# *Universidad de las Ciencias Informáticas*
## *FACULTAD 6*


# *Título: Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones*.

***Trabajo de Diploma para optar por el título de***

***Ingeniero en Ciencias Informáticas***

***Autor:*** *Leonardo González González*

***Tutores****: Ing. Lianet Cabrera González*

*Ing. Virgilio Suárez Bello*

*Ing. Yoander Iñiguez Bermúdez*

*La Habana, Junio de 2014*
*"Año 56 de la Revolución"*

*"Hay que tener fe en uno mismo. Ahí reside el secreto. Aun cuando estaba en el orfanato y recorría las calles buscando qué comer para vivir, incluso entonces, me consideraba el actor más grande del mundo. Sin la absoluta confianza en sí mismo, uno está destinado al fracaso."*

*Charles Chaplin*

## DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis y autorizo a 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 _____.

Leonardo González González

_____

Firma del Autor

Ing. Yoander Iñiguez Bermúdez

_____

Firma del Tutor

Ing. Virgilio Suárez Bello

_____

Firma del Tutor

Ing. Lianet Cabrera González

_____

Firma del Tutor

## Datos de Contacto

**Tutor:** Lianet Cabrera González.
**Especialidad de graduación:** Ingeniero en Ciencias Informáticas.
**Años de experiencia en el tema:** 1año.
**Años de graduado:** 1.
**Correo electrónico:** lcabrerag@uci.cu.

**Tutor:** Yoander Iñiguez Bermúdez.
**Especialidad de graduación:** Ingeniero en Ciencias Informáticas.
**Años de experiencia en el tema:** 3 años.
**Años de graduado:** 3.
**Correo electrónico:** yiniguez@uci.cu.

**Tutor:** Virgilio Suárez Bello.
**Especialidad de graduación:** Ingeniero en Ciencias Informáticas.
**Años de experiencia en el tema:** 1 año.
**Años de graduado:** 1.
**Correo electrónico:** vsuarez@uci.cu.

AGRADECIMIENTOS

Quisiera agradecer a todas las personas que me ayudaron, en especial a mi familia por estar hay para mí. A mis profesores y tutores por guiarme en estos cinco años. A mis amigos y a toda aquella gente que siempre ha tenido fe en mí.

A mi madre, la persona más importante de mi vida.

## Resumen

El departamento Integración de Soluciones perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC) en la Universidad de las Ciencias Informáticas (UCI), desarrolla software relacionados con las base de datos, almacenamiento, obtención y representación de la información. Para esta tarea cuenta con diferentes proyectos como Solución Integral de Gestión de Datos (SIGDAT), el cual realiza diseños de plantillas entre otras funciones. En su versión actual, la aplicación presenta deficiencias en cuanto a la gestión de trazas, al igual que SIGDAT estas irregularidades son encontradas en otros productos del departamento. El presente trabajo de diploma tiene como objetivo desarrollar un sistema que gestione trazas de forma genérica. Para guiar el proceso de desarrollo del sistema 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 y de rendimiento para verificar el correcto funcionamiento del sistema genérico para la gestión de trazas.

PALABRAS CLAVES: genérico, servicios web, trazas.

## Abstract

The Solutions Integration Technologies of the Center for Data Management (DATEC) at the University of Informatics Sciences (UCI), department develops related database, storage, retrieval and presentation of information software. This task has several projects such as Integrated Data Management Solution (SIGDAT), which performs template designs among other functions. In its current version, the application has deficiencies in managing traces, like SIGDAT these irregularities are found in other projects of the department. This diploma work is to develop a system to manage traces generically. To guide the process of system development methodology, tools and technologies used features. For the structure, design and implementation of the classes was used the Model View Controller architectural pattern and other design patterns. Functional and performance tests were made to verify correct operation of the generic system for trace management.

KEYWORDS: generic, traces, web services.

Índice de Contenidos

# Índice de Figuras

# Índice de Tablas

# INTRODUCCIÓN

Durante las tres últimas décadas se ha ido incrementando una segunda revolución tecnológica a causa de la integración de los ordenadores y los sistemas de información en la estrategia empresarial, siendo las Tecnologías de la Información y las Comunicaciones (TICs) el campo fundamental para su desarrollo. Actualmente, en las grandes compañías del mundo, el uso de la computación y programas informáticos es fundamental: sea ésta productiva, comercial o de servicios necesitan de dichas tecnologías para su crecimiento. Para tener un alto nivel de competitividad y posibilidades de desarrollo, la gestión de la información en una empresa es vital. La seguridad de la propia puede significar la diferencia entre el éxito o el fracaso para todos los proyectos que se emprendan dentro de una entidad empresarial.

La seguridad informática es una disciplina que tiene como objetivo garantizar la confidencialidad, integridad, disponibilidad y trazabilidad de los recursos gestionados por los sistemas informáticos en las organizaciones, apoyándose de procesos y funciones de software y hardware ya establecidos. Uno de sus procesos es la auditoría informática, que ha de velar por la correcta utilización de los recursos que la empresa gestiona para disponer de un eficiente y eficaz sistema de información. Con frecuencia, el auditor informático debe verificar que los sistemas realizan exactamente las funciones previstas y no otras. Para ello se apoya en productos de software muy potentes y modulares que, entre otras funciones, rastrean los caminos que siguen los datos a través del programa. Especialmente, las trazas se utilizan para comprobar la ejecución de las validaciones de datos previstos; en ellas se encuentran informaciones relevantes como quién, cuándo y/o qué modificaciones incurrieron sobre los recursos informáticos. También son usadas para registrar todas las funciones de un sistema, así como las alertas o mensajes de errores generados por una aplicación y poder detectar indicios de peticiones no autorizadas que pueden afectar la seguridad o la estabilidad del sistema. Las trazas son acumuladas generando grandes volúmenes de información, esto trae consigo problemas relacionados con la capacidad de almacenamiento. Una herramienta de registro oficial de eventos es útil para el control total de la información.

La Universidad de las Ciencias Informáticas (UCI) juega un papel esencial en la informatización industrial. Está conformada por centros de desarrollo de software con objetivos diferentes, que tienen como misión ofrecer al país soluciones informáticas que ayuden al desarrollo de las empresas cubanas y de la sociedad. El Centro de Tecnologías de Gestión de Datos (DATEC) perteneciente a la UCI, desarrolla

sistemas relacionados con las bases de datos, almacenamiento, análisis y representación de la información. Entre sus objetivos se encuentra satisfacer las necesidades de gestión de la información, y para esta tarea se apoya en tres departamentos: Almacenes de Datos, Postgres e Integración de Soluciones. Productos como Sistema Integral de Gestión de Datos (SIGDAT) y Generador Dinámico de Reportes (GDR) entre otros, desarrollados por este último departamento, son destinados como ayuda hacia una mayor precisión y rapidez para la toma de decisiones. Entre sus funciones se brinda un uso efectivo y eficiente de la información en función de los objetivos y metas de la empresa en cuanto a desempeño y calidad.

Actualmente los datos que se gestionan en SIGDAT, poseen un ciclo de vida a través del cual sufren modificaciones o pueden ser eliminados por diversos usuarios u otras causas. Sin embargo, no se registran las operaciones ejecutadas por los usuarios ni las alertas o errores emitidos por la aplicación. No contar con una herramienta capaz de registrar las operaciones realizadas en los sistemas, puede traer consigo problemas como desinformación ante posibles errores o alertas ocurridos en la aplicación; en cuanto a los usuarios involucrados en estos productos no se posee un control individual de cada persona. Esto imposibilita la realización de auditorías o conocer el estado del sistema, por lo que no se puede identificar con posterioridad cómo recuperar o revertir algún cambio realizado sobre los datos o configuraciones y/o quién fue el responsable de algún cambio o problema en específico en un momento dado. En conclusión, no se posee un control sobre la gestión de los datos, lo que puede conllevar a una mala gestión de la información. Al igual que SIGDAT estas irregularidades son encontradas en otros proyectos del departamento, en el caso de GDR, sí se gestionan las trazas, pero se presentan dificultades dado que para visualizar las propias se requiere de generar un reporte de la tabla que contiene dichas trazas.

Entre las herramientas más utilizadas por el departamento se encuentran el marco de trabajo *Symfony* y el servidor web *Apache*; en parte de sus funciones estas realizan un registro de *log* donde se archiva una amplia información referente a los errores ocurridos durante su ejecución. Este registro almacena gran volumen de información con compleja estructura y formato, lo cual hace casi imposible y engorroso el entendimiento para un humano. Existen múltiples herramientas para la visualización y/o análisis de las trazas almacenadas bajo estos estándares, como Webalizer, logstalgia y eventView entre otras. Donde generar resultados gráficos y exportar reportes estadísticos son funciones comunes en estos programas.

Aún presentando un alto nivel de efectividad, estas herramientas no son viables como solución para las dificultades existentes, ya que presentan la característica de ser aplicaciones de escritorio y requieren de ser instaladas directamente en el servidor contenedor de los archivos logs. Actualmente el departamento Integración de Soluciones requiere de una aplicación web que permita de forma genérica la gestión de trazas de todos los productos informáticos y evite los problemas antes mencionados sin tener que implementar una solución particular para cada uno de ellos.

Por lo anteriormente planteado se identificó como **problema de la investigación:** ¿Cómo gestionar las trazas de los productos desarrollados en el departamento Integración de Soluciones?

Para dar solución al problema planteado se identificó como **objeto de estudio**: El proceso de gestión de trazas informáticas, el cual se enmarca en el **campo de acción**: La gestión de trazas en los productos de software desarrollados en el departamento Integración de Soluciones.

Siendo el **objetivo general** de la investigación: Desarrollar un Sistema genérico de gestión de trazas que permita registrar las operaciones ejecutadas por los usuarios y los eventos ocurridos en los productos desarrollados por el departamento Integración de Soluciones.

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

1. Estudio de los fundamentos teóricos relacionados con la gestión de trazas informáticas existentes en el mundo.
2. Realizar el análisis y diseño del Sistema genérico de gestión de trazas.
3. Realizar la implementación y pruebas del Sistema genérico de gestión de trazas.

Para darle cumplimiento a todos los objetivos trazados, se han definido las siguientes **tareas de la investigación**:

1. Análisis de las aplicaciones para la gestión de trazas a nivel internacional y nacional, estableciendo similitudes con la investigación en curso.
2. Análisis de las formas de representación de la información obtenida para contribuir a la asimilación de la misma por el usuario final.
3. Definición de los requisitos funcionales y no funcionales para describir cómo se debe implementar el Sistema genérico de gestión de trazas.

4. Identificación de la arquitectura utilizada en los productos del departamento Integración de Soluciones para la integración del servicio web de trazas.
5. Elaboración del modelo de diseño para refinar el análisis del Sistema genérico de gestión de trazas que será desarrollado.
6. Implementación del Sistema genérico de gestión de trazas.
7. Integración de las funciones brindadas por el Sistema genérico de gestión de trazas a la arquitectura de los productos del departamento Integración de Soluciones.
8. Realización de pruebas de funcionalidades y rendimiento que validen el correcto funcionamiento del Sistema genérico de gestión de trazas.

## Diseño Metodológico

En el departamento Integración de Soluciones se detectan problemas ocasionados por la ausencia de un sistema de gestión de trazas en algunos de sus productos, por lo que se hace una investigación con un estudio de tipo descriptiva. Su principal objetivo es describir el fenómeno y reflejar lo esencial y más significativo del mismo, sin tener en cuenta las causas que lo originan, para lo que es necesario captar sus relaciones internas y regularidades, así como aquellos aspectos donde se revela lo general.

Para la obtención de información se utiliza el **método empírico** entrevista, en el cual se establece una conversación planificada entre el investigador y el entrevistado, donde es de suma importancia una buena comunicación. Mediante este método se concluye los siguientes datos relevantes: Actualmente en el producto SIGDAT no existe una forma de conocer las acciones llevadas a cabo por los usuarios sobre los datos que se gestionan, al igual que las alertas emitidas por operaciones no autorizadas. En el caso de GDR, sí se gestionan las trazas, pero se presentan dificultades dado que para visualizar las propias se requiere de generar un reporte de la tabla que contiene dichas trazas. Esto presenta como principal desventaja que no se obtiene una información detallada acerca de qué acción ocurrió, cuándo y a través de qué usuario del sistema. Además, se obtendría un reporte considerablemente extenso.

Se hará uso además de **métodos teóricos**, como el análisis histórico, por lo que se realiza un estudio en cuanto a tendencias de sistemas que gestionan trazas informáticas tanto a nivel nacional como internacional. También se utiliza el método lógico de la modelación mediante diagramas y diseños de clases entre otros artefactos, para obtener una abstracción previa con el objetivo de explicar la realidad.

Para establecer un orden de trabajo y guiar el ciclo de vida de la solución, se plantean las siguientes **preguntas científicas**:

1 ¿Qué fundamentos teóricos se encuentran asociados con el estudio de la gestión de trazas informáticas?

2 ¿Para el desarrollo e implementación de un sistema genérico de gestión de trazas qué metodología, herramientas y tecnologías utilizar?

3 ¿Qué características debe poseer el Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones que erradique los problemas antes mencionados?

4 ¿Dada la problemática, qué diseño es el más conveniente a utilizar en la implementación?

5 ¿Con la creación del Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones, qué beneficios se pueden obtener?

El presente documento consta de cuatro partes fundamentales: resumen, introducción, desarrollo y conclusiones. A continuación se explica brevemente el contenido de cada capítulo:

**Capítulo 1: Elementos teóricos para el desarrollo de sistemas de trazas en la informática:** En este capítulo se analizan aspectos teóricos, principales conceptos y definiciones asociadas a trazas informáticas. Se realiza una valoración crítica de los diferentes sistemas de software existentes para la gestión de trazas. Además, se describen las principales herramientas, tecnologías, arquitectura y metodología que serán usadas.

**Capítulo 2: Análisis y diseño del Sistema genérico de gestión de trazas:** En este capítulo se realiza una descripción de las principales características del sistema, detallando la propuesta de solución. Se realiza una representación visual de clases conceptuales del entorno real de los objetos del sistema a través del modelo de dominio. Se ilustran los principales requisitos funcionales y no funcionales, así como los artefactos generados teniendo en cuenta la metodología seleccionada.

**Capítulo 3**: **Implementación y prueba del Sistema genérico de gestión de trazas:** El presente capítulo contiene los elementos fundamentales referentes a la etapa de implementación según la metodología de desarrollo de software utilizada. Se hace referencia a los principales estándares de codificación para la nomenclatura de las clases, así como los artefactos generados durante esta fase. Se realiza además una validación de la solución mediante pruebas funcionales y de integración. Se explican cada una de las métricas utilizadas y se muestran los resultados de la implantación del sistema.

# CAPÍTULO 1: "ELEMENTOS TEÓRICOS PARA EL DESARROLLO DE SISTEMAS DE TRAZAS EN LA INFORMÁTICA"

## Introducción

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

## 1.1 Trazabilidad

La Organización Internacional para la Estandarización (ISO) la define en su *International Vocabulary of Basic and General Terms in Metrology* 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".

En pocas palabras, 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.

### *1.1.1 Importancia de 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 (2):

- 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 (2).

### *1.1.2 Análisis de trazas*

El análisis de trazas es el acto de separar las colecciones de eventos y datos que representan el rastro dejado por cierto proceso, para estudiar su naturaleza, su función y/o su significado y llegar a resultados más específicos. Es la capacidad de extraer y concentrar de un gran volumen de información antes gestionada, toda evidencia relacionada con un hecho (3).

