Una revisión técnica formal es un proceso manual que involucra a personas tanto de la organización del cliente como de la del contratista. El cliente verifica el documento de requisitos en cuanto a anomalías y omisiones. El proceso de revisión se puede gestionar de la misma forma que las inspecciones de programas. Alternativamente, se puede organizar como una actividad más amplia con diferentes personas que verifican diferentes partes del documento (Somerville, 2005).

Como técnicas de validación de requisitos se utilizaron las revisiones técnicas formales y el prototipado de interfaz. El tipo de prototipo de interfaz utilizado para la propuesta de solución fue evolutivos, de esta forma se reutiliza el prototipo diseñado en todo el proceso de desarrollo del sistema.

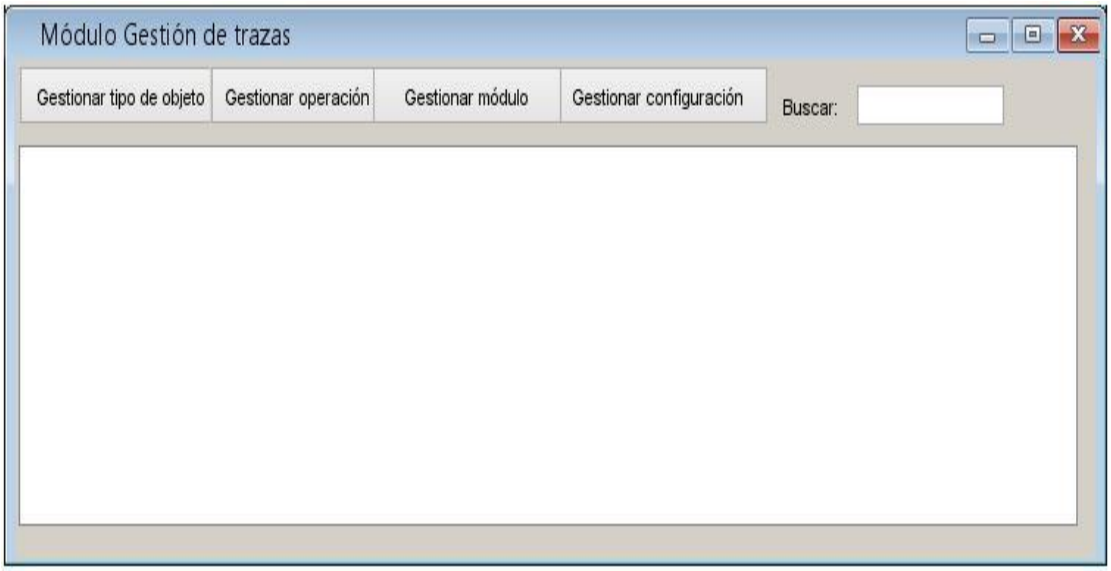
2.2 Análisis y diseño

2.2.1 Historia de usuario

Una historia de usuario es una representación que toma varias semanas desarrollar, es necesario planear en detalle cada una de estas historias de usuario descomponiendo la misma en pequeñas tareas. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos (acompañadas de las discusiones con los usuarios y las pruebas de validación). Cada historia de usuario debe ser limitada, esta debería poderse escribir sobre una nota adhesiva pequeña. Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiante (Sánchez, 2017). Según (Suaza, 2015) es el resultado de conversaciones entre los interesados del proyecto, los analistas de negocios, los encargados de pruebas y los desarrolladores. Una historia de usuario es una representación de un requisito de software escrito en una o dos frases utilizando el lenguaje común del interesado. Las

historias de usuario se utilizan en las metodologías de desarrollo ágil para la especificación de requisitos (acompañadas de las discusiones con los usuarios y las pruebas de validación). En otras palabras, una historia de usuario es utilizada en metodologías ágiles para administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales sin requerir de mucho tiempo para administrarlos y permite responder rápidamente a los requisitos cambiantes.

Tabla 3 Historia de usuario de requisito listar traza

Número: HU-21	**Nombre del requisito:** Listar traza
Programador: Alexis Roberto Valle Mazorra	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 5 días
Descripción: Se muestra un listado de las trazas de los eventos que se han ejecutado en el SISEN con los campos usuario, tipo de objeto, operación y módulo.	
Prototipo de interfaz: 	

2.2.2 Diseño arquitectónico

El diseño arquitectónico representa la estructura de los datos y de los componentes del programa que se requieren para construir un sistema basado en computadora. Considera el estilo de arquitectura que adoptará el sistema, la estructura y las propiedades de los componentes que lo constituyen y las interrelaciones que ocurren entre sus componentes arquitectónicos (Pressman, 2010).

El estilo arquitectónico o el patrón de arquitectura de Software empleado para el desarrollo de la aplicación es la arquitectura MVC. Este separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento (Torres Garces, y otros, 2015).

El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones.

El Controlador: Responde a eventos e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información.

La Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

En este sistema está reflejado el patrón de arquitectura Modelo Vista Controlador (MVC), el cual separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El estilo de llamada y retorno MVC se ve frecuentemente en aplicaciones web, donde la Vista que es la página HTML, maneja la visualización de la información y el código que provee de datos dinámicos a la página, separa presentación e interacción de los datos del sistema, define y gestiona cómo se presentan los datos al usuario. El componente Modelo maneja los datos del sistema y las operaciones asociadas a esos datos. El componente Controlador hace de intermediario entre la Vista y el Modelo ya que responde a eventos e invoca peticiones al Modelo cuando se hace alguna solicitud sobre la información y envía comandos a su Vista asociada si se solicita un cambio en la forma en que se presenta el Modelo.

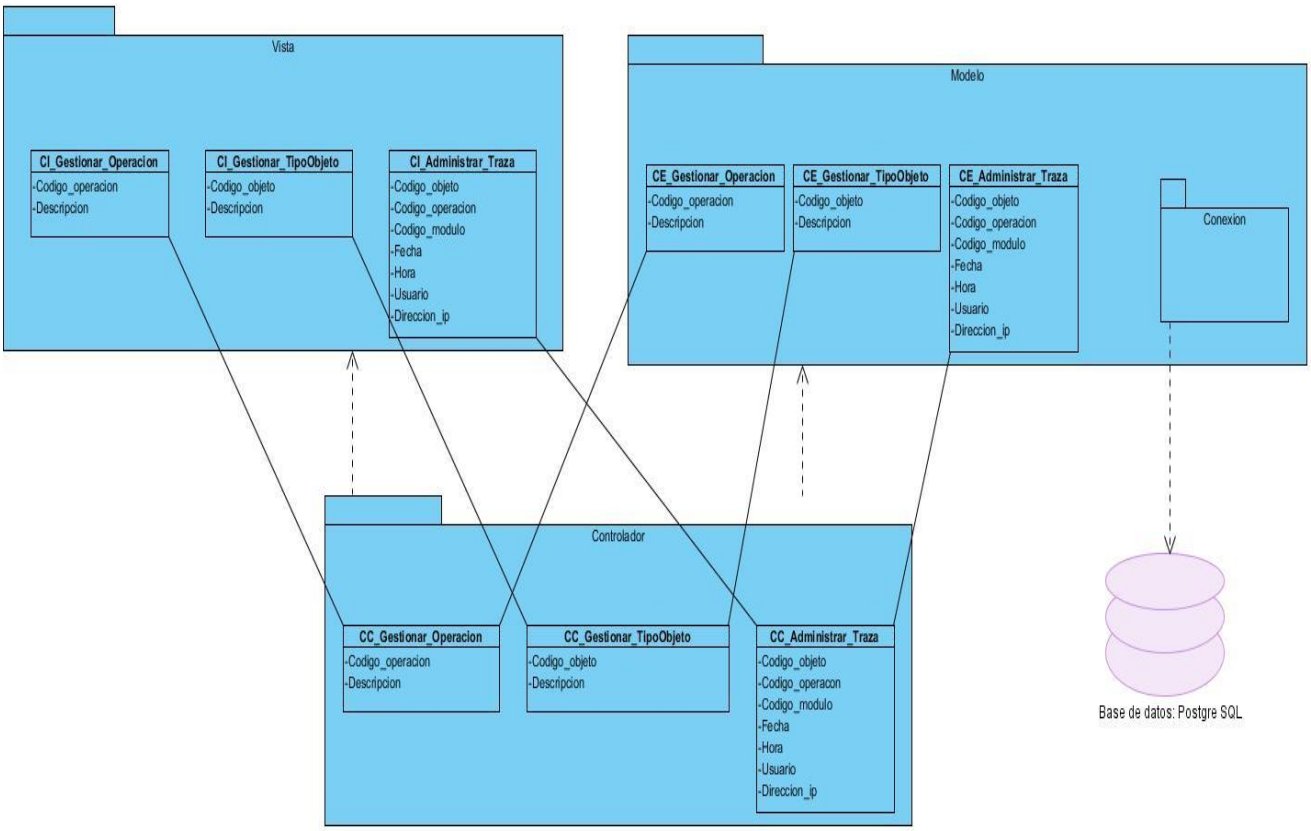


Ilustración 1 Diseño de la arquitectura del módulo de gestión de trazas de SISEN

2.2.3 Patrones de diseño

Según (Fernandez Rabilero, y otros, 2016) un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. El mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Estos modelos que se presentan como parejas de problema/solución con un nombre, codifican buenos principios y sugerencias relacionados con la asignación de responsabilidades, basados en la recopilación del conocimiento de los expertos en desarrollo de software. Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. De manera más simple, un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (Torres Garces, y otros, 2015). Es decir, Los patrones de diseño son unas técnicas para resolver problemas comunes en el desarrollo de *software* en el diseño de interacción o interfaces.