### *1.1.3 Sistemas de Gestión*

Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización. Ayuda a lograr los objetivos de la empresa mediante una

serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado (4). El trabajo en equipo y un funcionamiento organizado dentro de una empresa permite aprovechar y desarrollar el potencial existente en la institución, logra mejoras continuas, reduce los costos y mejora la efectividad operativa.

Un sistema de **gestión de trazas** es aquel que registra y permite auditar los eventos que ocurren dentro de un sistema determinado, lo cual es muy importante para su correcto funcionamiento. Un producto informático que no posea un componente con esta funcionalidad implica grandes riesgos. Al no existir dicho componente, muchos aspectos se verán afectados, a continuación se muestran tres de ellos:

- El control de las funciones informáticas de la empresa y/o institución.
- Cumplimiento de las normativas generales de la empresa y/o institución.
- Análisis de los controles y procedimientos tanto organizativos como operativos (2).

### *1.1.4 Herramientas existentes para el monitoreo de trazas*

Las trazas permiten a los auditores informáticos evaluar en alguna medida la confianza que se puede depositar en un sistema basado en la evidencia que brindan las mismas y contribuyen a avalar el sistema en cuanto a seguridad y fiabilidad de la información que se maneja. De ahí que las aplicaciones implementen un sistema de control de la trazabilidad que responda a sus necesidades y contribuya a garantizar la seguridad para alcanzar un mejor nivel, donde se satisface las necesidades del entorno que exige calidad y confianza. A continuación se presenta un estudio de algunos sistemas informáticos implementados en Cuba y en otros países que gestionan trazas, donde se identificaron en su solución algunas de las funcionalidades propuestas.

- ***TraceListener***

Básicamente *TraceListener* es una herramienta de software propietario para el registro de eventos a partir de las trazas generadas por el sistema operativo. Esta herramienta no contempla las acciones individuales de cada usuario y tampoco permite la interoperabilidad o capacidad de integrarse a otra aplicación. Tiene incorporado el registro de trazas a través de los siguientes lenguajes:

- NET / C # / VB / J #.
- Java.
- C + +.

- Perl.
- PHP.
- PLSQL / Oracle Databaselogging.

Entre sus principales características se destacan (15):

- Capaz de escuchar en los puertos TCP o UDP para el registro de eventos entrantes de aplicaciones.
- Cargar / Guardar y recibir registros de eventos simultáneamente desde múltiples fuentes.
- Vista por detalles de cada registro de eventos.
- Capacidad de búsqueda completa.
- Interfaz de usuario en inglés y alemán.
- Extensión de las posibilidades de filtro (por ejemplo, por fecha).
- Gestor de Base de Datos Oracle.

- ***SRNI2***

El Sistema de Reportes de la Navegación por Internet (SRNI) desarrollado en la UCI, proporciona reportes dinámicos de la navegación de los usuarios a través de internet. Crea reportes a partir de las trazas del servidor proxy y los usuarios pueden consultar la información de accesos por las siguientes formas:

- Acceso por dominios.
- Acceso por URL.
- Acceso por dirección IP.
- Acceso por días.
- Acceso por horas.

Además, el sistema tiene la característica de mostrar el consumo equivalente al tamaño de los recursos que fueron accedidos mediante la WWW *(World Wide Web)*. También ofrece la posibilidad de conocer el nuevo sistema de consumo, que se calcula en correspondencia con los sitios a los que se accede y la hora en la que se realizó este acceso (6).

Luego de estudiar algunos sistemas que gestionan trazas, se puede concluir que estos integran en su solución la información proveniente de las acciones realizadas por los usuarios en forma de logs. También se evidencia que ninguno de los sistemas estudiados sirven como solución, ya que no se ajustan a lo

requerido en cuanto a la información a registrar. Por lo que se reafirma la realización de un Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones.

## 1.2 Servicio Web

Un Servicio Web (en inglés, *Web Service*) es un sistema de software diseñado para soportar la interoperabilidad máquina - máquina a través de una red. Este tiene una interfaz descrita en un formato que puede ser procesado por una máquina, específicamente un WSDL *(Web Services Description Language)*. Otros sistemas interactúan utilizando mensajes SOAP (*Simple Object Access Protocol*), los cuales se encuentran establecidos previamente. Sirven para intercambiar datos entre distintas aplicaciones de software desarrolladas en lenguajes de programaciones diferentes y ejecutadas sobre cualquier plataforma (7).

Para operar con las distintas tecnologías empleadas en los productos desarrollados en el departamento Integración de Soluciones, se hizo uso de los servicios web, ya que poseen características que los diferencian de otros métodos como: DCOM (*Distributed Component Object Model*), CORBA(*Common Object Request Broker Architecture*) o DCE(*Distributed Computing Environment*) entre otros.

Entre sus principales características están:

- Acoplamiento débil: El cliente solamente necesita conocer su definición WSDL, la implementación del servicio le es transparente.
- Independencia del lenguaje de programación: El cliente puede hacer uso del servicio sin importar el lenguaje en que este haya sido escrito.
- Independencia del modo de transporte: SOAP funciona sobre protocolos tan disímiles como: HTTP, HTTPS, HTTP-R, BEEP, SMTP o FTP.
- Extensibilidad: Al estar basado en XML, los servicios web son fáciles de adaptar, extender y personalizar (7).

Entre las principales ventajas del uso de los servicios web se destacan:

- Aumenta la interoperabilidad entre programas independientemente de la plataforma en donde están instalados.
- Aumenta la interoperabilidad entre servicios y programas de diferentes compañías y ubicados en diferentes lugares geográficos.

- Fomentan los estándares y protocolos basados en texto, haciendo más fácil acceder y entender su contenido y funcionamiento (pero, en general, produciendo una baja en su rendimiento).
- Al emplear HTTP, pueden utilizar un sistema firewall sin cambiar las reglas de filtrado (7).

### *1.2.1 Comparación entre Rest y Soap*

Actualmente REST (*Representational State Transfer*) y SOAP se denominan "Web de servicios," y uno a menudo se utiliza en lugar de la otra, pero son totalmente diferentes enfoques. REST es un estilo de arquitectura mientras que SOAP es una especificación de protocolo para intercambiar datos entre dos extremos y usado mayormente en aplicaciones empresariales (7).

Según Mike Rozlog director de productos de Embarcadero Technologies, SOAP es bastante maduro y bien definido y viene con una especificación completa. El enfoque REST es sólo eso, un enfoque. Una de las ventajas de SOAP es el uso del transporte "genérico". Mientras REST utiliza HTTP / HTTPS, SOAP puede utilizar casi cualquier transporte para enviar la solicitud. Además REST no tiene un sistema de mensajería estándar y no puede lidiar con la comunicación de fallos, SOAP proporciona fiabilidad en este sentido.

Por lo anteriormente planteado en el presente trabajo de diploma se decide usar el servicio web SOAP, ya que se ajusta a las características del problema de investigación, donde se requiere el uso de servicio estructurado y la aplicación es de tipo empresarial.

### *1.2.2 Protocolo Simple de Acceso a Datos SOAP*

SOAP es un protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando el protocolo de transferencia de hipertexto HTTP. Está basado en XML como se muestra en la Fig 1, a diferencia del DCOM y CORBA que son binarios, esto facilita la lectura por parte de los humanos, pero también los mensajes resultan más largos y, por lo tanto, considerablemente más lentos de transferir. Existen múltiples tipos de modelos de mensajes en SOAP, pero el más común es RPC (Llamada a Procedimiento Remoto), en donde un nodo de red (el cliente) envía un mensaje de solicitud a otro nodo (el servidor) y el servidor inmediatamente responde el mensaje al cliente. Los mensajes SOAP son independientes del sistema operativo y pueden transportarse en varios protocolos de internet como SMTP, MIME y HTTP (9).

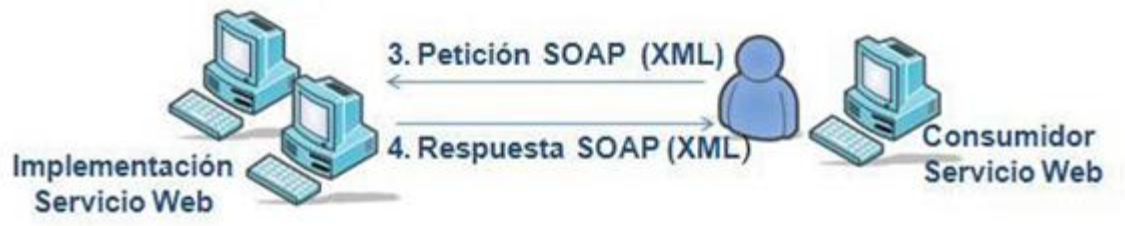


**Fig 1: Arquitectura SOA tradicional**

### *1.2.3 Lenguaje de descripción de servicios Web WSDL*

WSDL es el lenguaje de descripción de servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. También se usa a menudo en combinación con SOAP y *XML Schema*. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de *XML Schema*. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL (10).

### *1.2.4 Lenguaje de Marcado Extensible XML*

XML fue creado al amparo del W3C *(Word Wide Web Consortium)*, organismo que vela por el desarrollo de WWW partiendo de las amplias especificaciones de SGML *(Standard Generalized Markup Language)*. XML es un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. Es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados. Tiene otras aplicaciones entre las que destaca su uso como estándar para el intercambio de datos entre diversas aplicaciones o software con lenguajes privados como en el caso del SOAP. Es la base sobre el que se asientan los servicios web (8).

## 1.3 Arquitectura de la Solución

La arquitectura define el sistema de software en términos de componentes computacionales y las interacciones entre los mismos. Es una representación que permite analizar la efectividad del diseño para cumplir los requerimientos establecidos. Permite alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil. Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto (27).

La Arquitectura Orientada a Servicios **SOA**, define la utilización de servicios para dar soporte a los requisitos de software del usuario. Es una arquitectura para diseñar y desarrollar sistemas distribuidos, los cuales incluyen facilidad y flexibilidad de integración con sistemas legados, reduciendo costos de implementación. Permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e invocación de servicios. Se selecciona SOA como arquitectura a utilizar en la implementación de la solución ya que define las siguientes capas de software (35):

- Aplicaciones básicas: Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- De exposición de funcionalidades: Donde las funcionalidades de la capa aplicativa son expuestas en forma de servicios (generalmente como servicios web).
- De composición de procesos: Que define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio.
- De entrega: Donde los servicios son desplegados a los usuarios finales.

El modelo arquitectónico **cliente-servidor** es un modelo de sistema que se organiza como un conjunto de servicios y servidores asociados, más unos clientes que acceden y usan los servicios. Los principales componentes de este modelo son (28):

- Un conjunto de servidores que ofrecen servicios a otros subsistemas. Por ejemplo, los servidores de ficheros que ofrecen servicios de gestión de ficheros y servidores de compilación, que ofrecen servicios de compilación de lenguajes de programación.

- Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores. Estos son normalmente subsistemas en sí mismos. Puede haber varias instancias de un programa cliente ejecutándose concurrentemente.

- Una red que permite a los clientes acceder a estos servicios. Esto no es estrictamente necesario ya que los clientes y los servidores podrían ejecutarse sobre una única máquina. En la práctica, sin embargo, la mayoría de los sistemas cliente-servidor se implementan como sistemas distribuidos.

Los clientes pueden conocer los nombres de los servidores disponibles y los servicios que estos proporcionan. Sin embargo, los servidores no necesitan conocer la identidad de los clientes o cuántos clientes tienen. Los clientes acceden a los servicios proporcionados por un servidor a través de llamadas o procedimientos remotos usando un protocolo de petición-respuesta tal como el protocolo HTTP usado en la WWW. Básicamente, un cliente realiza una petición a un servidor y este le responde.

### *1.3.1 Patrón Arquitectónico MVC (Modelo Vista Controlador)*

Los patrones arquitectónicos definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades. También tienen una serie de directivas para organizar los componentes con el objetivo de facilitar la tarea del diseño de tal sistema (30).

El patrón arquitectónico **MVC** separa la lógica de negocio (el modelo) y la presentación (la vista), por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Por ejemplo, si una misma aplicación debe ejecutarse tanto en un navegador estándar como en el navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo, manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (31).

### *1.3.2 Patrones de diseño GRASP y GOF*

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. En general, un patrón de diseño se compone de cuatro elementos esenciales: nombre del patrón, problema, solución y consecuencias (32).

**GRASP** es un acrónimo que significa *General Responsibility Asignment Software Patterns* (Patrones Generales de Software para Asignar Responsabilidades). El nombre se eligió para indicar la importancia de captar estos principios. Estos patrones describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones; los principales son (33):

El patrón **Experto** asigna una responsabilidad a la clase que cuenta con la información necesaria para llevar a cabo la tarea requerida, creando así un comportamiento de clases sencillas con mejor estructura.

El patrón **Creador**, al igual que los sistemas orientados a objetos, crea un objeto de la clase con la información requerida. En otras palabras, le asigna a un creador la responsabilidad de crear una instancia de la clase con la información para cierta necesidad.

El patrón **Alta Cohesión** consiste en asignar una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

El patrón **Bajo Acoplamiento** estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases. Mientras que una con alto (o fuerte) acoplamiento necesita de terceras. Este tipo de clases no es conveniente ya que presentan los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

El patrón **Controlador** es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Los patrones de diseño **GoF** (*Gang of Four*) recopilan y documentan 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Los patrones de diseño en el grupo de GoF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (16):

Los patrones **Creacionales** tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o

inicializados. Los patrones contenidos en este grupo son: Fábrica Abstracta, Constructor virtual, Método de fabricación, Prototipo e Instancia única.

Los patrones **estructurales** describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. Los patrones contenidos en este grupo son: Adaptador, Puente, Objeto compuesto, Envoltorio, Fachada y Peso ligero.

Los patrones de **comportamiento** ayudan a definir la comunicación e interacción entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

## 1.4 Metodología para el desarrollo de software

Teniendo en cuenta que se tiene un proyecto pequeño integrado por una persona y se dispone de corto tiempo para su desarrollo, se decide el uso de una metodología ágil para ganar en tiempo, esfuerzo y evitar generar demasiada documentación.

Las **metodologías ágiles** se aplican en proyectos pequeños que resuelven problemas concretos, donde se busca un mayor funcionamiento del *software* que conseguir una buena documentación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste. Las metodologías ágiles presentan diversas ventajas, entre las que se encuentran (12):

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos breves de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Atención continúa a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.

La metodología **OpenUp** es una metodología ágil que aplica un enfoque iterativo e incremental dirigida a la gestión y desarrollo de proyectos de software. Se centra en una arquitectura temprana para reducir al mínimo los riesgos y organizar el desarrollo del software. Mantiene las características esenciales de RUP (Proceso Unificado de Rational), en el cual se incluyen el desarrollo incremental, el empleo de casos de uso y escenarios, y el diseño basado en la arquitectura. Permite el desarrollo de aplicaciones mediante la

generación de artefactos ligeros apropiados, utilizando UML (Lenguaje Unificado de Modelado) e incrementando así las probabilidades de éxito en función de costo, tiempo y alcance (13).

OpenUp propone seis flujos de trabajo (13):

- Requisitos: Se realizan entrevistas con el cliente para comprender el problema a resolver y se definen los requisitos.
- Arquitectura: Se realiza el diseño de los requisitos que serán después implementados.
- Desarrollo: Se realiza la implementación del sistema basándose en el diseño realizado.
- Prueba: Busca los defectos a lo largo del ciclo de vida.
- Gestión del Proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- Gestión de la configuración y cambio: explica el control de cambios en los artefactos, asegurando una evolución sincronizada del conjunto de productos de trabajo que componen el sistema de software.