Patrones Grasp

Controlador

- El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Este patrón se ve reflejado en la siguiente clase:

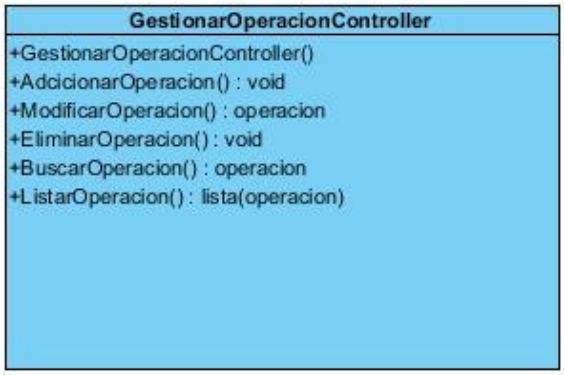


Ilustración 2 Clase GestionarOperaciónController

Creador

- Se encarga de guiar la asignación de responsabilidades relacionadas con la creación de objetos. Su intención es encontrar un creador que necesite conectarse al objeto creado en alguna situación.

Este patrón se ve reflejado en la siguiente clase:

Ilustración 3 Clase Operación

Bajo Acoplamiento

Asignar una responsabilidad de manera que la dependencia entre elementos permanezca baja. Este patrón se ve reflejado en la clase Tipo Objeto, esta clase solo tiene dos relaciones.

Alta Cohesión

Asignar responsabilidades de manera que una clase no tenga muchas funcionalidades no relacionadas o no realice un trabajo excesivo. Este patrón incrementa la claridad y facilita la comprensión del diseño. Este patrón se puede ver reflejado en la clase operación de la figura 3, esta tiene muy pocos métodos.

Patrones de interfaz de usuario

Llenar los espacios en blanco

- Permite introducir datos alfanuméricos en un cuadro de texto (Somerville, 2005).

Este patrón se utiliza en el siguiente código:

```
items: [
   {
      xtype: 'textfield',
      fieldLabel: 'Código objeto',
      allowBlank: false,
      name: 'codObjeto'

   },
   {
      xtype: 'textfield',
      fieldLabel:'Descripción',
      allowBlank: false,
      name: 'descripcionObjeto
```

Ilustración 4 Código para introducir datos

Editar (EditInPlace)

- Brinda capacidades de edición de texto sencillo para ciertos tipos de contenido en la ubicación que se muestra en la pantalla. No es necesario que el usuario introduzca explícitamente alguna función de edición de texto o algún modo (Somerville, 2005).

Este patrón se utiliza en el siguiente código:

```
EDITAR
@Action(id = "_actualizar", name = "Actualizar", value = "/{id}", method = {RequestMethod.PUT, RequestMethod.OPTIONS})
public ResponseEntity<ServerInfo> actualizar(@PathVariable("id") Integer id, @Valid @RequestBody Operacion object, BindingResult
result) throws Exception {
   if (result.hasErrors())
      throw new FormValidationError(result.getAllErrors());
   operacionService.actualizar(object, id);
   return new ResponseEntity(new ServerInfo(true, messageSource.getMessage("base.actualizacion.satisfactoria")), HttpStatus.OK);
}
```

Búsqueda Simple (SimpleSearch)

- Brinda la capacidad de hacer una búsqueda local (una página o un archivo) o global (todo el sitio o la base de datos completa) para la cadena de búsqueda. Genera una lista de “aciertos” ordenados según su probabilidad de satisfacer las necesidades del usuario (Somerville, 2005). Este patrón se puede ver en el prototipo de interfaz del requisito listar operación donde se realiza una búsqueda de operaciones.

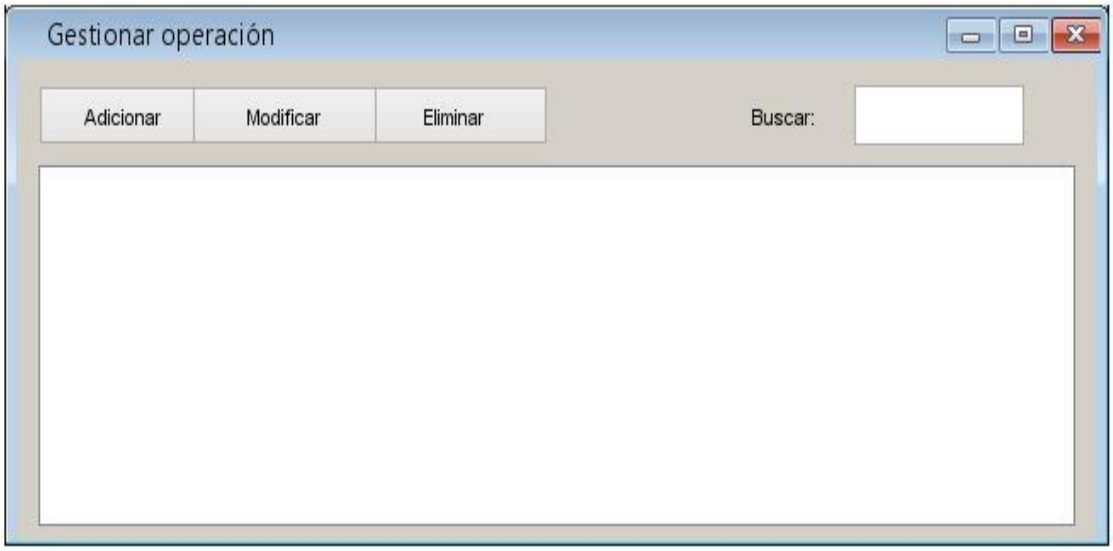


Ilustración 5 Historia de usuario listar operación.

2.2.4 Modelo de datos

El modelo de datos es un conjunto de conceptos que sirven para describir la estructura, la semántica y las relaciones que existen entre las entidades de una base de datos, mostrando los datos que serán contenidos en el sistema (Sánchez, 2016). Según (Torres Peregrino, y otros, 2013) el modelo de datos es de los artefactos generados en la fase de análisis y diseño de acuerdo con el modelo de desarrollo del centro. Tiene el propósito de garantizar que los datos persistentes sean almacenados coherente, eficazmente y definir el comportamiento que debe ser implementado en la base de datos. El modelo de datos define todos los objetos de datos que se procesan dentro del sistema, la relación entre ellos y otro

tipo de información que sea pertinente para las relaciones. El diagrama entidad-relación (DER) aborda dichos aspectos y representa todos los datos que se introducen, almacenan, transforman y generan dentro de la aplicación.