Se decide utilizar OpenUp como guía en el proceso de desarrollo de software ya que esta metodología es flexible al cambio, beneficia el desarrollo de proyectos pequeños y de corto plazo. Además, es la metodología de desarrollo propuesta por la línea base de la arquitectura del departamento Integración de Soluciones.

## 1.5 Tecnologías y herramientas empleadas en el desarrollo de la solución

Las tecnologías son el conjunto de conocimientos técnicos, que permiten diseñar y crear bienes y servicios. La correcta selección de las herramientas y tecnologías informáticas a través de un estudio profundo de las características que las identifican, permite agilizar y facilitar el desarrollo de los sistemas informáticos (33).

### *1.5.1 Lenguaje Unificado de Modelado (UML) 2.0*

El Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar elementos conceptuales como los procesos de negocio y funciones de sistema, además de otros más concretos como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables (14).

Es importante resaltar que UML se usa para especificar y no para describir métodos o procesos y para lo cual no constituye una guía al desarrollador en la forma de realizar el análisis y diseño orientado a objetos. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar (14).

### *1.5.2 Herramienta de Modelado*

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) proponen una nueva filosofía del concepto de ciclo de vida basado en la automatización. Proporcionan un conjunto de herramientas bien integradas que, enmarcadas dentro de una determinada metodología, permiten automatizar las fases del ciclo de vida de un sistema de software. Con estas se intenta conseguir una mejora en la productividad y en la calidad del producto final. Para alcanzar estas ventajas se define un conjunto de objetivos (15):

- Automatizar e integrar las tareas de las distintas etapas del ciclo de vida, junto con la gestión de proyectos de software.
- Mejorar la calidad mediante comprobación de errores.
- Automatizar la generación de la documentación.
- Facilitar que se pueda compartir la información entre varios proyectos y la reutilización del software.
- Aportar un entorno de desarrollo interactivo.
- Acercar el desarrollo al usuario facilitando la creación de prototipos.
- Simplificar la labor de mantenimiento.

***Visual Paradigm for UML 8.0*** es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (16).

Entre sus principales características están:

- Disponibilidad en múltiples plataformas (Windows, GNU/Linux).

- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia gratuita y comercial.
- Fácil de instalar y actualizar.
- Diagramas de procesos de negocio - proceso, decisión, actor de negocio.
- Diagramas de flujo de datos.
- Editor de figuras.

### *1.5.3 Lenguaje de programación*

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Puede usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación (17).

El lenguaje de programación **PHP** (*Hypertext Preprocessor*), es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es de código abierto, muy popular, especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Entre sus principales características están (18):

- Es un lenguaje multiplataforma.
- Soporte integrado para SOAP.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.

- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de Programación Orientada a Objetos (POO).

Se decidió emplear el lenguaje PHP en su versión 5.3.8 por su soporte para *namespace* (Espacio de nombre) una nueva incorporación y de los cambios más significativos de PHP 5.3.

El lenguaje de programación **JavaScript** es utilizado para realizar acciones dentro del ámbito de una página web, crea efectos especiales en las páginas y define interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único con que cuenta este lenguaje, es el propio navegador (19).

Se decidió, para dar solución al problema de investigación, utilizar como lenguaje de programación del lado del cliente a JavaScript, ya que es independiente de la plataforma en la que se programe y se podrá ejecutar en cualquier navegador, además por tener grandes propiedades para crear y manejar ricas aplicaciones web.

### *1.5.4 Marcos de Trabajo*

Los marcos de trabajo son estructuras lógicas de los medios en los que un software puede ser diseñado e implementado. Permiten la utilización de uno o más lenguajes de programación, a menudo acompañada de una serie de herramientas para apoyar el desarrollo de software, tales como un IDE, depurador y otras herramientas diseñadas para aumentar la velocidad de desarrollo del producto. El framework o la infraestructura digital es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software (20).

El marco de trabajo **ExtJS 4.2** es una librería JavaScript 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, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente.

Para dar solución al problema de la investigación se decidió escoger como marco de trabajo ExtJS por ser una tecnología potenciada de estructura orientada a componentes a diferencia de Dojo y jQuery+jQUI que se centran en el *Document Object Model* o DOM. Además, el Sistema genérico de trazas será creado por el departamento Integración de Soluciones el cual mantiene una línea de desarrollo donde se utiliza mayormente ExtJS a diferencia de otros como YUI y Quuxdoo también orientados a objetos (21).

El marco de trabajo ***Symfony* 2.1.1** es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. *Symfony* se diseñó para que se ajustara a los siguientes requisitos (20):

- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Extensible: está desarrollado en su totalidad alrededor de Bundles, los cuales son directorios que contienen todo tipo de archivos (clases php y archivos web como JavaScript, css e imágenes), dentro de una estructura jerarquizada de directorios.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.

*Symfony* es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas Unix, GNU/Linux, como en plataformas Windows. Está basado en el patrón Modelo Vista Controlador.

Se decidió, para dar solución al problema de investigación, escoger como marco de trabajo *Symfony* en su versión 2.1.1 por las herramientas que le brinda a los programadores para mejorar su productividad, por ser desarrollado totalmente en PHP 5. Además, por estar basado en el patrón Modelo-Vista-Controlador y

por la automatización de tareas comunes, lo que trae consigo la posibilidad de que un desarrollador se centre solamente en las características específicas del software que está desarrollando.

### *1.5.5 Entorno Integrado de Desarrollo NetBeans*

Un IDE (Entorno de Desarrollo Integrado) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien utilizarse para varios. El IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de GUI (interfaz gráfica).

El entorno de desarrollo integrado ***NetBeans*** es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE *NetBeans*. Es un producto libre y gratuito sin restricciones de uso, entre sus características se encuentra la modularidad: todas las funciones del IDE son provistas por módulos, cada módulo provee una función bien definida (22).

La versión que se seleccionó para el desarrollo de la solución del problema de la investigación fue el *NetBeans* IDE 7.2. Esta versión ofrece un rendimiento significativamente mejorado y brinda soporte al framework *Symfony* 2, tecnología a usar en el desarrollo del presente trabajo de diploma.

### *1.5.6 Sistema gestor de bases de datos PostgreSQL 9.1*

Un sistema gestor de bases de datos (SGBD) es un programa orientado a la gestión y diseño de bases de datos. Además de permitir el desarrollo y construcción de las BD (bases de datos), un SGBD permite operar directamente en las tablas de la base de datos, haciendo posible la navegación y visualización de los registros almacenados en las tablas de la misma, su edición, búsqueda, inserción y eliminación. En sí mismo, el SGBD no sólo actúa como una herramienta de construcción de BD sino como una interfaz que permite interactuar y explotar sus contenidos. Para ello un SGBD está compuesto por diversos subsistemas: el motor de la base de datos, el sistema de definición de datos, el sistema de manipulación y gestión, el sistema de herramientas y aplicaciones y el módulo de administración. Entre sus principales características están (23):

- Capacidad para almacenar datos en la BD, acceder a ellos, insertar otros nuevos, modificarlos y eliminarlos.
- Asegura el acceso de múltiples usuarios manipulando o editando sus contenidos mediante un control de la concurrencia. Esto significa que el sistema debe proporcionar un orden de prioridad en los procesos que se llevan a cabo. Al conjunto de procesos que llevan a cabo múltiples usuarios se le denomina transacciones.
- Permite la gestión de los privilegios de acceso al sistema para gestionar el acceso y las restricciones del mismo a diversos tipos de usuarios.
- Proporciona las herramientas de mantenimiento necesarias para la consistencia de los datos de la BD.

El sistema de gestión de bases de datos **PostgreSQL 9.1**, distribuido bajo licencia BSD (*Berkeley Software Distribution*) y con código fuente disponible libremente, utiliza un modelo cliente/servidor donde la conexión puede ocurrir vía TCP/IP ó sockets locales. Usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Entre sus principales características están (24):

- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Soporta funciones y procedimientos en el servidor.
- Está disponible para los Sistemas Operativos GNU/Linux y Windows.
- Soporta totalmente el modelo relacional de bases de datos.
- Posee extensiones propias de SQL para realizar consultas sobre la base de datos.

El gestor de bases de datos a utilizar en la solución es PostgreSQL versión 9.1 ya que permite gestionar grandes volúmenes de datos y fortalecer la seguridad de las bases de datos, aspectos claves a tener en cuenta en el manejo de las mismas.

### *1.5.7 Interfaz gráfica para administrar bases de datos pgAdmin 1.14.1*

PgAdmin es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia *Open Source* (Código Abierto). Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. Algunas de las nuevas características incorporadas a esta versión son (25):

- **Índice:** el cuadro de diálogo propiedades del índice muestra todas las intercalaciones disponibles y le permite establecer una colación para char, varchar y columnas de texto.
- **Propiedades del objeto:** el diálogo de propiedades está disponible como de costumbre (clic derecho sobre el nombre del objeto y se selecciona "Propiedades" del menú). Hay un nuevo acceso directo para obtener el mismo resultado: Ctrl-Alt-Enter.

### *1.5.8 Servidor web Apache 2.2.22*

Los servidores web permiten a los clientes compartir datos, documentos y multimedia. Aunque parte de la tecnología Cliente - Servidor, el servidor web aporta ventajas adicionales en aspectos muy importantes como son:

- El servidor web vuelca información con un simple clic del ratón a través de un proceso de hipervínculo.
- Dado que el servidor web es de tan fácil acceso, ello hace posible publicar información de forma instantánea mediante un simple almacenamiento de la misma en el servidor.

El servidor web **Apache 2.2.22** es de código abierto, compatible con plataformas Unix, Microsoft Windows y Macintosh que implementa el protocolo HTTP. Su arquitectura se rige por el esquema petición-respuesta. El servidor web Apache presenta muchas ventajas, entre ellas que es altamente configurable y permite la autenticación de bases de datos. Sin embargo, fue muy criticado porque carece de una interfaz gráfica que permita que usuarios sin conocimientos técnicos avanzados accedan a su configuración. La versión que será utilizada en la solución del problema de la investigación es Apache 2.2.22, teniendo en cuenta las siguientes características (26):

- Es principalmente una liberación de seguridad y corrección de errores.
- Filtro inteligente.
- Almacenamiento en caché mejorado.
- AJP Proxy.

- Equilibrio de carga.
- Soporte de proxy de cierre normal.
- Soporte de archivos grandes.
- Rediseñado y autenticación / autorización.

## Conclusiones del Capítulo

Luego de haber realizado una revisión de los sistemas de análisis de trazas existentes en el ámbito nacional e internacional y las ventajas y desventajas de las herramientas para el desarrollo de los subsistemas definidas para el análisis, se demuestra que las herramientas existentes no tienen las características requeridas para dar respuesta al problema planteado. Por lo que se hace necesaria la implementación de un sistema genérico que garantice la gestión de trazas de los productos desarrollados en el departamento Integración de Soluciones a través de un servicio web. Para guiar el proceso de desarrollo del software se utilizará la metodología OpenUP y como herramienta para el modelado *Visual Paradigm 8.0 for UML* empleando el lenguaje de modelado UML 2.0. Durante la implementación de la solución se hará uso del IDE de desarrollo Netbeans 7.2, utilizando como lenguajes de programación PHP 5.3.8 y JavaScript, soportados por los marcos de trabajos Symfony 2.1.1 y ExtJS 4.2 respectivamente. Además, para la definición de la base de datos se usará como gestor PostgreSQL 9.1, para la administración de la base de datos PgAdmin 1.14.1 y como servidor web Apache 2.2.22.

# CAPÍTULO 2 "ANÁLISIS Y DISEÑO DEL SISTEMA GENÉRICO DE GESTIÓN DE TRAZAS"

## Introducción

En el presente capítulo se realiza una descripción de las principales características del sistema, detallando la propuesta de solución. Se realiza una representación visual de clases conceptuales del entorno real de los objetos del sistema a través del modelo de dominio. Se ilustran los principales requisitos funcionales y no funcionales, así como los artefactos generados y los patrones de diseño teniendo en cuenta la metodología seleccionada. Se especifica la estructura física de la solución que se propone mediante el diagrama de despliegue.

## 2.1 Modelo de Dominio

Un Modelo del Dominio es una representación de las clases conceptuales o entidades del mundo real, no de componentes de software; puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Esto ayuda a los usuarios, clientes y desarrolladores e interesados a utilizar un lenguaje común para poder entender el contexto en que se desarrolla el sistema. Es decir, expresa los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, y no incluyen las responsabilidades de las personas que ejecutan las actividades (34). Basado en esta definición, se muestra a continuación el modelo de dominio del Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones.

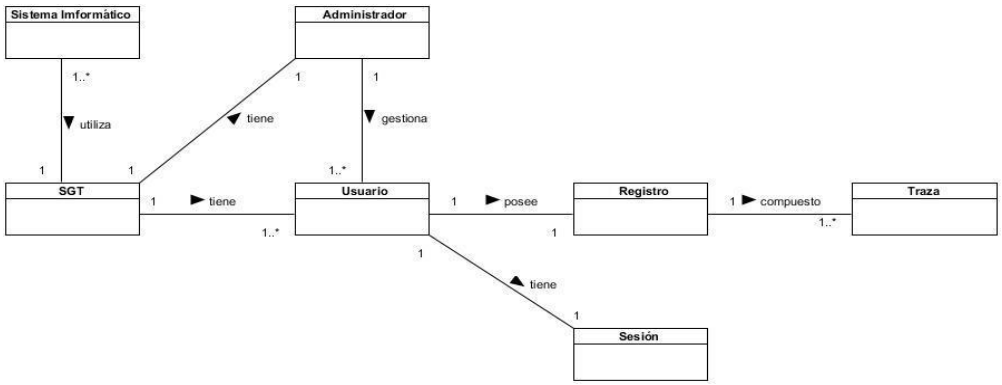


Fig 2: Modelo de Dominio del Sistema genérico de gestión de trazas

En el diagrama (ver Fig 2), el Sistema Informático representa a los productos desarrollados por el departamento Integración de Soluciones, como SIGDAT y SIGE entre otros. Los cuales realizan múltiples operaciones y utilizan el Sistema genérico de gestión de trazas (SGT) para almacenar información de las mismas en un registro en forma de trazas. A continuación se realiza una breve descripción del modelo de dominio explicando cada uno de los conceptos que se encuentran representados en el mismo:

- **Sistema Informático:** Entidad que representa a las aplicaciones desarrolladas por el departamento Integración de Soluciones, las cuales harán uso del SGT para gestionar sus trazas.
- **SGT:** representa el Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones, el cual le permite a los usuarios visualizar las trazas generadas por los sistemas informáticos. Este sistema debe permitir la realización de filtrado, búsqueda y exportación de registros.
- **Administrador:** Actor del sistema encargado de administrar los usuarios registrados en la aplicación.
- **Usuario:** Entidad que representa a los productos informáticos registrados en el Sistema genérico de gestión de trazas, los cuales tendrán una sesión donde podrán visualizar sus trazas.
- **Trazas:** representa una clase contenedora de información del negocio, en este caso recopila información de cada una de las operaciones realizadas por los sistemas informáticos.
- **Sesión:** representa la conexión realizada por un usuario al sistema, el cual carga las trazas asociadas al usuario.
- **Registro:** Entidad encargada de archivar todas las trazas generadas por las operaciones ejecutadas por los sistemas informáticos.

## 2.2 Requisitos del sistema

Un requisito es una “condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado”. Se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes (35).

Los **requisitos funcionales** (RF) definen los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares. Son condiciones que el sistema ha de cumplir (35). El Sistema genérico de gestión de trazas para los

productos desarrollados en el departamento Integración de Soluciones, debe cumplir con los requisitos funcionales que a continuación se describen:

**RF 1.** Adicionar usuario
**Descripción:** se registra en el sistema un nuevo usuario.
**Entrada:** se registran los detalles del nuevo usuario (nombre, usuario, contraseña).
**Salida:** nuevo usuario.

**RF 2.** Autenticar usuario
**Descripción:** permitirá a los usuarios realizar el proceso de autenticación en la aplicación, insertando a la misma los datos correspondientes (usuario y contraseña).
**Entrada:** contraseña y usuario de la persona que desea entrar al sistema.
**Salida:** se inicia la sesión con los datos asociados al usuario.

**RF 3.** Editar usuario
**Descripción:** permitirá modificar cualquiera de los campos que componen a la entidad usuario.
**Entrada:** el o los datos de los campos que se desea modificar del usuario (nombre, usuario, contraseña).
**Salida:** usuario modificado.

**RF 4.** Listar usuario
**Descripción:** permitirá visualizar un listado de los usuarios en el sistema.
**Salida:** listado de usuarios registrados en el sistema.

**RF 5.** Eliminar usuario
**Descripción:** se elimina del sistema el usuario seleccionado previamente.
**Entrada:** se selecciona de un listado de usuarios el que se desea eliminar.
**Salida**: listado de usuarios actualizado.

**RF 6.** Insertar Trazas
**Descripción:** permitirá la inserción de las trazas en la base de datos desde un servicio web. Los dos primeros parámetros (usuario, contraseña) son para la identificación del usuario proveedor del servicio.

**Entrada:** usuario, contraseña y los detalles de la nueva traza (operación, objeto, usuario, descripción, dirección).
**Salida:** se inserta la traza en la base de datos.

**RF 7.** Listar trazas
**Descripción:** permitirá mostrar el listado de trazas asociadas al usuario autenticado.
**Salida:** listado de las trazas almacenadas en el sistema asociadas al usuario conectado.

**RF 8.** Búsqueda Simple de trazas
**Descripción:** permitirá realizar una búsqueda de las trazas en el sistema a partir de un criterio introducido por el usuario.
**Entrada:** criterio de búsqueda.
**Salida:** listado de trazas que correspondan al criterio de búsqueda previamente introducido.

**RF 9.** Búsqueda Avanzada de trazas
**Descripción:** permitirá realizar una búsqueda de las trazas en el reporte a partir de los siguientes criterios: operación, objeto, usuario, descripción, fecha, hora, dirección.
**Entrada:** criterios de búsqueda (operación, objeto, usuario, descripción, fecha, hora, dirección).
**Salida:** listado de trazas que correspondan con los criterios de búsqueda previamente introducidos.

**RF 10.** Exportar trazas.
**Descripción:** permitirá exportar un reporte de trazas en formato pdf.
**Salida:** listado de trazas en formato pdf.

**RF 11.** Cerrar sesión
**Descripción:** permitirá cerrar la sesión de un usuario.
**Salida:** se cierra la sesión del usuario.

Los **requisitos no funcionales** (RNF) son restricciones de los servicios o funciones ofrecidos por el sistema. Definen propiedades emergentes, tales como el tiempo de respuesta, las necesidades de almacenamiento y la fiabilidad; son condiciones que todo sistema debe cumplir (35). A continuación se muestran los RNF asociados a la investigación:

**Requisitos de Usabilidad**

**RNF 1:** Tipo de Aplicación Informática.

El sistema deberá presentar facilidades al usuario para el manejo de la información, debe ser una herramienta WEB con características muy similares a las aplicaciones de escritorio en cuanto a diseño de interfaz. Se pretende una vista descriptiva, sencilla y fácil de usar para la realización de todas las operaciones, con el objetivo de propiciar un buen entendimiento a los usuarios finales.

**RNF 2:** Finalidad

El objetivo deseado por la aplicación es facilitar la visualización de las trazas, brindando las opciones de listar, buscar y exportar para cubrir las necesidades del usuario.

**RNF 3:** Software requerido para desplegar y utilizar la aplicación

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 hasta 14.04, Debian 4 GNU/Linux.
- Paquetes: apache2, php5, php5-cli, php5-pgsql, php-soap, php-curl.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 hasta 14.04, Debian 4 GNU/Linux.
- PostgreSQL versión 9.1 o superior.
- PGAdmin III versión 1.14.1 o algún administrador para postgreSQL.
- Usuario con privilegios para instalar la base de datos.

**Requisitos de Hardware**

**RNF 4.** Las PC clientes deben cumplir con los siguientes requisitos de hardware:

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 256 MB.

Las PC servidor deben cumplir con los siguientes requisitos de hardware:

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 1 GB.
- Disco Duro con 10Gb de capacidad para instalar el sistema.

**Requisitos de Confiabilidad**

**RNF 5:** Fiabilidad

El sistema debe estar disponible las 24 horas del día. En caso de fallo, pudiera estar fuera de servicio por un período de 72 horas máximo.

Algunos errores que pueden resultar críticos son:

- Que salgan de funcionamiento las bases de datos desde donde se extraen los reportes o que no exista conectividad hacia ellas.
- Que falle el servidor donde se despliegue la solución.

**Requisitos de Eficiencia**

**RNF 6:** Requisito de Eficiencia

La eficiencia del sistema depende en gran medida de la velocidad de conexión a las bases de datos donde se encuentre, así como del volumen de información contenido en las mismas.

**RNF 7:** Tiempo máximo de respuesta para visualizar un reporte de trazas y las opciones de buscar.

El tiempo máximo de ejecución para estas opciones no debe sobrepasar los 8 segundos.

**RNF 8:** Tiempo máximo de respuesta para la opción de exportar un reporte de trazas.

El tiempo máximo de ejecución para estas opciones no debe sobrepasar los 10 segundos.

**Requisitos de Restricciones de diseño e implementación.**

**RNF 9:** Lenguaje y marco de trabajo para el desarrollo del sistema del lado del servidor.

El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.3.8 y hacer uso del marco de trabajo Symfony 2.1.1.

**RNF 10:** Lenguaje y marco de trabajo para el desarrollo del sistema del lado del cliente.

El sistema deberá ser implementado en el lenguaje de programación JavaScript y hacer uso del marco de trabajo ExtJS.

**Requisitos de Interfaz**

**RNF 11:** Interfaces de usuario

Las interfaces de usuario deberán diseñarse a modo de aplicaciones RIA (*Rich Internet Application*), permitiendo a los usuarios contar con aplicaciones web con una apariencia de usuario similar a la de las aplicaciones de escritorio.

**RNF 12:** Interfaces de Comunicación

El sistema puede ser desplegado sobre red LAN, MAN o WAM; siempre y cuando la velocidad de conexión sea mayor que 1 Mbit/s.

**Requisitos Legales, de Derecho de Autor y otros**

**RNF 13:** Legales y derecho de autor

Al finalizar el desarrollo de la aplicación Informática, la Dirección Jurídica de ALBET definirá la licencia por la que se regirá el mismo.

**Estándares Aplicables**

**RNF 14:** Estándares aplicables

Durante el proceso de desarrollo de software, cada entregable es sometido a pruebas de liberación, en las que se evalúan las características y sub-características de calidad definidas por la ISO/IEC 9126. Como estándar de codificación se usará *CamelCase*. Para el diseño de la Base de Datos se usará un convenio para el nombre de las tablas, así como para las vistas y funciones que sean necesarias implementar.

## 2.3 Diagrama de casos de uso del sistema

Un modelo conceptual es una representación de conceptos en un dominio del problema, cuya cualidad esencial es representar elementos o conceptos del mundo real y no componentes del software. Un diagrama de casos de uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa (30). A continuación el diagrama de casos de uso del sistema.

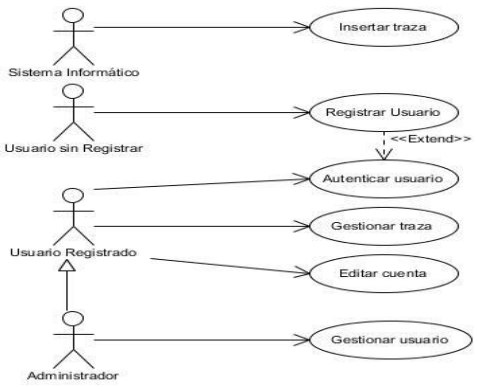


Fig 3: Diagrama de casos de usos del Sistema genérico de gestión de trazas

**Descripción de los casos de uso:**

El caso de uso **Insertar traza** permite insertar las trazas, generadas por las operaciones realizadas en los sistemas informáticos que consumen del servicio web publicado para enviar dichas trazas.

El caso de uso **Registrar Usuario** permite el registro de un nuevo usuario al Sistema de gestión de trazas, este pasa de ser un actor no registrado a un actor registrado.

El caso de uso **Autenticar usuario** permite realizar el proceso de autenticación de los usuarios del sistema, además de posibilitarles desconectarse del mismo. En este caso de uso se agrupa además el requisito adicionar usuario, ya que para autenticarse un usuario, este debe estar registrado en el sistema.

El caso de uso **Gestionar traza** permite a los usuarios registrados en el sistema visualizar, buscar y exportar las trazas almacenadas en el Sistema genérico de gestión de trazas.

El caso de uso **Editar cuenta** permite al usuario previamente autenticado modificar sus datos en el sistema.

El caso de uso **Gestionar usuario** permite al usuario con permisos de administrador del sistema eliminar o listar los usuarios registrados y visualizar sus registros de trazas.

### *2.3.1 Patrones de casos de uso del sistema*

Los patrones de casos de uso son comportamientos que deben existir en el sistema, describen el uso del mismo y cómo este interactúa con los usuarios. Modelan las operaciones que pueden ser realizadas tales como creación, lectura, actualización.

**Extensión:** Se aplica en los casos de uso Autenticar y Registrarse ya que para autenticarse se necesita estar registrado.

**Rol Común:** Se aplica cuando dos actores juegan el mismo rol sobre un caso de uso. Este patrón se manifiesta en el actor Usuario Registrado, el cual inicia los casos de usos Autenticar Usuario, Gestionar traza y Editar cuenta.

**CRUD parcial:** La variación denominada CRUD Parcial, indica que en caso de que solo algunas de las operaciones sean implicadas y no todo el conjunto de un CRUD completo no significa que necesariamente cada operación se deba expresar como casos de usos separados. Según el patrón CRUD, se pueden agrupar. Este patrón se manifiesta en el caso de uso Gestionar Usuario y Gestionar Traza.

### 2.3.2 Descripción del caso de uso significativo del sistema.

Tabla 1: Descripción del caso de uso Gestionar traza

| | | |
|---|---|---|
| **Caso de Uso:** | Gestionar traza | |
| **Actor:** | Usuario Registrado | |
| **Resumen:** | El caso de uso inicia con la autenticación de un usuario, inmediatamente se le muestra el reporte de trazas asociadas a este junto a las opciones de buscar y exportar. El caso de uso finaliza cuando se termina una de las operaciones que permite el caso de uso. | |
| **Precondiciones:** | El sistema debe estar instalado y ejecutándose correctamente.<br>El actor debe estar autenticado. | |
| **Referencias:** | RF7, RF8, RF9, RF10 | |
| **Prioridad:** | Critico | |
| **Complejidad:** | Media | |
| **Flujo Normal de Eventos** | | |
| **Acción del Actor** | | **Respuesta del Sistema** |
| **1.** | El caso de uso inicia cuando el actor Usuario Registrado es autenticado. | |
| **2.** | | El sistema muestra el listado de trazas existentes organizado por fechas, permitiendo las siguientes funcionalidades:<br>• Listar trazas. Ver Sesión 1: “Listar trazas”.<br>• Exportar en formato pdf, ver Sesión 2: “Exportar”.<br>• Actualizar listado de trazas, ver Sesión 3: “Actualizar listado”.<br>• Buscar una traza, ver Sesión 4: “Buscar traza”. |
| **Flujo Alterno al paso 1** | | |
| **1.a No se estableció la conexión con el servidor** | | |
| **1.** | | El sistema muestra un mensaje de notificación al actor Usuario informando que no se pueden obtener datos de la Base de Datos. |

<table>
<tr><th colspan="3">Sesión 1: “Listar trazas”</th></tr>
<tr><td>1.</td><td>El usuario se autentica en el sistema.</td><td></td></tr>
<tr><td>2.</td><td></td><td>El sistema muestra un listado con paginado de las trazas almacenadas por el usuario autenticado.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><th colspan="3">Sesión 2: “Exportar”</th></tr>
<tr><td>1.</td><td>El usuario selecciona la opción “Exportar”.</td><td></td></tr>
<tr><td>2.</td><td></td><td>El sistema muestra un menú brindando las opciones:<ul><li>Exportar la lista actual ver Sesión 2.1: “Vista Actual”.</li><li>Exportar todo el reporte ver Sesión 2.2: “Reporte Completo”.</li></ul></td></tr>
<tr><th colspan="3">Sesión 2.1: “Vista Actual”</th></tr>
<tr><td>1.</td><td>El usuario selecciona la opción “Vista Actual”.</td><td></td></tr>
<tr><td>2.</td><td></td><td>El sistema exporta el reporte ver Sesión 2.3: “Reporte” con la lista que esta siendo mostrada en ese momento.</td></tr>
<tr><th colspan="3">Sesión 2.2: “Reporte Completo”</th></tr>
<tr><td>1.</td><td>El usuario selecciona la opción “Reporte Completo”.</td><td></td></tr>
<tr><td>2.</td><td></td><td>El sistema exporta el reporte ver Sesión 2.3: “Reporte” con todas las trazas almacenadas.</td></tr>
</table>

| | | |
|---|---|---|
| **Sesión 2.3: "Reporte"** | | |
| **1.** | | El sistema muestra una vista previa del contenido a exportar junto al mensaje de aviso "El reporte se encuentra listo para exportar" y las opciones de "Cancelar" o "Exportar". |
| **4.** | El usuario selecciona la opción **"Exportar"**. | |
| **5.** | | El sistema muestra una ventana con las opciones de Guardar o Abrir el archivo creado en pdf. |
| Prototipo de Interfaz | | |
|  | | |
| **Flujo Alterno al paso 2.3** | | |
| **2.3.a Se cancela la operación** | | |
| **1.** | El usuario selecciona la opción **"Cancelar"**. | |
| **2.** | | El sistema cierra la ventana de muestra y cancela la operación. |
| **Sesión 3: "Actualizar listado"** | | |
| **1.** | El usuario selecciona la opción **"Actualizar lista"**. | |
| **2.** | | El sistema recarga el reporte completo de trazas. |
| **Sesión 4: "Buscar"** | | |
| | | El sistema muestra un campo para buscar un criterio y la opción de buscar de forma avanzada ver Sesión 4.1: "**Avanzada**". |

| | | |
|---|---|---|
| **2.** | El usuario introduce un criterio y selecciona la opción de **Buscar**. | |
| **3.** | | El sistema realizará una búsqueda donde el resultado serán las trazas en las cuales se encontraron coincidencias con el criterio introducido. |
| Prototipo de Interfaz | | |
|  | | |
| **Sesión 4.1: "Avanzada"** | | |
| **1.** | El usuario selecciona la opción **"Avanzada"**. | |
| **2.** | | El sistema muestra una ventana con campos para la búsqueda de una traza y los botones de Cancelar o Buscar. |
| **3.** | El usuario introduce el o los criterios y selecciona la opción de Buscar. | |
| **4.** | | El sistema realizará una búsqueda donde el resultado serán las trazas que coincidan con el o los criterios introducidos. |
| Prototipo de Interfaz | | |
|  | | |
| **Flujo Alterno al paso 4** | | |
| **4.a Se cancela la operación** | | |
| **1.** | El usuario selecciona la opción **"Cancelar"**. | |
| **2.** | | El sistema cancela la operación y cierra la interfaz de buscar. |

## 2.4 Modelo de Diseño

El modelo de diseño describe los atributos y métodos de cada clase, que sirve como una abstracción del modelo de aplicación y su código fuente. Representa los componentes de aplicación, su colaboración y las responsabilidades de cada una de ellas. En conclusión, se representa el uso de las clases dentro de la arquitectura en general del sistema (34).