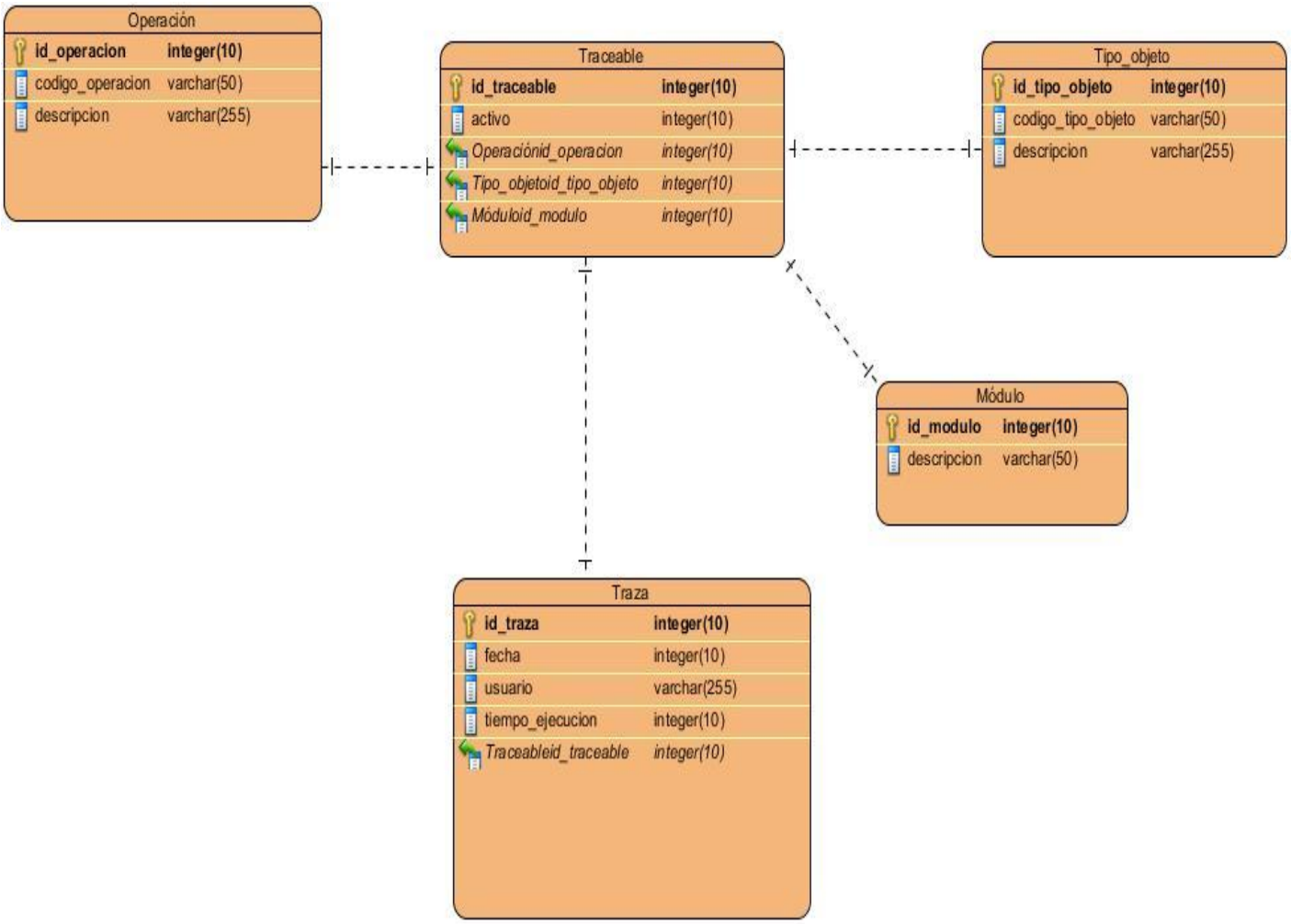


Ilustración 6 Modelo de datos.

Tabla 4 Descripción de la tabla Traza.

Nombre: Traza		
Descripción: Almacena las trazas de todos los usuarios registrados en el sistema.		
Atributo	Tipo	Descripción
id	intiger(10)	Identificador auto-incremental.
usuario	varchar(255)	Nombre del usuario registrado en el sistema.
fecha	date	Fecha en la que se produce la traza
direccion_ip	intiger(10)	Dirección ip donde se realizó la operación.
tiempo_ejecucion	intiger(10)	Tiempo que dure la operación.

Id_traceable	intiger(10)	Identificador auto-incremental.(Llave foránea que representa la relación la tabla Traceable)

Tabla 5 Descripción de la tabla Traceable.

Nombre: Traceable		
Descripción: Decide que traza va ser auditada o no según su módulo, operación u objeto		
Atributo	Tipo	Descripción
id	intiger(10)	Identificador auto-incremental.
activo	boolean	Traza auditada o no.
id_operacion	intiger(10)	Identificador auto-incremental.(Llave foránea que representa la relación la tabla Operación)
id_objeto	intiger(10)	Identificador auto-incremental.(Llave foránea que representa la relación la tabla Tipo objeto)
id_modulo	intiger(10)	Identificador auto-incremental.(Llave foránea que representa la relación la tabla Módulo)

Tabla 6 Descripción de la tabla Operación.

Nombre: Operación		
Descripción: Gestiona las operaciones.		
Atributo	Tipo	Descripción
id	intiger(10)	Identificador auto-incremental.
codigo_operacion	varchar(255)	Código de la operación
Descripción	varchar(255)	Describe la operación que se realizó.

Tabla 7 Descripción de la tabla Objeto.

Nombre: Tipo_objeto		
Descripción: Gestiona los objetos		
Atributo	Tipo	Descripción
id	intiger(10)	Identificador auto-incremental.
codigo_objeto	varchar(255)	Código del objeto.
Descripción	varchar(255)	Describe el nombre del objeto.

Tabla 8 Descripción de la tabla Módulo.

Nombre: Módulo		
Descripción: Gestiona los módulos		
Atributo	Tipo	Descripción
id	intiger(10)	Identificador auto-incremental.
Descripción	varchar(255)	Describe el nombre del modulo

2.3 Métricas de software orientadas a clases para evaluar el diseño

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas, de manera que se puedan desarrollar los remedios y mejorar el proceso del software (Arteaga, 2014). La evaluación de un producto, mediante métricas, es un aspecto fundamental a tener en cuenta, aunque las métricas del producto del software no suelen ser absolutas, brindan la posibilidad de evaluar la calidad a partir de varias reglas definidas claramente. Las métricas son también utilizadas para señalar áreas con problemas, de manera que se puedan desarrollar los remedios y mejorar el proceso del software (González, 2016). Es decir, las métricas permiten evaluar de alguna manera la calidad de los procesos del software y de los productos teniendo en cuenta varios análisis y comparaciones de promedios obtenidos en el proceso.

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

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

2.3.1 Tamaño operacional de clase (TOC)

Consiste en medir el tamaño general de una clase tomando el valor de la cantidad de operaciones que están encapsuladas dentro de dicha clase (León Mendoza, y otros, 2015).

Tabla 9 Descripción de métrica TOC

Atributo	Modo en que lo afecta	Categoría	Criterio
Responsabilidad	Un aumento en el TOC implica un aumento de la responsabilidad asignada a la clase.	Baja	CP <= Promedio
		Media	Promedio<=CP<=2*Promedio
		Alta	CP > 2 * Promedio
Complejidad de implementación	Un aumento en el TOC implica un aumento de la complejidad de implementación de la clase.	Baja	CP<= Promedio
		Media	Promedio<= CP<=2*Promedio
		Alta	CP > 2 * Promedio
Reutilización	Un aumento en el TOC implica una disminución del grado de reutilización de la clase.	Baja	CP>2*Promedio
		Media	Promedio<= CP<=2 * Promedio
		Alta	CP <= Promedio

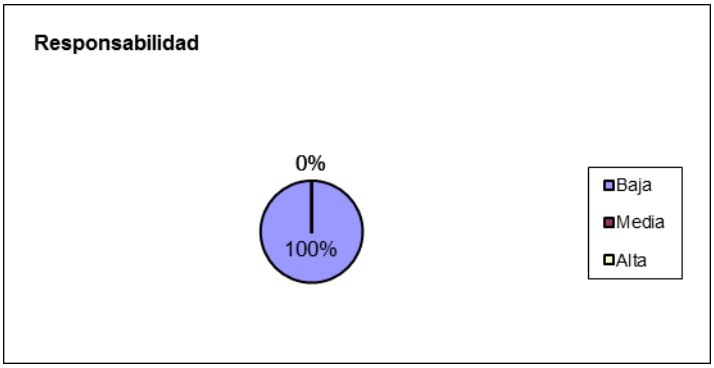


Ilustración 7 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad

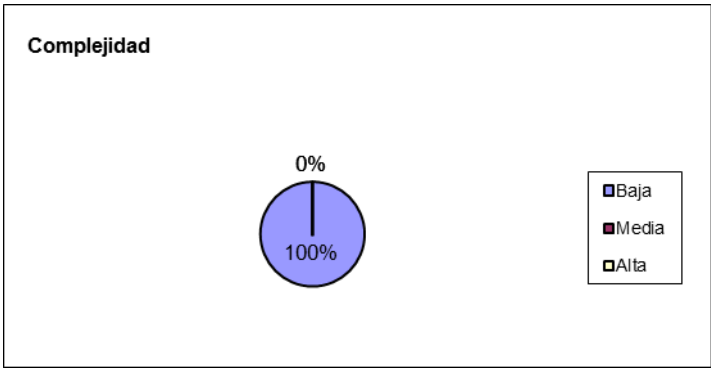


Ilustración 8 Resultados de la evaluación de la métrica TOC para el atributo Complejidad

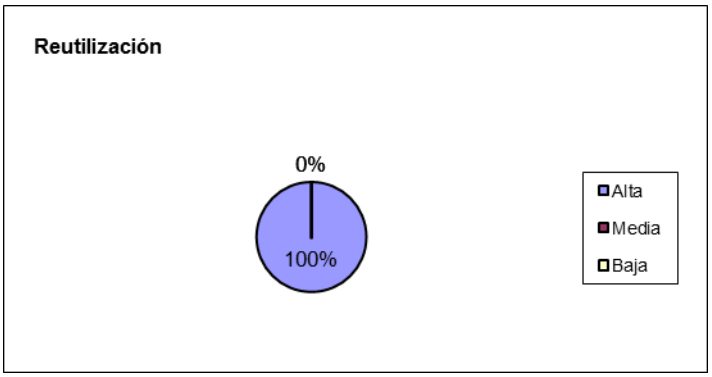


Ilustración 9 Resultados de la evaluación de la métrica TOC para el atributo Reutilización

Al hacer un análisis de los resultados obtenidos se concluye que todas las clases incluidas poseen de 1 a 5 procedimientos en su composición representando el 100% de la muestra. Se puede observar que la mayoría de las clases que conforman el sistema para los atributos responsabilidad y complejidad están dentro de la categoría Baja para un 100% respectivamente mientras que el atributo Reutilización cuenta con similares porcientos en las categoría Alta mostrando así que el componente cuenta con un buen grado de reutilización, baja complejidad y responsabilidad en el diseño propuesto. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

2.3.2 Relaciones entre Clases (RC)

La métrica RC está dada por el número de relaciones de uso de una clase con otra. Permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase, teniendo en cuenta las relaciones existentes entre ellas (León Mendoza, y otros, 2015).