### *2.4.1 Diagrama de Paquetes*

La estructura física de los componentes que conforman el sistema se realizará basado en los principios establecidos por Symfony 2.1.1 que estructuran el proyecto en paquetes o bundles. A continuación se muestra la estructura utilizada en el bundle del Sistema genérico de trazas para la gestión de trazas a través de servicios web.

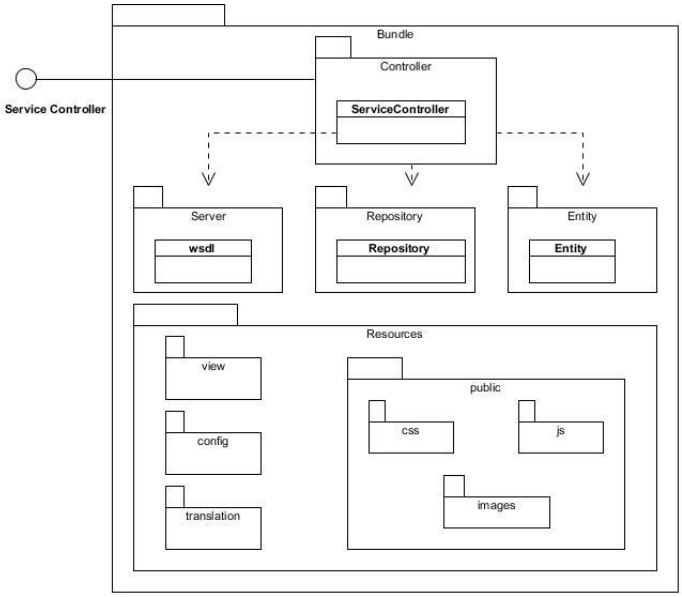


Fig 4: Diagrama de Paquetes del Sistema genérico de gestión de trazas

**Descripción del Diagrama de Paquetes**

El bundle de la aplicación estará compuesto por tres carpetas principales, la denominada *Controller* contiene todos los controladores del sistema. Estos manejaran la lógica con el uso de los repositorios y las entidades, en dependencia de las operaciones llevadas a cabo. Los repositorios, encargados de crear las sentencias del lenguaje de consulta Doctrine o DQL (del inglés *Doctrine Query Language*) de las entidades, se almacenarán en la carpeta *Repository*. La carpeta *Entity* contendrá todas las entidades

creadas para el manejo de datos. Otro bundle necesario será el *DirectBundle*, que permite mediante rutas la comunicación de ExtJs con Symfony.

Para la elaboración de la presentación se hará uso del marco de trabajo ExtJs 4.2, sus componentes también estarán organizados en forma de paquetes. Las acciones se llevarán a cabo en la carpeta *Controller*. Los *stores* encargados de realizar las peticiones en los controladores del servidor estarán contenidos en el paquete nominado *store* y los modelos en la carpeta *model*. En la carpeta *view* se almacenarán todas las vistas a usar en la aplicación.

### *2.4.2 Diagrama de clases del diseño*

Es un tipo de diagrama estático que describe la estructura de un sistema, mostrando sus clases y las relaciones estructurales y de herencia entre ellas. Incluye mucha más información, como los conjuntos de operaciones y propiedades que son implementadas para una interfaz gráfica. A continuación se muestra el diagrama de clases del diseño del caso de uso Gestionar Traza, donde se interpreta de forma gráfica el contenido de la implementación.

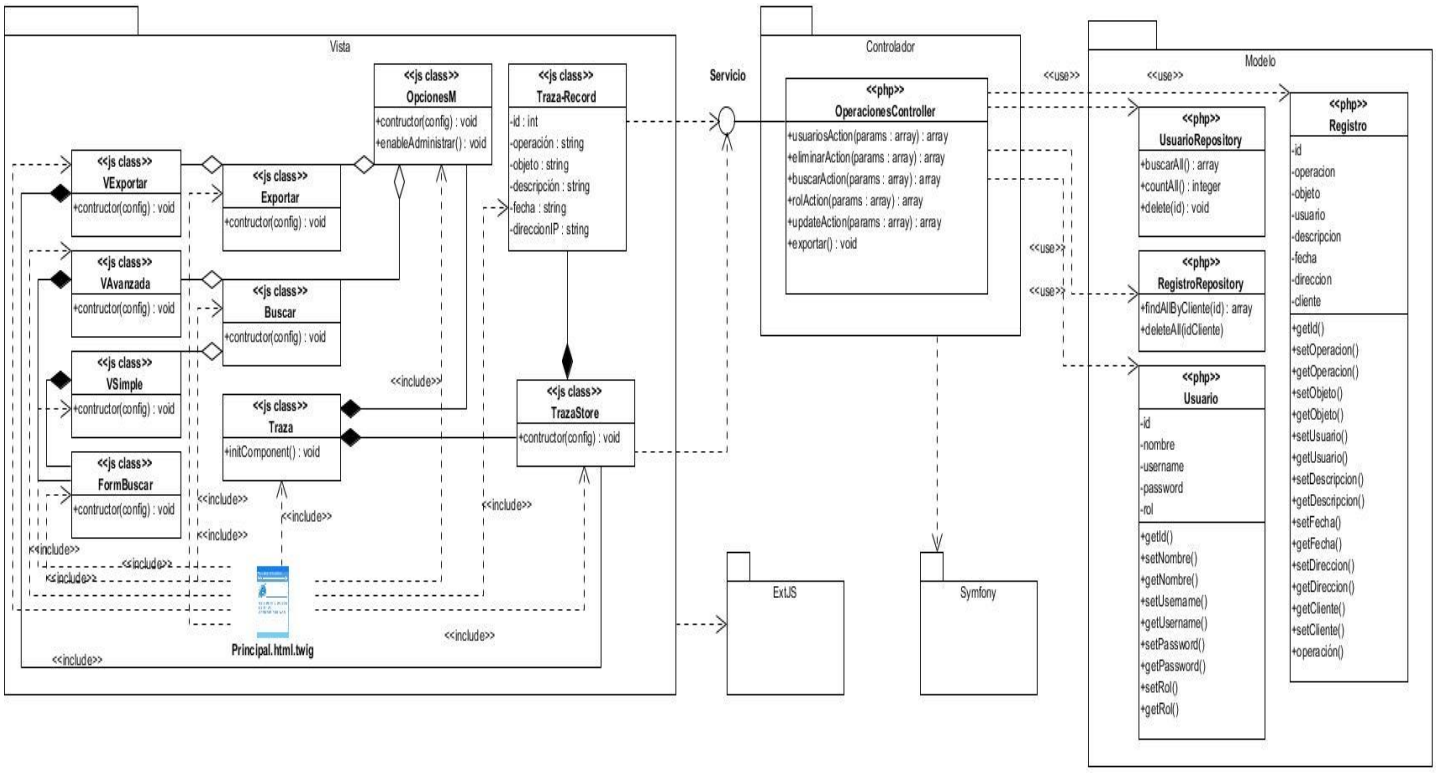


**Fig 5: Diagrama de clases del diseño del caso de uso Gestionar traza**

**Descripción del diagrama de clases del diseño del CU Gestionar traza**

El diagrama de clases del diseño (ver Fig 5) incluye todas las clases con sus métodos y atributos correspondientes del caso de uso Gestionar traza. Se encuentra organizado siguiendo la estructura del patrón arquitectónico Modelo-Vista-Controlador, lo cual trae consigo que las clases implementadas en el lado del cliente se encuentran en el paquete *Vista*, las clases por parte del servidor encargadas de la lógica del negocio en el paquete *Controlador* y las clases de acceso a datos en el paquete *Modelo*. Los tres paquetes separados conforman el subsistema con el cual interactúa el sistema.

Las clases contenidas dentro del paquete *Vista* proporcionan las distintas interfaces a mostrar al usuario, iniciando por la clase *Traza,* la cual carga todas las trazas almacenadas en la base de datos asociadas al usuario previamente autenticado. Esta se compone por la clase *TrazaStore* y este en si de *Traza-Record,* encargados de realizar la petición al servidor para cargar dichas trazas. Otra clase iniciada de forma recurrente es *OpcionesM*, la cual es encargada de mostrar las opciones representadas por las clases: *Exportar* y *BuscarTraza,* la que puede ser simple o avanzada.

Para la comunicación entre las vistas y los controladores se utilizó el bundle *DirectBundle*, el cual es una implementación de *ExtDirect* para Symfony 2. *ExtDirect* es una arquitectura de comunicación remota de JavaScript que permite reflejar los métodos del lado del servidor en el lado del cliente. Para realizar lo descrito anteriormente se utiliza como mecanismo de comunicación el RPC del marco de trabajo ExtJS. Los servicios a consumir del lado del cliente serán definidos a través de un API, que es la encargada de especificar cada una de las funciones a llamar en el controlador del lado del servidor, dependiendo de cada una de las acciones que se ejecuten en el lado del cliente. La clase *OperacionesController* contenida en el paquete controlador es la encargada de atender todas las peticiones realizadas por el cliente. Para esto hace uso de las clases *Usuario*, *Registro* y las responsables de facilitar las búsquedas sobre la base de datos *UsuarioRepository* y *RegistroRepository*.

### *2.4.3 Diagrama de secuencia*

Los casos de usos describen cómo interactúan los actores externos con el sistema. Durante esta interacción, un actor genera eventos en el sistema solicitando alguna operación. Los diagramas de secuencia pueden representar estas interacciones de los actores y las operaciones que inician. Se puede decir que muestran la interacción de un conjunto de objetos en una aplicación y se modela para cada caso

de uso. A continuación se muestra el diagrama de secuencia del requisito funcional ListarTrazas asociado al caso de uso Gestionar traza.

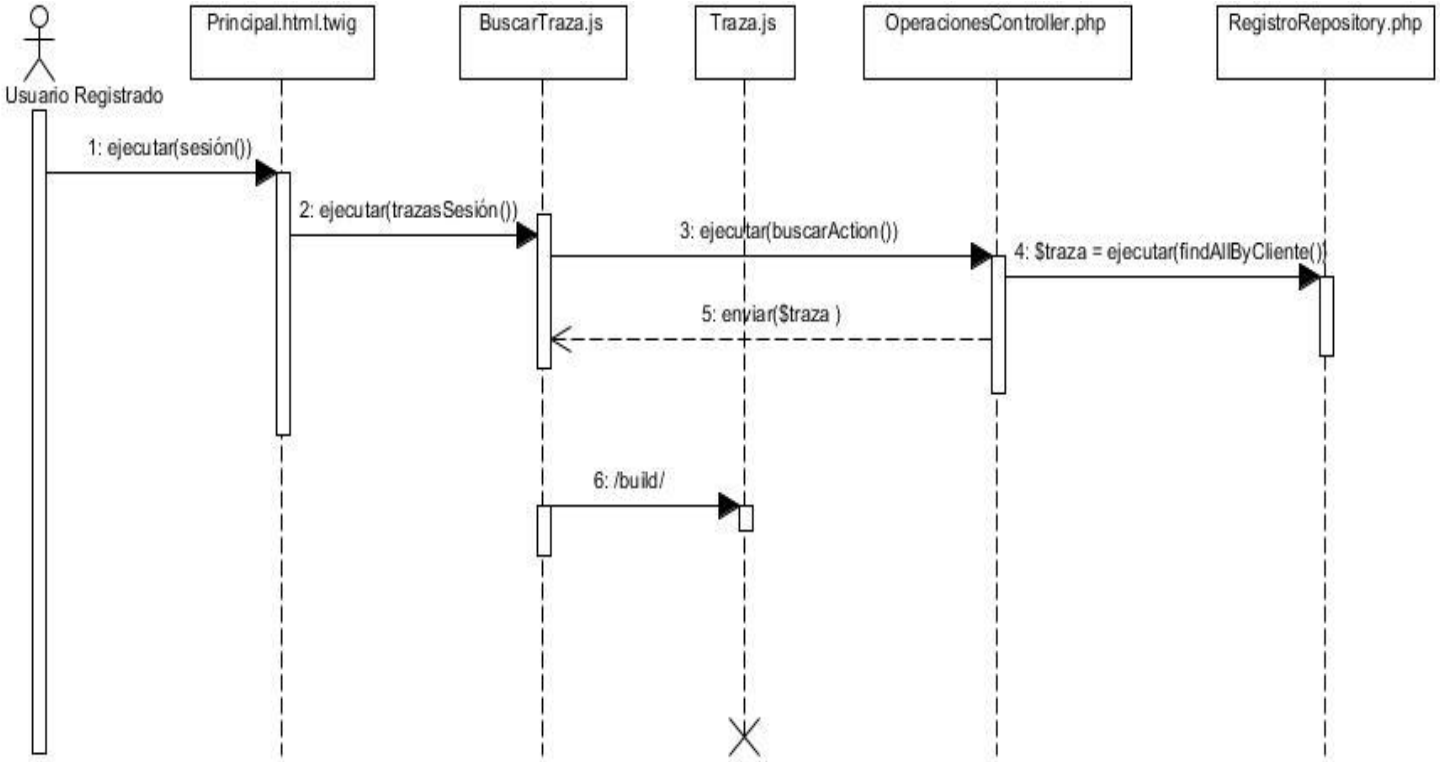


Fig 6: Diagrama de secuencia del caso de uso Gestionar traza, sección Listar Trazas.

**Descripción del diagrama de secuencia del requisito Listar Trazas**

En el diagrama de secuencia (ver Fig 6) el actor Usuario Registrado se autentica en el sistema invocando el método *ejecutar (sesión ())* iniciando la clase *BuscarTraza,* la cual realiza una petición al servidor mediante el método *ejecutar buscarAction ()* donde se requiere todas las trazas acumuladas en la base de datos asociadas al usuario previamente autenticado. Este es recibido por el controlador *OperacionesController* invocando el método *ejecutar(findAllByCliente())* de la clase *RegistroRepository,* retornando un arreglo con todas las trazas del usuario autenticado, las que son enviadas a la clase que inició la petición y esta lo muestra mediante la interfaz *Traza*.

### *2.4.4 Patrones GRASP*

Como se afirma en el Capítulo 1 (ver en la página 14) del presente documento, la utilización de patrones de diseño GRASP resulta de vital importancia para la asignación de responsabilidades a objetos. A continuación se describen los patrones de diseño GRASP aplicados en la solución.

**Controlador:** Es un objeto que no pertenece a la interfaz de usuario, es el responsable de recibir o manejar un evento del sistema. Define el método para la operación del sistema. El uso de este patrón se pone de manifiesto en la utilización de diversos controladores en el sistema, como por ejemplo *OperacionesController* y *ServicioController*.