Tabla 10 Descripción de métrica RC

Atributo	Modo en que lo afecta	Categoría	Criterio
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.	Ninguno	0
		Bajo	1
		Medio	2
		Alto	> 2
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.	Bajo	Cantidad de relaciones de uso <= Promedio
		Medio	Promedio <= Cantidad de relaciones de uso <= 2* Promedio
		Alto	Cantidad de relaciones de uso > 2*Promedio.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesaria para probar una clase.	Bajo	Cantidad de relaciones de uso <= Promedio
		Medio	Promedio <= Cantidad de relaciones de uso <= 2* Promedio
		Alto	Cantidad de relaciones de uso > 2* Promedio
Reutilización	Un aumento en el RC implica una disminución en el grado de reutilización de la clase.	Baja	Cantidad de relaciones de uso >2* Prom
		Media	Promedio <= Cantidad de relaciones de uso <= 2* Promedio
		Alta	Cantidad de relaciones de uso <= Promedio

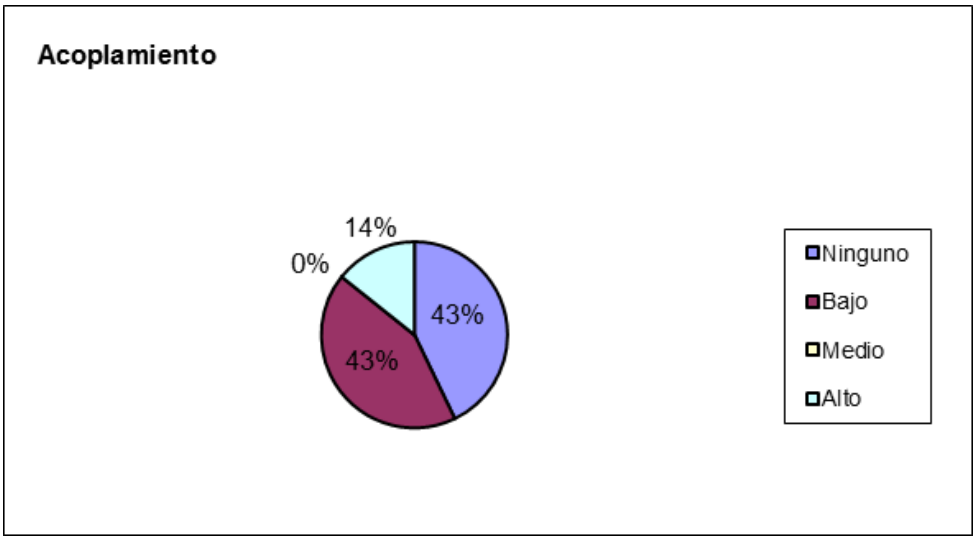


Ilustración 10 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento

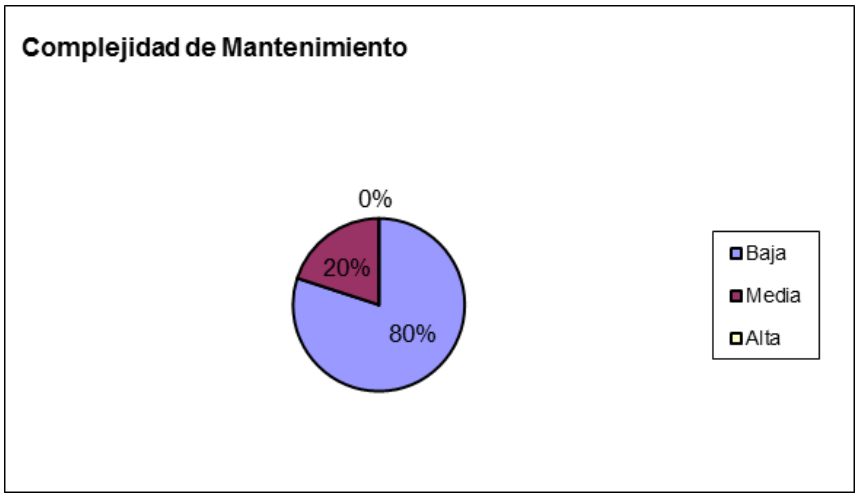


Ilustración 11 Resultados de la evaluación de la métrica RC para el atributo Complejidad de mantenimiento

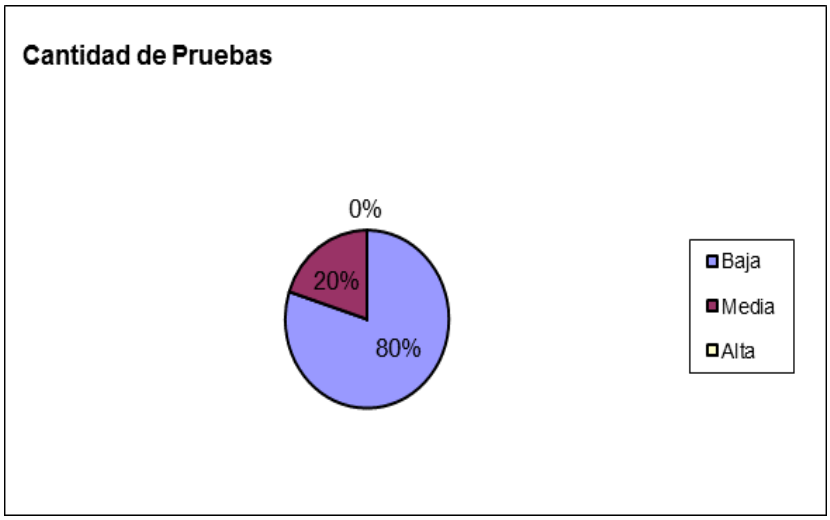


Ilustración 12 Resultados de la evaluación de la métrica RC para el atributo Cantidad de pruebas

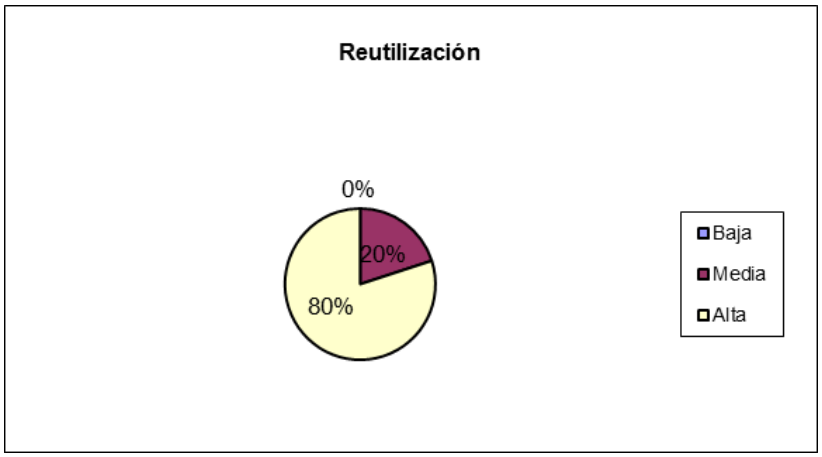


Ilustración 13 Resultados de la evaluación de la métrica RC para el atributo Reutilización

Al realizar un análisis de los resultados obtenidos a través de la métrica de software RC aplicada se obtiene que el 43% de las clases incluidas poseen bajo acoplamiento lo cual ayuda significativamente en el sentido de que al ser afectada una clase este cambio no repercutirá en las demás. El 80% de reutilización de clases es alto lo cual fomenta y potencia el uso de las mismas en otras clases, en caso de ser necesario, además la complejidad de mantenimiento es de 80% aproximadamente baja, facilitando que no resulte difícil el mantenimiento de las mismas (por ejemplo la optimización de métodos) y por último el atributo relacionado con la cantidad de pruebas es 80% bajo lo cual señala que a las clases se les puede realizar un bajo número de pruebas de poca complejidad y de bajos costes.

2.4 Conclusiones parciales del capítulo

Después de haber identificado y validado los requisitos, además de haber realizado y validado un análisis y diseño del sistema se llegó a la conclusión:

- La captura y validación de los requisitos son necesarias para que el desarrollador pueda brindarle al cliente el producto esperado.
- Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y permiten responder rápidamente a los requisitos cambiante.
- Con la utilización de la arquitectura Modelo-Vista-Controlador, el uso de los patrones de diseño y el modelo de datos propuestos, se garantiza una mayor organización, reutilización de funciones y código más legible.

- Las métricas Tamaño Operacional de la Clase y Relaciones entre Clases brindan la posibilidad de evaluar la calidad y señalar áreas con problemas, de manera que se puedan desarrollar los remedios y mejorar el proceso del software.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Este capítulo tiene como objetivo evaluar y validar la calidad del sistema aplicando un conjunto de técnicas como las que se explicarán seguidamente. Se definirán los estándares de codificación para la implementación de la solución. También se aplicarán pruebas de caja negra y caja blanca. Y se validará la investigación mediante el método Delphy.

3.1 Estándares de codificación

Los estándares de codificación de aplicaciones deben comprender todos los aspectos de la programación. Este debe tender a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código. Por tanto, debe definirse al comenzar un proyecto el estándar de codificación, asegurando así que los programadores trabajen de forma coordinada. Con el propósito de lograr un estándar de codificación para la homogenización del código.

3.1.1 Estándares de codificación para las interfaces

Indentación

- Usar cuatro espacios como unidad de indentación.
- Cuando una expresión no entre en una sola línea, se debe romper de acuerdo a estos principios generales:

Romper después de una coma.

Romper antes de un operador.

Alinear la nueva línea al principio de la expresión al mismo nivel de la línea anterior.

Si las reglas anteriores conducen a la confusión del código o el código se alínea contra el margen derecho, se indenta con menos espacios.

Declaraciones

- Se debe declarar cada variable en su propia línea, si utiliza la declaración de varias variables separadas por coma, se debe romper después de la coma.
- Es válido inicializar variables al momento de su declaración.

Sentencias

- Se deben usar sentencias simples, cada sentencia debe ser escrita en su propia línea.
- No utilizar el operador coma para agrupar declaraciones múltiples.

Internacionalización

- Se deben seguir las reglas definidas para los mensajes en el apartado “Con respecto a los mensajes” del propio documento.
- Se deben declarar todos los identificadores para la internacionalización en la carpeta correspondiente al idioma, no dentro de los archivos de código.

3.1.2 Estándares de codificación para Java

Longitud de línea

- Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

Rompiendo líneas

- Cuando una expresión no entre en una línea, romperla de acuerdo con estos principios:
- Romper después de una coma.
- Romper antes de un operador.
- Preferir roturas de alto nivel (más a la derecha que el "padre") que de bajo nivel (más a la izquierda que el "padre").
- Alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea anterior.
- Si las reglas anteriores llevan a código confuso o a código que se aglutina en el margen derecho, indentar justo 8 espacios en su lugar.

Declaraciones

- Se recomienda una declaración por línea, ya que facilita los comentarios**.**
- Intentar inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.
- Al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:
- Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.
- La llave de cierre "}" empieza una nueva línea indentada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{".

Sentencias simples

- Cada línea debe contener como mucho una sentencia.

3.2 Diagrama de componentes

Un diagrama de componentes es la representación del sistema en componentes físicos y las dependencias entre ellos. Un componente es una parte física de un sistema (módulo, base de datos), que es una unidad autónoma que implementa una o más clases que representan un contrato de servicios que el componente ofrece (Pressman, 2010).

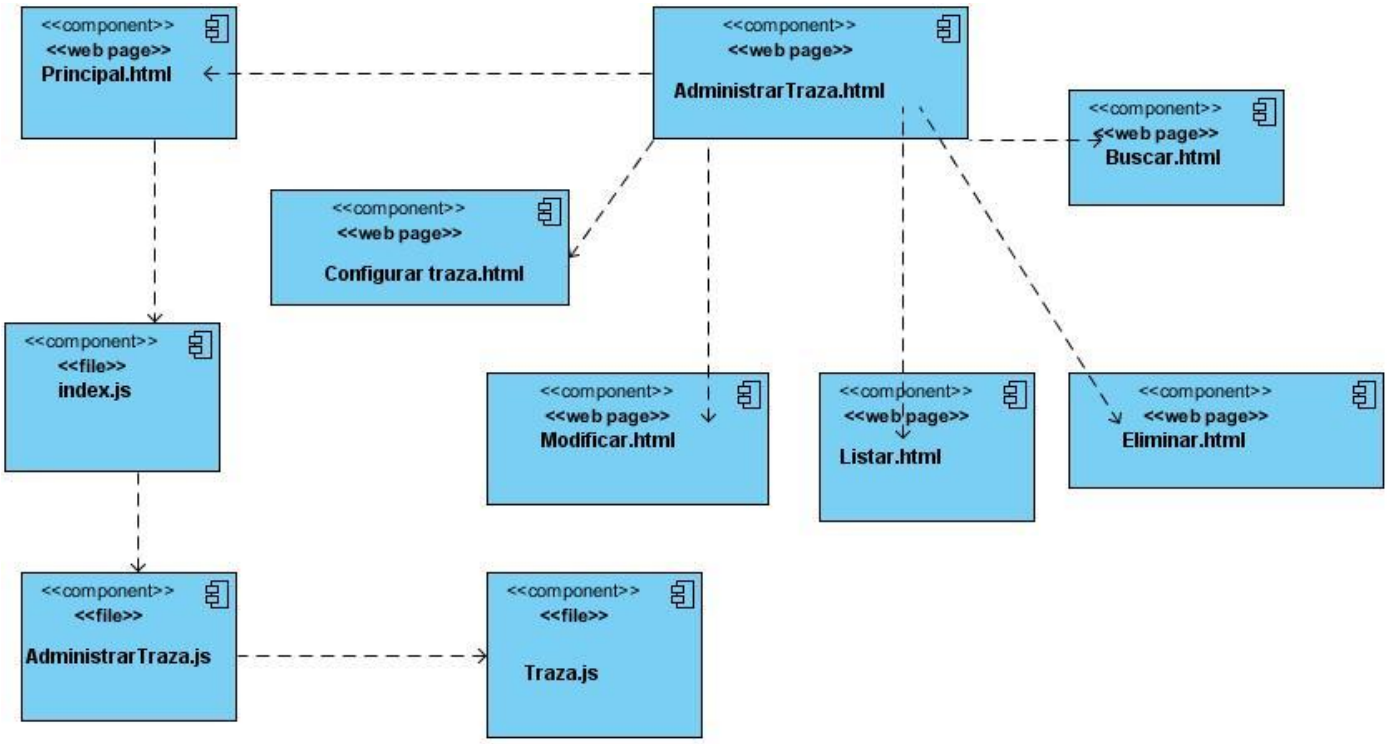


Ilustración 14 Diagrama de componente de administrar traza

3.2 Pruebas del software

3.2.1 Prueba de caja blanca

En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y examinado el estado del programa en varios puntos. Aunque consecuentemente esto llevaría un estudio exhaustivo, para este sistema solo se realizó una valoración de los caminos lógicos importantes (Morejón, 2015). Según (Fernandez Rabilero, y otros, 2016) las pruebas de unidad se realizan mediante el método de caja

blanca o estructural, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. La técnica de prueba de caja blanca utilizada en la investigación es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

A continuación, se muestra el código del método registrar operación al cual se le aplicó la métrica de complejidad ciclomática debido a que es uno de los métodos relevantes en la solución informática brindada:

```
public void registrarOperacion(AdminTraza adminTraza, String metodo) {
if (metodo.equals("registrar")) {
adminTraza.setIdOperacion(new Operacion(3620));
}
if (metodo.equals("actualizar")) {
adminTraza.setIdOperacion(new Operacion(3626));
}
if (metodo.equals("eliminar")) {
adminTraza.setIdOperacion(new Operacion(3632));
}
if (metodo.equals("obtener")) {
adminTraza.setIdOperacion(new Operacion(4012));
}
if (metodo.equals("paginar")) {
adminTraza.setIdOperacion(new Operacion(3638));
}
if (metodo.equals("login")) {
adminTraza.setIdOperacion(new Operacion(4029));
}
}
```

Ilustración 15 Código del método registrar operación.

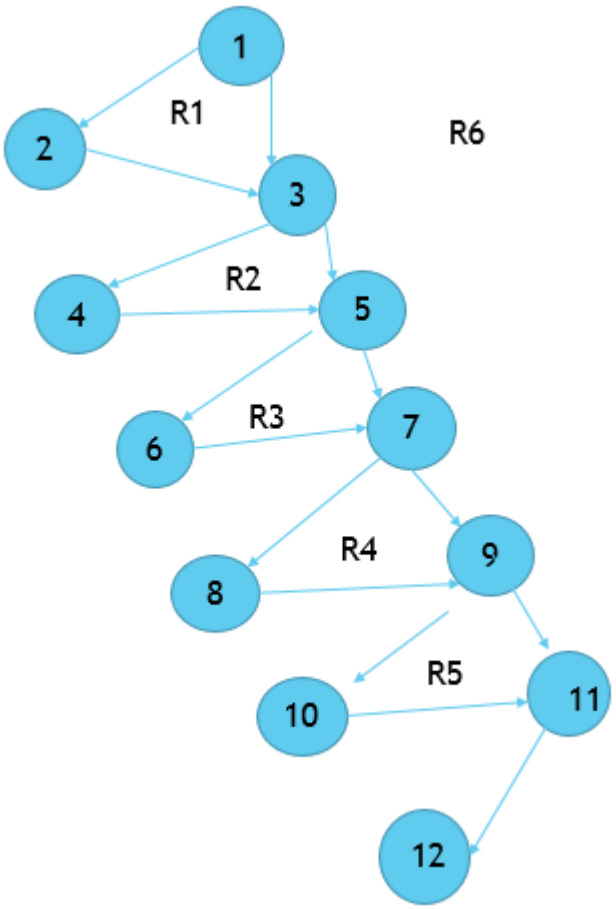


Ilustración 16 Código del método registrar operación.