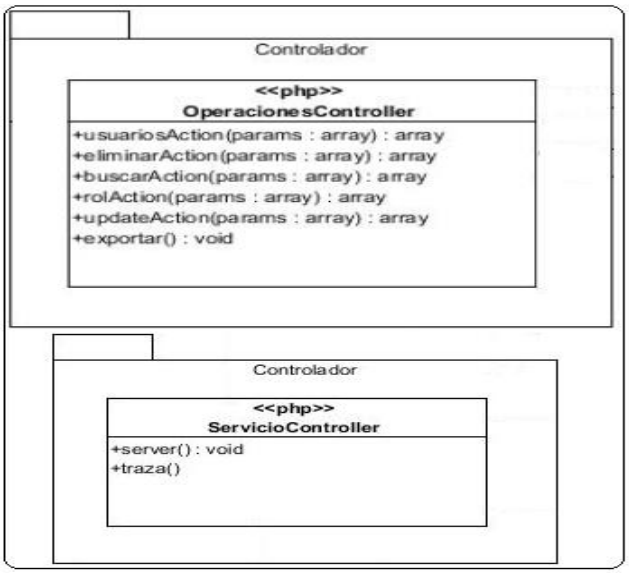


Fig 7: Aplicación del patrón Controlador en el desarrollo de la solución

**Experto:** Crea un objeto de la clase con la información requerida. Este patrón se muestra en las clases implementadas en la parte del cliente, donde las distintas vistas y operaciones son separadas por clases que cuentan con la información necesaria para llevar a cabo la tarea requerida.

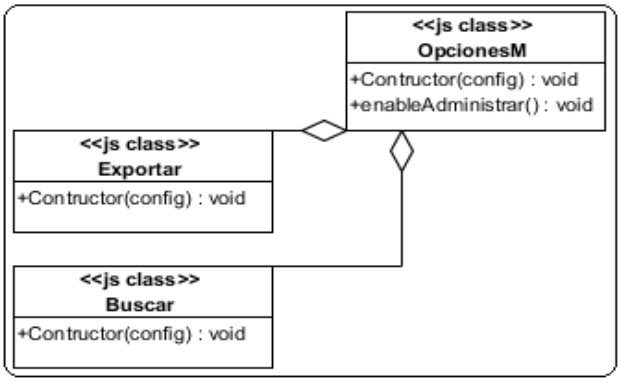


Fig 8: Aplicación del Patrón Experto en el desarrollo de la solución

**Alta Cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Este patrón se ve presente en las clases que conforman el modelo, como por ejemplo *Usuario.php*, *Registro.php*, *UsuarioRepository.php* y *RegistroRepository.php*.

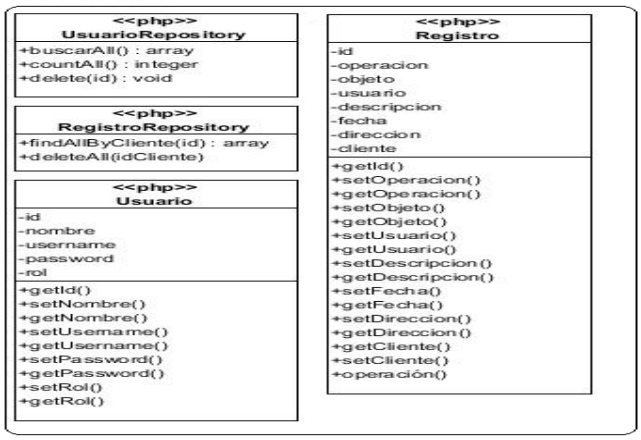


Fig 9: Aplicación del Patrón Alta Cohesión en el desarrollo de la solución

**Bajo acoplamiento:** Facilita la reutilización y una dependencia escasa, mejorando la programación y el diseño del sistema. En la solución se observa el uso de este patrón en las clases *Usuario.php* y *Registro.php*. Estas clases denominadas de acceso a datos son independientes de las de abstracción de datos (clases controladoras), lo que trae consigo mayor reutilización y flexibilidad del sistema.

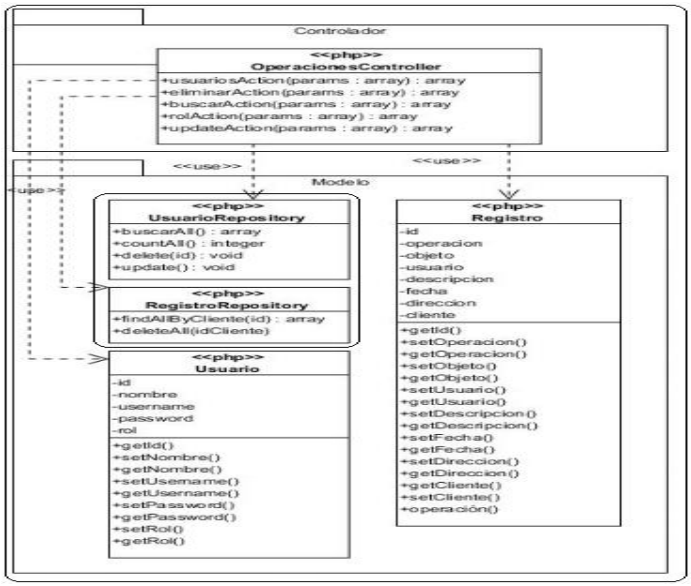


Fig 10: Aplicación del Patrón Bajo Acoplamiento en el desarrollo de la solución

### 2.4.5 Patrones GOF

Como se afirma en el Capítulo 1 (ver en el epígrafe 1.3.2) del presente documento, la utilización de patrones de diseño GOF resulta de vital importancia para la asignación de responsabilidades a objetos. A continuación se describen los patrones de diseño GOF aplicados en la solución.

**Fachada:** Este patrón brinda una interfaz sencilla que funcione de intermediaria entre un cliente o usuario y otra interfaz o grupo de estas. Este patrón se evidencia en la solución en la clase *OpcionesM*, la cual representa la interfaz principal del sistema genérico para la gestión de trazas.

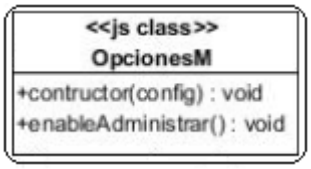


Fig 11: Aplicación del Patrón Fachada en el desarrollo de la solución

## 2.5 Modelo de Datos

El modelo de datos muestra dónde residen actualmente los objetos, la composición de cada uno y qué atributos describe el objeto, cuál es la relación entre los ellos y los procesos que los transforman. Esto es representado en el diagrama Entidad-Relación, el cual define todos los datos que se introducen, se almacenan, se transforman y se producen dentro de una aplicación. A continuación se muestra una representación del modelo de datos (17).

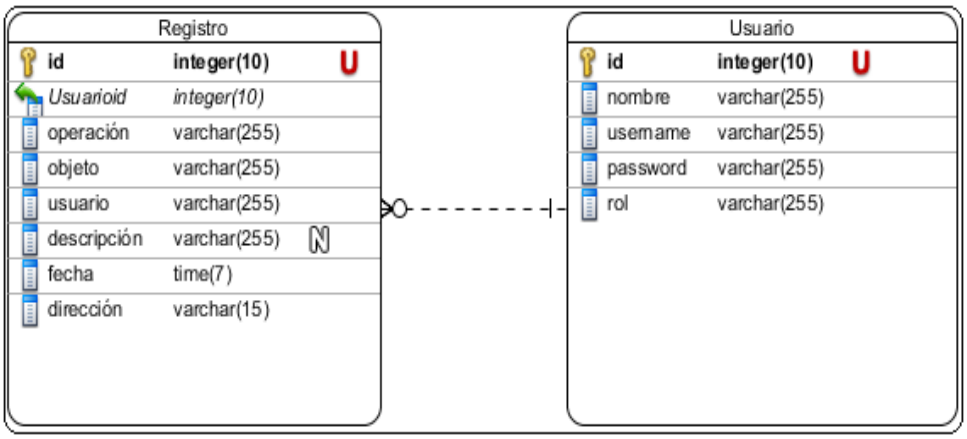


Fig 12: Modelo de Datos del Sistema de gestión de trazas.

**Descripción de las principales tablas del modelo de datos**

Tabla 2: Descripción de la tabla Registro

| **Nombre: Registro** | | |
|---|---|---|
| Descripción: Almacena las trazas de todos los usuarios registrados en el sistema. | | |
| **Atributo** | **Tipo** | **Descripción** |
| id | Integer(10) | Identificador auto-incremental. |
| Usuarioid | Integer(10) | Identificador auto-incremental (llave foránea que representa la relación uno-mucho entre las entidades usuario y registro). |
| operación | varchar(250) | Indica la acción realizada en la traza. |
| objeto | varchar(250) | Indica sobre qué objeto fue realizada la acción en la traza. |
| usuario | varchar(250) | Indica qué usuario realizó la acción en la traza. |
| descripción | varchar(250) | Brinda información sobre la traza para un mejor entendimiento. |
| fecha | time(7) | Indica cuándo se produce la traza. |
| dirección | varchar(15) | Indica la dirección ip donde se realizó la operación. |

Tabla 3: Descripción de la tabla Usuario

| **Nombre: Usuario** | | |
|---|---|---|
| Descripción: Almacena a los usuarios registrados en el sistema y al administrador. | | |
| **Atributo** | **Tipo** | **Descripción** |
| id | Integer(10) | Identificador auto-incremental. |
| nombre | varchar(250) | Nombre del usuario. |
| username | varchar(250) | Usuario por el cual se registrará. |
| password | varchar(250) | Contraseña de usuario. |
| rol | varchar(250) | Rol del usuario. |

## 2.6 Diagrama de despliegue

El modelo de despliegue se utiliza para modelar la disposición física de los artefactos de software en nodos. Detalla las capacidades de red, las especificaciones del servidor y otra información relacionada al despliegue del sistema propuesto.


Fig 13: Diagrama de despliegue del Sistema de gestión de trazas.

**Descripción del Diagrama de despliegue**

El diagrama representado anteriormente cuenta con tres nodos: PC Cliente, Servidor Web y Servidor de Base de Datos. En el nodo PC Cliente debe estar instalado un navegador web que soporte JavaScript, en el nodo Servidor Web debe estar instalado el servidor web y en el nodo Servidor de Base de Datos debe estar instalado el Sistema Gestor de Base de Datos (SGBD) PostgreeSQL.

Para la conexión entre los tres nodos se hace necesaria la utilización de protocolos de comunicación como HTTPS y TCP/IP. La conexión entre la PC Cliente y el Servidor Web se realizará mediante el protocolo HTTPS (por sus siglas en inglés *Hypertext Transfer Protocol Secure*) y la conexión entre el Servidor Web y el Servidor de Base de Datos será mediante el protocolo TCP/IP.

## Conclusiones del capítulo

Durante el desarrollo del presente capítulo se realizó una representación de las clases conceptuales del sistema a través del modelo de dominio, donde se identificaron los requisitos no funcionales y 11 requisitos funcionales contenidos en seis casos de usos para asegurar un buen funcionamiento del sistema. Para una representación de la estructura estática del sistema, se diseñó el diagrama de clases para cada caso de uso y los diagramas de secuencia por cada sección de estos. También se identificaron los patrones de diseño a utilizar en la aplicación y la distribución física del sistema a partir del diagrama de despliegue.

# CAPÍTULO 3 : "IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA GENÉRICO DE GESTIÓN DE TRAZAS "

## Introducción

El presente capítulo contiene los elementos fundamentales referentes a la etapa de implementación según la metodología de desarrollo de software utilizada. Se hace referencia a los principales estándares de codificación para la nomenclatura de las clases, así como a los artefactos generados durante esta fase. Se realiza además una validación de la solución mediante pruebas funcionales y de integración. Se explican cada una de las métricas utilizadas y se muestran los resultados de la implantación del componente.

## 3.1 Modelo de Implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Brinda la posibilidad de probar y desarrollar componentes como unidades, que finalmente serán integrados como un sistema ejecutable. Tiene como objetivo describir las relaciones existentes entre los diferentes componentes, basándose en las especificaciones de diseño (37).

### *3.1.1 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), se puede decir que es una unidad autónoma que implementa una o más clases que representan un contrato de servicios que el componente ofrece (38). A continuación se muestra el diagrama de componentes del CU Gestionar Traza, el cual está compuesto por los siguientes paquetes de implementación:

- El paquete de clases *Vista*: es el encargado de agrupar todos aquellos componentes que son utilizados en las interfaces y permiten la interacción directa con el usuario final del sistema.

- El paquete de clases *Controlador*: es el encargado de agrupar todos los componentes relacionados con la lógica del negocio, los cuales obtienen las peticiones del usuario y dan respuesta a las mismas.

- El paquete de clases *Modelo*: cuenta con un conjunto de componentes relacionados con las clases de acceso a datos, las cuales son utilizadas por el controlador para dar respuestas a las peticiones, ver Fig 14.

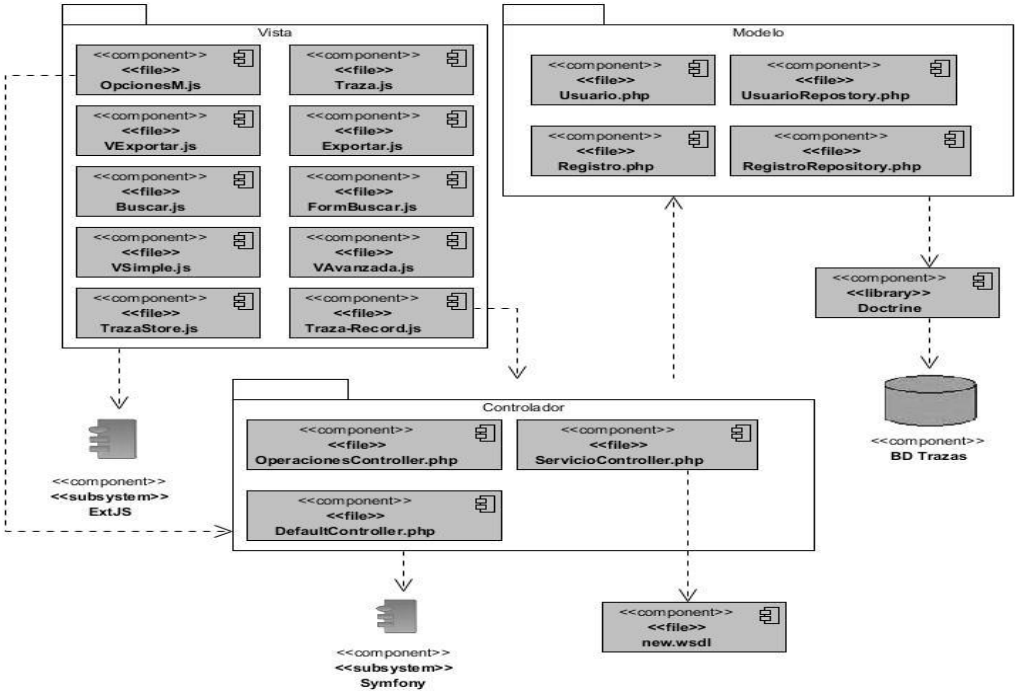


Fig 14: Diagrama de componentes del caso de uso Gestionar Traza

## 3.2 Código Fuente

El código fuente de un sistema informático es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para poder ejecutar las funcionalidades requeridas por el software. Un programador escribe en un lenguaje de programación específico, pero en este primer estado no es directamente ejecutable por la computadora. Para esto, el código debe ser traducido a lenguaje de máquina mediante un compilador.

### *3.2.1 Estándares de codificación*

Un estándar de codificación, es una regla que se sigue para crear un estilo cuando se genera un código fuente. Esta norma debe ser definida al comienzo de la implementación, así permitirá revisar, mantener y actualizar el código de una manera más sencilla y ordenada, principios que se deben cumplir en la escritura de un código.