A = aristas

N= nodos

R= región

P= nodos predicados (1,3,5,7,9)

Cálculo de complejidad ciclomática:

V(G)= R= 6

V(G)= A- N+ 2= 16-12+2=6

V(G)= P+ 1= 6

Caminos:

Camino 1: 1- 2- 3- 5- 7- 9- 11

Camino 2: 1- 3- 4- 5- 7- 9- 11

Camino 3: 1- 3- 5- 6- 7- 9- 11

Camino 4: 1- 3- 5- 7- 8- 9- 11

Camino 5: 1- 3- 5- 7- 9- 10- 11

Camino 6: 1- 3- 5- 7- 9-10- 11- 12

Resultados de caja blanca

Para la validación del código generado en el desarrollo del módulo se seleccionaron los métodos más relevantes. A estos métodos se le realizaron pruebas para poder evaluar si el funcionamiento de cada uno de ellos se comportó de la manera esperada. Los resultados fueron satisfactorios en una primera iteración de pruebas. Con la aplicación de las pruebas de caja blanca se obtuvo un 100% de pruebas con resultados satisfactorios, comprobándose la estabilidad de la lógica aplicada en el código.

3.2.2 Prueba de caja negra

El objetivo de las pruebas de caja negra es verificar desde la interfaz de usuario que dado una determinada entrada del sistema responda como está dispuesto en su concepción (Santana, 2015). Según (Troche, 2015) son las que realizan pruebas de forma que se compruebe que cada función del sistema es operativa. Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa funcione bien. En el documento se describirán los casos de pruebas para los requisitos funcionales de mayor prioridad.

Tabla 11 Casos de pruebas

Escenario	**Descripción**	**Variables**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Adicionar Operación	Permite adicionar las operaciones al sistema	Operación	Muestra la operación adicionada en la lista de operaciones	1- El usuario elige la opción de gestionar operación. 2- Luego selecciona Adicionar. 3- Introduce los valores y oprime el botón Aceptar.
EC 1.2	Permite	Operación	Muestra la	1- El usuario elige la opción

Modificar Operación	modificar las operaciones al sistema		operación adicionada en la lista de operaciones	de gestionar operación. 2- Luego selecciona Modificar. 3- Introduce los valores y oprime el botón Aceptar.
EC 1.3 Eliminar Operación	Permite eliminar las operaciones al sistema	Operación	Elimina la operación y muestra la lista de operaciones	1- El usuario elige la opción de gestionar operación. 2- Selecciona una operación y luego selecciona Eliminar. 3- Oprime el botón Aceptar.
EC 1.4 Listar Operación	Permite ver el listado de las operaciones registradas en el sistema	Operación	Muestra la lista de operaciones registradas en el sistema	1- El usuario elige la opción de gestionar operación.
EC 1.5 Buscar Operación	Permite buscar según un nombre las operaciones	Operación	Muestra la lista de operaciones registradas que coinciden con el nombre introducido en el sistema	1- El usuario elige la opción de gestionar operación. 2- Introducir el nombre de la operación y oprimir buscar.
EC 2.1 Adicionar Tipo de Objeto	Permite adicionar los tipos de objetos al sistema	Tipo de Objeto	Muestra el tipo de objeto adicionado en la lista de tipo de objetos	1- El Usuario elige la opción de gestionar tipo de objeto. 2- Luego selecciona Adicionar. Introduce los valores y oprime el botón Aceptar.

EC 2.2 Modificar Tipo de Objeto	Permite modificar los tipos de objetos al sistema	Tipo de Objeto	Muestra el tipo de objeto adicionada en la lista de tipo de objeto	1- El Usuario elige la opción de gestionar tipo de objeto. 2- Luego selecciona Modificar. Introduce los valores y oprime el botón Aceptar.
EC 2.3 Eliminar Tipo de Objeto	Permite eliminar los tipos de objetos al sistema	Tipo de Objeto	Elimina la operación y muestra la lista de operaciones	1- El usuario elige la opción de gestionar tipo de objeto. 2- Selecciona un tipo de objeto y luego selecciona Eliminar. Oprime el botón Aceptar.
EC 2.4 Listar Tipo de Objeto	Permite ver el listado de los tipos de objetos registradas en el sistema	Tipo de Objeto	Muestra la lista de tipo de objeto registradas en el sistema	1- El usuario elige la opción de gestionar tipo de objeto.
EC 2.5 Buscar Tipo de Objeto	Permite buscar según un nombre los tipos de objetos	Tipo de Objeto	Muestra la lista de tipo de objeto registradas que coinciden con el nombre introducido en el sistema	1- El Usuario elige la opción de gestionar tipo de objeto. 2- Introducir el nombre del tipo de objeto y oprimir buscar.
3.1 EC Listar traza	Permite ver el listado de las trazas	Traza	Muestra la lista de trazas registradas en	Al abrir el módulo se muestra una lista de las trazas

	registradas en el sistema		el sistema	
3.2 EC Buscar traza	Permite buscar según un parámetro las trazas	Traza	Muestra la lista de trazas registradas que coinciden con el parámetro introducido en el sistema	Introducir el parámetro por el cual va a buscar la traza y oprimir buscar
3.3 EC Configurar traza	Mostrar las opciones para la configuración	Traceable	Muestra el panel donde el usuario podrá realizar la configuración	1- Seleccionar la opción de configurar traza. 2- Hacer la configuración y oprimir el botón aceptar.

Resultados de caja negra

Las pruebas de caja negra efectuadas al sistema fueron realizadas por un especialista de calidad de CEIGE y arrojaron un total de 3 no conformidades en la primera iteración. En la segunda iteración el sistema fue liberado. A continuación se muestra una tabla donde se refleja la cantidad de no conformidades detectadas por iteración.

Iteraciones	**No conformidades**
1	3
2	-

3.3 Validación de la investigación

En la validación de la investigación se utiliza el criterio de expertos método Delphy. Define (Astigarraga, 2003) la técnica Delphi como un método de estructuración de un proceso de comunicación grupal que es efectivo a la hora de permitir a un grupo de individuos, como un todo, tratar un problema complejo. Una Delphi consiste en la selección de un grupo de expertos a los que se les pregunta su opinión sobre cuestiones referidas a acontecimientos del futuro. Las estimaciones de los expertos se realizan en sucesivas rondas, anónimas, al objeto de tratar de conseguir consenso, pero con la máxima autonomía

por parte de los participantes. La capacidad de predicción de la Delphi se basa en la utilización sistemática de un juicio intuitivo emitido por un grupo de expertos. El método Delphi procede por medio de la interrogación a expertos con la ayuda de cuestionarios sucesivos, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos. La encuesta se lleva a cabo de una manera anónima para evitar los efectos de "líderes". Según (García Valdés, y otros, 2013) el Delphi es una metodología estructurada para recolectar sistemáticamente juicios de expertos sobre un problema, procesar la información y a través de recursos estadísticos, construir un acuerdo general de grupo. Permite la transformación durante la investigación de las apreciaciones individuales de los expertos en un juicio colectivo superior. Es un método de estructuración de un proceso de comunicación grupal que es efectivo a la hora de permitir a un grupo de individuos, como un todo, tratar un problema.

Resultado de método Delphy

Para la aplicación práctica del método es necesario considerar metodológicamente dos cuestiones fundamentales, la elaboración del cuestionarios y la selección del grupo de expertos a encuestar. Para la aplicación práctica del método es necesario considerar metodológicamente dos cuestiones fundamentales, la elaboración del cuestionario y la selección del grupo de expertos a encuestar. El cuestionario realizado se encuentra en los anexos del trabajo. Mientras que los encuestados son especialistas y desarrolladores del proyecto SISEN. Para el procesamiento de la información obtenida en las encuestas se hizo corresponder un valor numérico a cada posible resultado de la escala utilizada. Los valores utilizados se muestran en la siguiente tabla:

Tabla 12 Valores numéricos

Respuesta	Valor numérico
Mucho	5
Bastante	4
Poco	3
Muy poco	2
Nada	1

El procesamiento de las respuestas mediante la utilización de esta escala sugiere empleo de una tabla de doble entrada de la forma siguiente:

Tabla 13 Resultado de la aplicación de método Delphy

Expectos	Preguntas						
	1	2	3	4	5	6	7
1	5	4	5	4	4	4	4
2	5	4	5	4	4	4	4
3	4	4	5	4	4	5	4
4	5	5	5	5	4	4	3
5	4	5	5	5	5	5	4
6	4	4	4	5	5	5	3
7	4	4	5	4	4	4	4
8	5	4	4	5	5	5	4
9	5	5	5	4	4	4	3
10	5	5	5	4	4	4	3
Cj media	4,60	4,4	4,8	4,4	4,3	4,4	3,6
Varianza	0,27	0,27	0,18	0,27	0,23	0,27	0,27
Coef. De Variación	0,11	0,12	0,09	0,12	0,11	0,12	0,14

A partir de la tabla anterior se obtienen los siguientes estimados:

Media aritmética por pregunta:

$$\overline{C_j} = \frac{\sum_{i=1}^{m_j} C_{ij}}{m_j}$$

Constituye la media aritmética de los expertos que evalúan la pregunta j.

Grado de concordancia de los expertos para una pregunta dada:

$$\sigma_j^2 = \frac{\sum_{i=1}^{m_j} (C_{ij} - \overline{C_j})^2}{m_j - 1}$$

A partir de la fórmula anterior para determinar la varianza, se determina el coeficiente de variación (Vj), este coeficiente es una medida del grado de concordancia de los expertos por cada pregunta, donde mientras mayor sea el valor de Vj menor será el grado de concordancia de los expertos.

$$Vj = \frac{\sqrt{\sigma_j^2}}{Cj}$$

De los resultados obtenidos se puede afirmar que existe un alto grado de concordancia en el criterio de los expertos en todas las preguntas realizadas pues el coeficiente de variación mas alto es de 0.12. De este modo se puede concluir que, de acuerdo al consenso en el juicio de los expertos, la solución propuesta contribuye con el monitoreo y control de SISEN.

El análisis de la tabla anterior permitió concluir lo siguiente por cada uno de las preguntas:

Tabla 14 Evaluación de las preguntas del cuestionario

Preguntas	Valoración de los expertos
¿Es necesario un módulo de gestión de trazas para SISEN?	Mucho
¿La gestión de trazas contribuye al monitoreo y control de SISEN?	Mucho
¿Es importante la gestión de trazas?	Mucho
¿Es importante la realización de auditorías?	Bastante
¿Es necesario llevar un control de las actividades de los usuarios del sistema?	Bastante
¿La gestión de trazas mejora el rendimiento del sistema?	Bastante
¿La gestión de trazas permite recuperar cambios sobre los datos?	Bastante

3.4 Conclusiones parciales

Mediante la realización de este capítulo se pudo llegar a las siguientes conclusiones:

1. Las pruebas de caja blanca y caja negra realizadas a la solución demostraron el correcto funcionamiento del módulo.
2. Mediante el método Delphy se pudieron obtener criterios de especialistas que validaron que la solución era factible para el problema.

CONCLUSIONES

1. La gestión de trazas contribuye a la seguridad del sistema y mejora la gestión de información posibilitando el monitoreo y control de sistemas informáticos.
2. Las tecnologías y herramientas utilizadas para el desarrollo del módulo están encaminadas hacia el código abierto y multiplataforma.
3. A través del análisis y diseño se definió la solución a implementar para un mejor entendimiento de la misma, teniendo en cuenta las necesidades del cliente.
4. La validación de la solución mediante la utilización de las métricas TOC y RC permitió comprobar la calidad del diseño propuesto, obteniéndose resultados positivos.
5. Las ejecuciones de las pruebas para la validación del sistema propuesto permitieron detectar las no conformidades del mismo y la corrección de estas.
6. La validación de la investigación mediante la utilización del método Delphy, permitió concluir la necesidad de desarrollar un módulo de gestión de trazas, que permita el monitoreo y control de las operaciones que se ejecutan sobre el SISEN.

RECOMENDACIONES

Para contribuir a la continuidad de la investigación, se hace la siguiente recomendación:

1. A medida de que se incorporen nuevos campos en los registros del historial de trazas del SISEN se recomienda agregar la funcionalidad ver traza, que posibilite la opción de mostrar los nuevos registros almacenados en la base de datos que son útiles para que el usuario realice un mejor monitoreo y control de las acciones del sistema.
2. Seguir actualizando el módulo en consecuencia a la evolución del SISEN en cuanto a tecnología.

BIBLIOGRAFÍA

Finogeev, Alexey , Parygin , Danila y Finogeev, Anton . 2017. The convergence computing model for big sensor data mining and knowledge discovery. *The convergence computing model for big sensor data mining and knowledge discovery.* [En línea] 2017. [Citado el: 30 de 4 de 2018.] https://hcis-journal.springeropen.com/articles/10.1186/s13673-017-0092-7.

Marín Sánchez, Jacqueline y Alejandro Lugo García, José. 2016. *Control de proyectos de software: actualidad y retos para la industria cubana.* 2016.

Torres Peregrino, Carlos Alberto y Peguero Alvarez, Héctor David. 2013. *Módulo para el registro y transformación de trazas de eventos a formato XES.* La Habana : s.n., 2013.

Arteaga, René Olazabal. 2014. *Módulo de gestión de trazas para el marco de trabajo Synfony 2.* La Habana : s.n., 2014.

2018. AWStats. *AWStats official web site.* [En línea] 2018. [Citado el: 1 de 2 de 2018.] https://awstats.sourceforge.io/.

Cañete Pupo, Y. 2010. *Libro de ayuda del marco de trabajo Sauxe, en su verión 2.0.* Ciudad de la Habana : s.n., 2010.

Corrales Martinez, Y. 2009. *Componentes para la configuración de la gestión del multilenguaje y la gestión de trazas del sistemas Cedrux.* Ciudad de la Habana : s.n., 2009.

De la Peña Gutierrez, Alberto. 2007. Auditoría un enfoque práctico. [En línea] 2007. [Citado el: 7 de 12 de 2017.] https://books.google.com.cu/books?id=337WBN_QaBEC&printsec=frontcover&dq=auditoria&hl=es&sa=X&ved=0ahUKEwjd2dHDoIHYAhUBMt8KHZeGDXMQ6AEIJTAA#v=onepage&q=auditoria&f=false.

Delgado Delgado, Manuel Emilio. 2013. *Componente de software para la administración y el análisis centralizado de las trazas generadas en la solución alas PACS-RIS.* La Habana : s.n., 2013.

ESEN. [En línea] [Citado el: 8 de 12 de 2017.] www.esen.cu/.

Fernandez Rabilero, Ramón y Varona Carmenates, Arian. 2016. *Sistema para la gestión de transportaciones nacionales de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2016.

Ferrer Oduardo, Miguel. 2013. *Sistema de control de medios materiales del CEIGE.* La Habana : s.n., 2013.

Fonseca Vega, Martha Rocío y Izaguirre Delgado, Fernando. 2015. *Desarrollo de una herramienta para la evaluación de la mantenibilidad mediante métricas en el Laboratorio de Pruebas del CEIGE.* La Habana : s.n., 2015.

Fuentes Torrubia, Jose Alberto. 2011. [En línea] 2011. [Citado el: 10 de 12 de 2017.] https://ddd.uab.cat/pub/trerecpro/2012/hdl_2072_195716/FuentesTorrubiaJoseAlbertoR-ETISa2010-11.pdf.

García Peñalvo, Francisco José. 2018. *INGENIERÍA DE SOFTWARE I.* 2018.

González González, Leonardo. 2014. *Sistema genérico de gestión de trazas para los productos.* La Habana : s.n., 2014.

—. 2014. *Sistema genérico de gestión de trazas para los productos desarrollados en el departamento de Integración de Soluciones.* 2014.

González, Leyriel Zurita. 2016. *Componente para la extracción de registros de eventos en formato XES del Sistema de Planificación de Actividades.* La Habana : s.n., 2016.

Groussar, Thierry. 2012. [En línea] 2012. [Citado el: 10 de 12 de 2017.] https://books.google.es/books?hl=es&lr=&id=JaPTzKZxbN4C&oi=fnd&pg=PA9&dq=Java+JDK+7+&ots=pV0Eocwo0b&sig=WiZYnDv_pvJaCYdttpSu5SlIa-c#v=onepage&q=Java%20JDK%207&f=false.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2011. *El Procceso de investigación científca.* 2011.

Investigacíon Arqueometría y Analisis Arquelogíco. [En línea] [Citado el: 7 de 12 de 2017.] http://pendientedemigracion.ucm.es/info/arqueoanalisis/Service%201.2.html.

Johnson, Rod. 2013. Spring Framework Reference Documentation. *Spring Framework Reference Documentation.* [En línea] 2013. [Citado el: 30 de 4 de 2018.] https://docs.spring.io/autorepo/docs/spring-framework/3.2.17.RELEASE/spring-framework-reference/pdf/spring-framework-reference.pdf.

León Mendoza, Anabel Fé y López Rami, Ebrik René . 2015. *Sistema de Gestión Integral de la Calidad de Software para el Centro de Gobierno Electrónico.* La Habana : s.n., 2015.

Martin Frías, Henrry y Usatorres Ramirez , Yanmichel. 2014. *Gestión de trazas en el Sistema para el control Farmacológico del Centro de Informática Médica.* La Habana : s.n., 2014.

Miranda, Idalmis Fonseca. 2015. *Sistema de Gestión del Patrimonio Informático para la Dirección Municipal de Gastronomía de Boyeros.* La Habana : s.n., 2015.

Morales, Y, Marrero, M y Oliva, A. Extensión de la herramienta Visual Paradigm para la generación de clases de acceso a datos con Doctrine 2.0. Serie Científica de la UCI. [En línea]

Morejón, Dayana Donatien. 2015. *Sistema para la Gestión de la información y Control de Libros v 2.0 para la librería de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2015.

Mozilla. 2014. Mozilla. [En línea] 2014. [Citado el: 11 de 12 de 2017.] http://www.mozilla.org/es-ES/firefox/features/.