**Estilo de codificación utilizado para la implementación de las clases PHP:**

- El nombre de una clase siempre empieza con mayúscula, en caso de estar compuesta por varias palabras se escribe una seguida de la otra y poniendo cada letra inicial en mayúscula (ejemplo: *OperacionController.php*).
- Se deben usar los comentarios para dar descripciones de código y facilitar información adicional que no es legible en el código mismo.
- En los archivos de código PHP se comienza por la etiqueta (<?php) pero no se utiliza el de cierre (?>) para evitar la inyección de espacios en blanco en la respuesta.
- Los nombres de funciones y variables deben empezar siempre con una letra minúscula. Cuando un nombre de función consiste en más de una palabra, la primera letra de cada nueva palabra debe estar en mayúsculas. Esto es llamado comúnmente como formato "camelCase".

**Estilo de codificación utilizado para la implementación de las clases JavaScript:**

- Se deben usar los comentarios para dar descripciones de código y facilitar información adicional que no es legible en el código mismo.
- Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas.

**Ejemplo de código fuente:**

A continuación la Fig 15 muestra el fragmento de código del método: “eliminar usuario”:

```
/**
 *
 * @remote
 * @param array $params
 * @return array
 */
public function eliminarAction($params) {
    try {
        $repositoryU = $this->getDoctrine()->getRepository('TrazasBundle:Usuario');
        $registroU = $repositoryU->delete($params['id']);
        return array(
            'success' => true,
            'id' => $params['id']
        );
    } catch (\Exception $exc) {
        return array(
            'success' => false,
            'error' => array(
                'message' => $exc->getMessage(),
                'code' => $exc->getCode()
            )
        );
    }
}
```

Fig 15 : Ejemplo de código fuente del Método "eliminar usuario"

## 3.3 Pruebas del software

Las pruebas son procesos de ejecución de un programa con la intención de descubrir errores cometidos al crear el diseño y la construcción. Se centran en los procesos lógicos del software, comprobando que todas las sentencias internas y los procesos externos funcionen como se espera. “La prueba no puede asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software” (39). Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

### *3.3.1 Niveles de prueba*

Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en el ciclo de vida del software. En el desarrollo de la fase de pruebas del Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones, se aplicarán las **pruebas de desarrollador** para comprobar los requisitos funcionales antes mencionados en la fase de diseño.

Las pruebas a nivel de desarrollo se refieren a pruebas ejecutadas sobre el código fuente por los desarrolladores. Usualmente esto se denomina “pruebas unitarias”. El Propósito es validar un elemento de implementación, es decir, una operación, clase o procedimiento almacenado, por medio de una o un conjunto de pruebas unitarias (40).

## Tipos de prueba

Para comprobar el correcto funcionamiento del Sistema genérico para la gestión de trazas se seleccionaron los siguientes tipos de pruebas:

- **Pruebas funcionales:** Son pruebas basadas en la ejecución para probar y validar que el software hace lo que debe y sobre todo, lo que se ha especificado.

- **Pruebas de rendimiento:** Se centra en determinar la velocidad con la que el sistema, bajo pruebas, realiza una tarea en las condiciones particulares del escenario de pruebas.

## Método de prueba

En la prueba de la **caja negra**, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Analiza principalmente la compatibilidad entre sí, en cuanto a las interfaces, de cada uno de los componentes del software.

### *3.3.2 Diseño de caso de prueba*

Los casos de pruebas se realizan con el objetivo de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo. Son un conjunto de condiciones o variables bajo las cuales el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. Cada caso de prueba debe definir el resultado de salida esperado que se comparará con el realmente obtenido. Al generar casos de prueba, se deben incluir tanto datos de entrada válidos y esperados como no válidos e inesperados. En la siguiente tabla se detallan las variables que se encuentran asociadas al caso de uso Gestionar traza:

Tabla 4: Tabla de descripción de variables asociadas al caso de uso Gestionar Traza

| No | Nombre del Campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Fecha Inicio | Campo lista desplegable | sí | Campo de tipo lista desplegable donde se especifica la fecha de inicio del rango de trazas que se desea buscar. |
| 2 | Fecha Fin | Campo lista desplegable | sí | Campo de tipo lista desplegable donde se especifica la fecha de fin del rango de trazas que se desea buscar. |
| 3 | Operación | Campo texto | sí | Campo de texto donde se especifica la operación que generó la traza. Admite la entrada de letras con longitud máxima de 30 caracteres. |
| 4 | Objeto | Campo texto | sí | Campo de texto donde se especifica el objeto sobre el cual se realizó la operación. Admite una cadena de longitud máxima de 255 caracteres. |

| | | | | |
|---|---|---|---|---|
| 5 | Usuario | Campo texto | sí | Campo de texto donde se especifica el usuario que realizó la operación. Permite una cadena de longitud máxima de 30 caracteres. |
| 6 | Descripción | Campo texto | sí | Campo de texto donde se describe información sobre la traza. Admite una cadena de longitud máxima de 255 caracteres. |
| 7 | Dirección ip | Campo texto | sí | Campo de texto en el cual se especifica la dirección IP donde se realizó la traza ejemplo: 254.254.254.254. Admite solo las entradas de direcciones IP. |
| 8 | Hora | Campo texto | sí | Campo de tipo lista desplegable donde se especifica la hora que se desea buscar. |
| 9 | Criterio | Campo texto | sí | Campo de texto donde se especifica un criterio a buscar. Admite una cadena de longitud máxima de 100 caracteres. |
| 10 | Exportar | Campo lista desplegable | sí | Campo de tipo lista desplegable donde se especifica qué conjunto de trazas se desea exportar. Se selecciona una de las siguientes opciones: Vista Actual o Reporte Completo. |

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se identificó el empleo de la técnica de **partición de equivalencia**.

## Matriz de datos

Consiste en un análisis de datos recibidos por las ejecuciones de funcionalidades del sistema donde se realiza con la entrada de distintas variables, sean estas válidas y no válidas para poder comprobar que el funcionamiento es el correcto y esperado. A continuación se muestra la matriz de datos para el escenario Buscar Traza de modo Avanzado del caso de uso Gestionar traza, el cual cuenta con cuatro aspectos: Escenario, Variables (Enumeradas según la descripción anterior), Respuesta del sistema y el Flujo Central.

Tabla 5: Tabla de matriz de datos para el escenario Buscar Traza

| **Escenario** | **Variables** | | | | | | | **Descripción** | **Respuesta del sistema** | **Flujo Central** |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | | | |
| EC 1.1 Buscar Trazas | **NA** | **NA** | **NA** | **NA** | **NA** | **NA** | **V** 10.8.5.1 | En este escenario se realiza la búsqueda de una traza en el sistema correctamente. | El sistema actualiza el listado de trazas según lo especificado. | 1. SGT/Gestionar traza/Búsqueda Avanzada (botón)<br>2. Se introduce el o los valores indicando el criterio para la búsqueda avanzada y se presiona la opción Aceptar.<br>3. Se cierra la ventana y se actualiza el listado de trazas según lo especificado. |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| EC 1.2 Buscar Trazas con datos incorrectos | **NA** | **NA** | **NA** | **NA** | **NA** | **NA** | **I** 10.0.54 | En este escenario se realiza la búsqueda de una traza en el sistema con datos incorrectos. | El sistema muestra el mensaje: Error en los campos. | 1. SGT/Gestionar traza/Búsqueda Avanzada (botón)<br>2. Se muestra la interfaz correspondiente a la funcionalidad y se llenan los campos con valores incorrectos.<br>3. Se presiona la opción Aceptar y se muestra una notificación indicando el error en el o los campos. |

### *3.3.3 Resultados de las pruebas de partición de equivalencia*

Después de realizar las pruebas de **caja negra** mediante los casos de prueba asociados a cada caso de uso, se evidenció la correcta validación de los campos y el correcto funcionamiento del sistema, comprobando que solo se acepten los caracteres válidos. Durante el desarrollo se realizaron tres iteraciones de pruebas. En la primera iteración fueron detectadas seis no conformidades y en la segunda se encontraron tres, a las que se le dieron seguimiento y fueron eliminadas a medida que se fue avanzando en el proceso de pruebas. Para la última iteración no se encontraron no conformidades, estos resultados se representan de forma gráfica en la Fig 16.

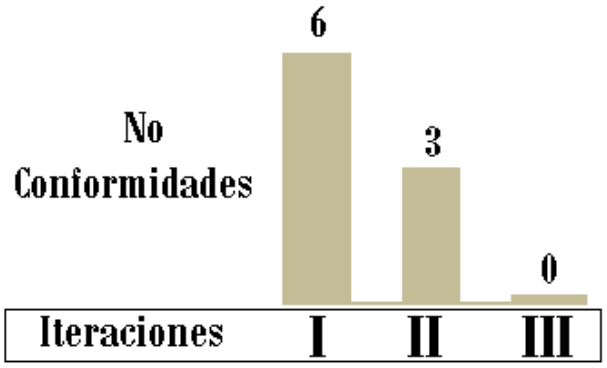


Fig 16: Resultado de las pruebas aplicadas a la solución

## Pruebas de rendimiento (Carga y Stress)

Estas pruebas se realizan para medir la respuesta de la aplicación a distintos volúmenes de carga esperados (cantidad de usuarios y/o peticiones). Ejemplo: velocidad de respuesta al procesar el ingreso de 30, 70 y 160 usuarios en forma simultánea. Se compara con el rendimiento esperado. Uno de sus objetivos es encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones.

### *3.3.4 Resultados de las pruebas de Carga y Stress*

Tabla 6 : Tabla de resultados de las pruebas de carga y estrés

| Cantidad de usuarios | Tiempo de respuesta | Valoración de los resultados obtenidos por la herramienta |
|---|---|---|
| 30 | 3 | Para la entrada de 30 usuarios se obtuvo un tiempo de respuesta de 3s. Se considera buen tiempo de respuesta ya que está por debajo de los 8s. |
| 70 | 6 | Para la entrada de 70 usuarios se obtuvo un tiempo de respuesta de 6s. Se considera buen tiempo de respuesta ya que está por debajo de los 8s. |
| 130 | 45 | Para la entrada de 130 usuarios se obtuvo un tiempo de respuesta de 45s. No se considera un buen tiempo de respuesta ya que está por encima de los 8s. |
| 160 | - | Para la entrada de 150 usuarios o más no se obtuvo respuestas. |

Esta prueba fue realizada sobre la base de datos con un total de 500 tuplas. Para el despliegue del sistema, se espera el almacenamiento de mas de 10 000 tuplas insertadas por los sistemas informáticos consumidores del servicio. El excesivo volumen de una base de datos implica en el tiempo de respuesta debido a la sobrecarga en el procesamiento de la consulta. Sin embargo, luego de realizar la prueba de carga y stress, se puede afirmar el correcto funcionamiento del sistema en cuanto a rendimiento, ya que a diferencia de 70 usuarios conectados de forma concurrente se espera la entrada de uno a tres usuarios, dado que normalmente una auditoría de trazas se realiza con poca frecuencia y por pocos auditores.

## Conclusiones del Capítulo

En el desarrollo del presente capítulo se realizó el modelo de implementación del Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones. Para entender la estructura del sistema se describieron los paquetes y las relaciones entre sí mediante el diagrama de componentes. Se especificaron los estándares de codificación a utilizar en los lenguajes de programación en la fase de implementación. Para comprobar el correcto funcionamiento y realización de los requisitos expuestos en la fase de diseño, se realizaron pruebas a nivel de programador utilizando el método de caja negra basado en la técnica de partición de equivalencia. También se midió el tiempo de respuesta y rendimiento del sistema a través de las pruebas de carga y estrés, automatizándolas con la herramienta JMeter 2.9. Durante el desarrollo del sistema se identificaron nueve no conformidades entre las dos primeras iteraciones, a las cuales se les dio un seguimiento y posteriormente fueron resueltas. Para una tercera iteración se obtuvo cero no conformidades, lo que demuestra la correcta implementación y diseño de pruebas del Sistema genérico para la gestión de trazas a través de un sistema web.

## Conclusiones generales

Con la realización del presente trabajo de diploma se ha cumplido con el objetivo general propuesto, así como con las tareas de la investigación definidas, arribándose a las siguientes conclusiones:

- El estudio de los sistemas que gestionan trazas a nivel internacional y nacional evidenció que estos integran en su solución la información proveniente del registro de las acciones realizadas por los usuarios en forma de *logs,* centrando su objetivo en la vinculación traza-seguridad.

- Se identificaron y analizaron las herramientas y tecnologías propuestas por el departamento Integración de Soluciones para el desarrollo de la solución, concluyendo que este conjunto de técnicas responden al entorno de desarrollo del proyecto, viabilizando las soluciones y el objeto de trabajo. A partir de ellas fueron generados todos los artefactos relacionados con los flujos definidos por la metodología de desarrollo de software OpenUp.

- Se implementaron las funcionalidades propuestas y descritas mediante los casos de uso y requisitos funcionales identificados en las fases de análisis y diseño.

- El diseño y realización de las pruebas de funcionalidad y rendimiento comprobó el correcto funcionamiento del Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones.

# RECOMENDACIONES

- Implementar un módulo en el Sistema genérico de gestión de trazas que permita visualizar los archivos log generados por el servidor web Apache.
- Extender el uso del Sistema genérico de gestión de trazas más allá del alcance del departamento Integración de Soluciones.

## REFERENCIAS BIBLIOGRÁFICAS


1- MSc. Oiner Gómez Baryolo. (2012) MODELO DE CONTROL DE ACCESO PARA SISTEMAS DE INFORMACIÓN EN ENTORNOS MULTIDOMINIOS, La Habana Cuba.

2- Conceptos relativos a trazabilidad. [online]. [Accessed 11 February 2014]. Available from: http://sede.aecoc.es/web/codificacion.nsf/0/925B46B62071AAB5C1256F2E00506B2E?OpenDocument.

3- Investigación Arqueometría y Analísis Arqueológico. [online]. [Accessed 11 February 2014]. Available from: http://pendientedemigracion.ucm.es/info/arqueoanalisis/Service%201.2.htm

4- ¿Qué son los sistemas de gestión? [online]. [Accessed 11 February 2014]. Available from: http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/

5- TraceListener (Clase) (System.Diagnostics). [online]. [Accessed 11 February 2014]. Available from: http://msdn.microsoft.com/es-es/library/system.diagnostics.tracelistener(v=vs.110).aspx

6- Bienvenida. [online]. [Accessed 11 February 2014]. Available from: https://srni2.uci.cu/bienvenida

7- Definicion de Servicio web - ¿qué es Servicio web? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/servicio%20web.php

8- Definicion de XML - ¿qué es XML? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/xml.php

9- Definicion de SOAP - ¿qué es SOAP? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/soap.php

10- Definicion de WSDL - ¿qué es WSDL? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/wsdl.php

11- Definicion de UDDI - ¿qué es UDDI? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/uddi.php

12- PRESSMAN,Roger S. 2002. Ingeniería del Software: Un enfoque práctico. Sexta edición. Madrid : Mc Graw-Hill, 2002. pág. 601.

13- Open Up. [online]. [Accessed 12 February 2014]. Available from: http://www.eclipse.org/epf/general/Open Up.pdf

14- Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) por Visual Paradigm International Ltd. - reporte y descarga. [online].