Oliva Perez, Alianskis y Soler Orellana, Claudia. 2013. *Desarrollo del componente Evaluación del Desempeño del Subsistema Capital Humano de Cedrux.* La Habana : s.n., 2013.

Porven Rubier, Joelsy. 2015. *MARCO DE TRABAJO PARA LA GESTIÓN CENTRALIZADADE TRAZAS DE SEGURIDADUTILIZANDO HERRAMIENTAS DECÓDIGO ABIERTO.* La Habana : s.n., 2015.

PostgreSQL-es. [En línea] [Citado el: 10 de 12 de 2017.] http://www.postgresql.org.es/sobre_postgresql.

Pressman, Roger. 2010. *Ingeniería de software. Un enfoque práctico.* 2010.

Ribiaux, Aliuska Powell. 2013. *Desarrollo del componente Trabajo con cuadros del Sistema Integral de Gestión de Cedrux.* La Habana : s.n., 2013.

Rodríguez Sánchez, Tamara. 2014. Metodología de desarrollo para la actividad productiva de la UCI. 2014.

Romeu, Ivaniet Díaz. 2015. *Sistema Infomático de gestión de indicadores para la formación de estudiantes potencialmente talentosos en la UCI.* La Habana : s.n., 2015.

Sacerio Martinez, A. 2010. *Análisis, diseño e implentación del Framework de registro de eventos y su herramienta de seguimiento.* Ciudad de la Habana : s.n., 2010.

Sánchez, Daniel Herrera. 2016. *Interfaz gráfica para la configuración de los componentes Caché, Excepciones, IUX, Aspectos y Trazas del marco de trabajo Bosón .* La Habana : s.n., 2016.

Sánchez, Yoiry López. 2017. *SISTEMA WEB PARA LA GESTIÓN DEL CONTROL DE ALMACÉN EN LA MINI-INDUSTRIA EL MAMBÍ DEL MUNICIPIO DE FLORENCIA EN LA PROVINCIA DE CIEGO DE ÁVILA.* Ciego de Avila : s.n., 2017.

Santana, Mario Manuel Melo. 2015. *Portal Web de la Oficina de Asuntos Históricos del Consejo de Estado.* La Habana : s.n., 2015.

Sawmill. 2017. Sawmill, Universal Log File Analysis and Reporting. [En línea] 2017. [Citado el: 10 de 12 de 2017.] http://www.sawmill.net.

2014. Sencha. [En línea] 2014. [Citado el: 10 de 12 de 2017.] http://www.sencha.com/products/extjs/.

Sencha. 2018. Sencha. [En línea] 2018. [Citado el: 26 de 4 de 2018.] http://docs.sencha.com/cmd/guides/intro_to_cmd.html.

Sistema para el control de una traza de un servidor proxy . **Crespo González, Yankier y Benedico Aguilera, Yoel. 2017.** 2, 2017, Vol. 6.

Somerville, Ian. 2005. *Ingeniería del software. Séptima edición.* 2005.

Suaza, Katerine Villamizar. 2015. *Mejora de historias de usuario y casos de prueba de metodologías ágiles con base en TDD.* 2015.

Torres Garces, Yaicel y Sosa Vázquez, Yunier José. 2015. *Sistema para la Sincronización de calendarios.* La Habana : s.n., 2015.

Troche, Maiker Rodríguez. 2015. *Sistema para el diagnóstico y seguimiento de riesgo en el centro DATEC.* La Habana : s.n., 2015.

Vukotic, Aleksa y Goodwill, James. *Apache Tomcat 7.*

Webalizer. 2014. Webalizer. *Webalizer.* [En línea] 2014. [Citado el: 1 de 2 de 2018.] http://www.webalizer.org/.

Webspy. 2017. Webspy. *Webspy.* [En línea] 2017. [Citado el: 8 de 12 de 2017.] http://www.webspy.com/.

ANEXOS

1 GLOSARIO

[1] Conjunto de empresas dedicadas a la gestión y desarrollo de la actividad aseguradora, re-aseguradora, financiera, de asistencia, inspección y ajuste de averías y de servicios conexos en general.

[2] Organismo encargado de dirigir, ejecutar y controlar la aplicación de la Política Financiera, Tributaria, de Precios, de Auditoria y de Seguros, del Estado y del Gobierno, asesorándolos en esta política, dirigir y controlar la organización de la Administración Financiera del Estado.

[3] Es un contrato, denominado *póliza de seguro*, por el que una Compañía de Seguros (el asegurador) se obliga, mediante el cobro de una prima y para el caso de que se produzca el evento cuyo riesgo es objeto de cobertura a indemnizar, dentro de los límites pactados, el daño producido al *asegurado*; bien a través de un capital, una renta, o a través de la prestación de un servicio.

[4] Es el método por el cual una aseguradora cede parte de los riesgos que asume con el fin de reducir el monto de su pérdida posible.

[5] Un debugger (en español, depurador), es un programa usado para probar y depurar (eliminar) los errores de otros programas (el programa objetivo.

[6] Zend Framework es un framework de código abierto para aplicaciones web orientado a objetos para PHP 5. Zend Framework a menudo se denomina 'biblioteca de componentes' porque tiene muchos componentes poco compactos que puede usar de manera más o menos independiente. Pero Zend Framework también proporciona una implementación avanzada Modelo-Vista-Controlador (MVC) que se puede utilizar para establecer una estructura básica para sus aplicaciones de Zend Framework. Puede encontrar una lista completa de los componentes de Zend Framework junto con descripciones breves en la descripción general de los componentes.

[7] Doctrine es un mapeador relacional de objetos (ORM) para PHP 5.2.3 que se encuentra sobre una poderosa capa de abstracción de base de datos (DBAL). Una de sus características principales es la opción de escribir consultas de bases de datos en un dialecto SQL orientado a objetos patentado llamado Doctrine Query Language (DQL), inspirado en Hibernates HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad sin requerir la duplicación de código innecesaria.

[8] ExtJS es una librería de componentes JavaScript. Esta librería es muy rica en componentes visuales. Algunas cosas que más me gustan de ExtJS: Tiene soporte nativo para realizar llamadas Ajax. El

componente tabla es uno de los mejores (Antes yo usaba displaytag). Su formulario tiene la opción de información al servidor sin hacer submit de toda la pagina.

[9] Un cortafuego (firewall) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar o descifrar el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios.

[10] El antispam es lo que se conoce como método para prevenir el correo basura. Tanto los usuarios finales como los administradores de sistemas de correo electrónico utilizan diversas técnicas contra ello. Algunas de estas técnicas han sido incorporadas en productos, servicios y software para aliviar la carga que cae sobre usuarios y administradores. No existe la fórmula perfecta para solucionar el problema del spam por lo que entre las múltiples existentes unas funcionan mejor que otras, rechazando así, en algunos casos, el correo deseado para eliminar completamente el spam, con los costes que conlleva de tiempo y esfuerzo.

[11] Un **log** es un registro de actividad de un sistema, que generalmente se guarda en un fichero de texto, al que se le van añadiendo líneas a medida que se realizan acciones sobre el sistema. Se utiliza en muchos casos distintos, para guardar información sobre la actividad de sistemas variados.

[12] Interfaz de entrada común (en inglés *Common Gateway Interface*, abreviado **CGI**) es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME. Las aplicaciones que se ejecutan en el servidor reciben el nombre de **CGIs**.

[13] Derecho exclusivo de un autor, editor o concesionario para explotar una obra literaria, científica o artística durante cierto tiempo.

[14]Constituye una metodología estándar para el análisis, implementación y documentación de sistemas orientados a objetos.

[15] El **lenguaje unificado de modelado** (**UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

[16] ***Business Process Model and Notation*** (**BPMN**), en español **Modelo y Notación de Procesos de Negocio**, es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (*workflow*).

[17] **XML**, siglas en inglés de *eXtensible Markup Language*, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible.

[18] Las Enterprise JavaBeans (también conocidas por sus siglas EJB) son una de las interfaces de programación de aplicaciones (API) que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE) de Oracle Corporation (inicialmente desarrollado por Sun Microsystems).

[19] Las librerías JavaScript son archivos con instrucciones para agregarle diversas funcionalidades y efectos a las páginas de internet, usando este lenguaje de programación. Todos los navegadores modernos incluyen de forma nativa JavaScript, lo que les permite interpretar funciones básicas, estas librerías agregan otros recursos para manipular la estructura de las páginas (DOM) de forma dinámica y el estilo visual (CSS), como demanda la web moderna.

[20] El **código abierto** es un modelo de desarrollo de *software* basado en la colaboración abierta. Se enfoca más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.

[21] La expresión **cliente servidor** se utiliza en el ámbito de la informática. En dicho contexto, se llama cliente al dispositivo que requiere ciertos servicios a un servidor. La idea de servidor, por su parte, alude al

equipo que brinda servicios a las computadoras (ordenadores) que se hallan conectadas con él mediante una red.

[22] ***Berkeley Software Distribution*** o **BSD** (en español, «distribución de software Berkeley») fue un sistema operativo derivado de Unix que nace a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley.

[23] **Unidad de procesamiento gráfico** o ***GPU*** (*Graphics Processing Unit*) es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.