[Accessed 10 December 2013]. Available from: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/

15- Herramientas Case. [online]. [Accessed 12 February 2014]. Available from: http://www.slideshare.net/JaibooMurillo/herramientas-case-7495889

16- Visual Paradigm for UML 6.4 release notes - October 20, 2008. [online]. [Accessed 12 February 2014]. Available from: http://www.visual-paradigm.com/support/vpuml/releasenotes/640.jsp

17- Programación: Lenguajes de programacion. [online]. [Accessed 10 December 2013]. Available from: http://programingmaster1992.blogspot.com/p/lenguajes-de-programacion.html

18- PHP: ¿Qué es PHP? - Manual. [online]. [Accessed 12 February 2014]. Available from: http://www.php.net/manual/es/intro-whatis.php

19- Javascript - Home. [online]. [Accessed 12 February 2014]. Available from: http://javascriptdotnet.codeplex.com/

20- symfony.es, el mejor framework PHP para crear aplicaciones web. [online]. [Accessed 12 February 2014]. Available from: http://symfony.es/

21- ExtJS y Sencha Touch en español - Cursos, ejemplos, tutoriales, desarrollos y proyectos. [online]. [Accessed 12 February 2014]. Available from: http://www.extjs.mx/

22- Welcome to NetBeans. [online]. [Accessed 12 February 2014]. Available from: https://netbeans.org/

23- Definición de SGBD - ¿qué es SGBD? [online]. [Accessed 12 February 2014]. Available from: http://www.alegsa.com.ar/Dic/sgbd.php

24- www.postgresql.org.es. [online]. [Accessed 12 February 2014]. Available from: http://www.postgresql.org.es/

25- pgAdmin: PostgreSQL administration and management tools. [online]. [Accessed 12 February 2014]. Available from: http://www.pgadmin.org/

26- Welcome! - The Apache HTTP Server Project. [online]. [Accessed 13 February 2014]. Available from: http://httpd.apache.org/

27- Arquitecturas de software - Parte 1. [online]. [Accessed 13 February 2014]. Available from: http://www.slideshare.net/mstabare/arquitecturas-de-software-parte-1

28- Entorno cliente/servidor. [online]. [Accessed 13 February 2014]. Available from: http://es.kioskea.net/contents/148-entorno-cliente-servidor

29- Libro UML y Patrones – Introducción al análisis y diseño orientado a objetos. Tomo 1 por Craig Larman.

30- PRESSMAN,Roger S. 2005. Ingeniería del Software: Un enfoque práctico. Quinta edición. Madrid : Mc Graw-Hill, 2002. pág. 277. 84-481-3214-9.

31- Patrón de arquitectura Modelo Vista Controlador (MVC). [online]. [Accessed 13 February 2014]. Available from: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html

32- ¿Qué es un Patrón de Diseño? [online]. [Accessed 13 February 2014]. Available from: http://msdn.microsoft.com/es-es/library/bb972240.aspx

33- La Tecnología [online]. [Accessed 21 February 2014]. Available from: http://tecnologiashadai2012.blogspot.com/2012/08/tecnologia-es-el-conjunto-de_7.html

34- Modelo de Dominio | Tecnología y Synergix. [online]. [Accessed 24 February 2014]. Available from: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/

35- Arquitectura SOA [online]. [Accessed 29 February 2014]. Available from: www.soa.org

36- Arquitectura Orientada a Servicios SOA [online]. [Accessed 26 February 2014]. Available from:www.soa.iti.es

37- www.software-engin.com

38- Diagramas de components [online]. [Accessed 21 March 2014]. Available from: www.upedu.org/process/artifact/ar_impmd.htm

39- Pruebas del software [online]. [Accessed 21 March 2014]. Available from: http://indalog.ual.es/mtorres/LP/Prueba.pdf

40- Niveles de pruebas de software [online]. [Accessed 21 March 2014]. Available from: www.pruebasdelsoftware.wordpress.com

# BIBLIOGRAFÍA


1- MSc. Oiner Gómez Baryolo. (2012) MODELO DE CONTROL DE ACCESO PARA SISTEMAS DE INFORMACIÓN EN ENTORNOS MULTIDOMINIOS, La Habana Cuba.

2- Rolando Alfredo Hernández León y Sayda Coello González EL PROCESO DE INVESTIGACIÓN CIENTÍFICA. 2011.

3- Conceptos relativos a trazabilidad. [online]. [Accessed 5 February 2014]. Available from: http://sede.aecoc.es/web/codificacion.nsf/0/925B46B62071AAB5C1256F2E00506B2E?OpenDocument.

4- Implementación de un Sistema de Gestión de Trazabilidad [online]. [Accessed 5 February 2014]. Available from: http://www.gs1chile.org/productos_04_04.php.

5- Sistema de Gestión de Trazabilidad [online]. [Accessed 5 February 2014]. Available from: http://www.gs1pa.org/index.php/sistema-de-gestion-de-trazabilidad

6- Investigación Arqueometría y Analísis Arqueológico. [online]. [Accessed 11 February 2014]. Available from: http://pendientedemigracion.ucm.es/info/arqueoanalisis/Service%201.2.htm

7- ¿Qué son los sistemas de gestión? [online]. [Accessed 11 February 2014]. Available from: http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/

8- TraceListener (Clase) (System.Diagnostics). [online]. [Accessed 11 February 2014]. Available from: http://msdn.microsoft.com/es-es/library/system.diagnostics.tracelistener(v=vs.110).aspx

9- Bienvenida. [online]. [Accessed 11 February 2014]. Available from: https://srni2.uci.cu/bienvenida

10- Definicion de Servicio web - ¿qué es Servicio web? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/servicio%20web.php

11- Guía Breve de Servicios Web - W3C [online]. [Accessed 11 February 2014]. Available from: http://w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

12- Securing a web service by using a WS-Security policy – IBM [online]. [Accessed 11 February 2014]. Available from: http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.wlp.express.doc%2Fae%2Ftwlp_wssec_securing.html.

13- New Technologies Help You Make Your Web Services More Secure [online]. [Accessed 11 February 2014]. Available from: http://msdn.microsoft.com/en-us/magazine/cc164158.aspx

14- Editing WSDL security configurations [online]. [Accessed 11 February 2014]. Available from: http://pic.dhe.ibm.com/infocenter/rpthelp/v8r1m0/index.jsp?topic=/com.ibm.rational.ttt.common.doc/topics/tgsceditsec.html

15- Definicion de XML - ¿qué es XML? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/xml.php

16- Definicion de SOAP - ¿qué es SOAP? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/soap.php

17- Definicion de WSDL - ¿qué es WSDL? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/wsdl.php

18- Applying WS-Security | SOAP and WSDL – SoapUI [online]. [Accessed 11 February 2014]. Available from: www.soapui.org/...WSDL/applying-ws-security.html

19- Definicion de UDDI - ¿qué es UDDI? [online]. [Accessed 11 February 2014]. Available from: http://www.alegsa.com.ar/Dic/uddi.php

20- PRESSMAN,Roger S. 2002. Ingeniería del Software: Un enfoque práctico. Sexta edición. Madrid : Mc Graw-Hill, 2002. pág. 601.

21- Open Up. [online]. [Accessed 12 February 2014]. Available from: http://www.eclipse.org/epf/general/Open Up.pdf

22- Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME) (Visual Paradigm for UML (ME) por Visual Paradigm International Ltd. - reporte y descarga. [online]. [Accessed 10 December 2013]. Available from: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/

23- Herramientas Case. [online]. [Accessed 12 February 2014]. Available from: http://www.slideshare.net/JaibooMurillo/herramientas-case-7495889

24- Visual Paradigm for UML 6.4 release notes - October 20, 2008. [online]. [Accessed 12 February 2014]. Available from: http://www.visual-paradigm.com/support/vpuml/releasenotes/640.jsp

25- Programación: Lenguajes de programacion. [online]. [Accessed 10 December 2013]. Available from: http://programingmaster1992.blogspot.com/p/lenguajes-de-programacion.html
26- PHP: ¿Qué es PHP? - Manual. [online]. [Accessed 12 February 2014]. Available from: http://www.php.net/manual/es/intro-whatis.php
27- Javascript - Home. [online]. [Accessed 12 February 2014]. Available from: http://javascriptdotnet.codeplex.com/
28- symfony.es, el mejor framework PHP para crear aplicaciones web. [online]. [Accessed 12 February 2014]. Available from: http://symfony.es/
29- ExtJS y Sencha Touch en español - Cursos, ejemplos, tutoriales, desarrollos y proyectos. [online]. [Accessed 12 February 2014]. Available from: http://www.extjs.mx/
30- Welcome to NetBeans. [online]. [Accessed 12 February 2014]. Available from: https://netbeans.org/
31- Definición de SGBD - ¿qué es SGBD? [online]. [Accessed 12 February 2014]. Available from: http://www.alegsa.com.ar/Dic/sgbd.php
32- www.postgresql.org.es. [online]. [Accessed 12 February 2014]. Available from: http://www.postgresql.org.es/
33- pgAdmin: PostgreSQL administration and management tools. [online]. [Accessed 12 February 2014]. Available from: http://www.pgadmin.org/
34- Welcome! - The Apache HTTP Server Project. [online]. [Accessed 13 February 2014]. Available from: http://httpd.apache.org/
35- Arquitecturas de software - Parte 1. [online]. [Accessed 13 February 2014]. Available from: http://www.slideshare.net/mstabare/arquitecturas-de-software-parte-1
36- Entorno cliente/servidor. [online]. [Accessed 13 February 2014]. Available from: http://es.kioskea.net/contents/148-entorno-cliente-servidor
37- Libro UML y Patrones – Introducción al análisis y diseño orientado a objetos. Tomo 1 por Craig Larman.
38- PRESSMAN,Roger S. 2005. Ingeniería del Software: Un enfoque práctico. Quinta edición. Madrid : Mc Graw-Hill, 2002.
39- Patrón de arquitectura Modelo Vista Controlador (MVC). [online]. [Accessed 13 February 2014]. Available from: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html

40- ¿Qué es un Patrón de Diseño? [online]. [Accessed 13 February 2014]. Available from: http://msdn.microsoft.com/es-es/library/bb972240.aspx

41- La Tecnología [online]. [Accessed 21 February 2014]. Available from: http://tecnologiashadai2012.blogspot.com/2012/08/tecnologia-es-el-conjunto-de_7.html

42- Modelo de Dominio | Tecnología y Synergix. [online]. [Accessed 24 February 2014]. Available from: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/

43- Arquitectura SOA [online]. [Accessed 29 February 2014]. Available from: www.soa.org

44- Arquitectura Orientada a Servicios SOA [online]. [Accessed 26 February 2014]. Available from:www.soa.iti.es

45- www.software-engin.com

46- Diagramas de components [online]. [Accessed 21 March 2014]. Available from: www.upedu.org/process/artifact/ar_impmd.htm

47- Pruebas del software [online]. [Accessed 21 March 2014]. Available from: http://indalog.ual.es/mtorres/LP/Prueba.pdf

48- Niveles de pruebas de software [online]. [Accessed 21 March 2014]. Available from: www.pruebasdelsoftware.wordpress.com

49- Rest o Soap [online]. [Accessed 21 March 2014]. Available from: http://msdn.microsoft.com/es-es/magazine/dd942839.aspx#id0070002

50- Estandares de Servicios Web [online]. [Accessed 21 March 2014] Available from: http://www.consultorjava.com/wp/2012/04/24/estandares-para-la-implementacion-de-servicios-web/

51- Sommerville. 2004. Requerimientos del Software. [online]. [Accessed 23 March 2014] Available from: http://lsi.ugr.es/~ig1/docis/requeintro.pdf.

52- OASIS Web Services Security (WSS). [online]. [Accessed 23 March 2014] Available from: https://www.oasis-open.org/committees/wss/

53- Pressman, Roger. 2001. Ingenieía de Software: Un Enfoque Práctico. 2001.

54- XML. 2010. O´REILLY. xml.com. [online]. [Accessed 23 March 2014] Available from: http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58.

55- Huanca, Daniel. Herramientas de Prueba de Software. [online]. [Accessed 23 March 2014] Available from: http://herrorsoft.zxq.net/.

56- Blades, Steve,Nigel, Ramsay, Colin y Frederick, Shea. Learning Ext JS. 2008.

57- Frank W. Zammetti. Practical Ext JS Projects with Gears. 2009.
58- Javier Eguiluz. Desarrollo_agil_symfony 2.1. 2012.
59- Jorge Ramon. Ext JS 3.0 Cookbook. 2009.
60- Jesus Garcia. Manning.ExtJS.in.Action.2010.
61- Frank W. Zammetti. Practical Ext JS Projects with Gears. 2009.

# GLOSARIO DE TÉRMINOS

PDF - Formato de documento portátil (por sus siglas en inglés de *Portable Document Format*), es un formato de almacenamiento de documentos.

HTML - Lenguaje de marcado de hipertexto (por sus siglas en inglés de *HyperText Markup Language*), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto.

TICs - Tecnologías de la Información y la Comunicación.

ISO - Organización Internacional para la Estandarización.

TCP- (Por sus siglas en inglés de Transmission Control Protocol) dentro de una red de datos compuesta por computadoras, pueden usar TCP para crear conexiones entre sí a través de las cuales puede enviarse un flujo de datos.

UDP- User Datagram Protocol, es un protocolo del nivel de transporte basado en el intercambio de datagramas (encapsulado de capa 4 Modelo OSI). Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión.

WSDL - son las siglas de Web Services Description Language describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación

SOAP - (siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

DCOM - Distributed Component Object Model (DCOM), en español Modelo de Objetos de Componentes Distribuidos, es una tecnología propietaria de Microsoft para desarrollar componentes de software distribuidos sobre varios ordenadores y que se comunican entre sí.

HTTP - Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

HTTPS - Hypertext Transfer Protocol Secure (en español: Protocolo seguro de transferencia de hipertexto), es un protocolo de aplicación basado en el protocolo HTTP. Utiliza un cifrado basado en SSL/TLS para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP.

SMTP - Simple Mail Transfer Protocol (SMTP) (*Protocolo para la transferencia simple de correo electrónico*), es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos.

XML- siglas en inglés de *eXtensible Markup Language* ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes.

RPC - Remote Procedure Call (RPC) (del inglés, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

UDDI- son las siglas del catálogo de negocios de Internet denominado *Universal Description, Discovery and Integration*. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web.

TCP/IP - describe un conjunto de guías generales de diseño e implementación de protocolos de red específicos para permitir que un equipo pueda comunicarse en una red. TCP/IP provee conectividad de extremo a extremo especificando cómo los datos deberían ser formateados, direccionados, transmitidos y recibidos por el destinatario.

URL- Un localizador de recursos uniforme, más comúnmente denominado URL, es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet.

# ANEXOS

Universidad de las Ciencias Informáticas

***Centro de Tecnologías de Gestión de Datos***
***DATEC***

La Habana, 10 de Junio del 2014

"Año 56 de la Revolución".

## ACTA DE ACEPTACIÓN

**De una parte**, el Centro de Tecnologías de Gestión de Datos, en lo sucesivo DATEC, de la Universidad de las Ciencias Informáticas (UCI), representado en este acto por: Ing. Alberto Mendoza Gamache, Ing. Yoander Iñiguez Bermúdez, y de **otra parte** el estudiante: Leonardo Gonzalez Gonzalez.

**Primero**: Que en cumplimiento de los requisitos se ha efectuado el desarrollo del Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones.

**CONSIDERANDO**: Que los hitos realizados cumplen con los requisitos establecidos, han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.

**CONSIDERANDO**: Que el sistema desarrollado constituye un aporte relevante para el desarrollo del departamento Integración de Soluciones, permitiendo de forma genérica la gestión de trazas en los productos desarrollados por el departamento.

**POR TANTO: Las Partes** acuerdan formalizar mediante la presente Acta, la aceptación del producto: Sistema genérico de gestión de trazas para los productos desarrollados en el departamento Integración de Soluciones.

Y para que así conste, se extiende la presenta **Acta** en dos (2) ejemplares, rubricados por **Las Partes**.

Leonardo Gonzalez Gonzalez

Universidad de las Ciencias Informáticas

Jefe Dpto. Integración de Soluciones
Ing. Alberto Mendoza Gamache

Líder del Proyecto SIGDAT
Ing. Yoander Iñiguez Bermúdez

Reciben