UNIVERSIDAD INTERNACIONAL DE LAS AMERICAS
ESCUELA DE INGENIERÍA INFORMÁTICA

PRÁCTICA PROFESIONAL DIRIGIDA

Para optar por el grado de Bachillerato en
Ingeniería de Software

PROTOTIPO FUNCIONAL PARA LA VALIDACIÓN DE COMPONENTES DE SOFTWARE EN EL DEPARTAMENTO DE ASEGURAMIENTO DE CALIDAD DE LA DIVISIÓN DE SERVICIOS DE INGENIERÍA GLOBAL DE HEWLETT PACKARD

PABLO CURCÓ HERNÁNDEZ

AUTOR

ING. LEONARDO DELGADO ARROYO, MAP

TUTOR

MÁSTER FABIÁN RODRÍGUEZ SIBAJA

LECTOR

San José, Costa Rica

JULIO, 2017

TABLA DE CONTENIDOS

Tabla de Cuadros

Tabla de Figuras

Tabla de Gráficos

APROBACIÓN DEL TRIBUNAL EXAMINADOR

Esta Práctica Profesional Dirigida fue aprobada por el Tribunal Examinador de la Carrera de Ingeniería de Software de la UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS, como requisito para optar por el Grado de Bachillerato.

_____	_____
Máster Olda Bustillos Ortega	Ing. Leonardo Delgado Arroyo, MAP
Directora Escuela de Ingeniería Informática	Tutor

_____
Máster Fabián Rodríguez Sibaja
Lector

RESUMEN EJECUTIVO

Este proyecto consiste en la elaboración de un prototipo funcional para la validación de componentes de software, el cual consta de un sistema de análisis automatizado, que tenga la capacidad de analizar y validar componentes producidos por un equipo de ingenieros de software.

El apartado correspondiente a la introducción, incluye el análisis de la problemática que constituye el objeto de estudio sobre el que se fundamenta este trabajo, los estudios de viabilidad de la propuesta, los objetivos correspondientes - general y específicos -, así como el alcance del proyecto.

El primer capítulo corresponde al apartado de diagnóstico, en el cual se elabora dictamen de la situación interna y externa del proyecto, con el fin de conocer mejor su condición real y establecer las estrategias apropiadas para su desarrollo.

El segundo capítulo incluye el marco teórico sobre el que se sustenta el trabajo. Se han incluido tópicos afines a la temática del hardware y de los servidores, conceptos afines a los lenguajes de programación y al desarrollo de software orientado a objetos.

El tercer capítulo comprende el marco metodológico utilizado durante la producción del trabajo, el cuadro de variables correspondiente a cada objetivo planteado y el instrumento de recolección de datos utilizado junto al análisis de los resultados obtenidos.

Durante el cuarto capítulo se detalla el proceso de elaboración del software. Partiendo de la obtención de requerimientos de sistema, durante la etapa de análisis, pasando por la etapa de diseño, en la que se presenta la arquitectura del sistema, así como el diseño de módulos e interfaces, entre otros. El trabajo finaliza con una presentación de las conclusiones obtenidas durante la investigación y las recomendaciones pertinentes.

INTRODUCCIÓN

Tema

PROTOTIPO FUNCIONAL PARA LA VALIDACIÓN DE COMPONENTES DE SOFTWARE EN EL DEPARTAMENTO DE ASEGURAMIENTO DE CALIDAD DE LA DIVISIÓN DE SERVICIOS DE INGENIERÍA GLOBAL DE HEWLETT PACKARD

Planteamiento del problema de estudio

Durante el desarrollo de cada nueva versión de diagnósticos de fábrica, un equipo de ingenieros, parte de la división de Servicios de Ingeniería Global de Hewlett Packard Enterprise, procede a integrar los cambios requeridos, según lo establecido por los requerimientos definidos por los clientes y que han sido aprobados para ese ciclo de desarrollo determinado. Estos cambios pueden incluir soporte para nuevas órdenes de fabricación por pedido, actualizaciones o correcciones a órdenes ya existentes (que ya han sido fabricadas con anterioridad) o la eliminación de órdenes obsoletas.

En conformidad con el enfoque de fabricación por pedido, en el cual las órdenes no son fabricadas, sino hasta que hayan sido confirmadas por un cliente final y, debido además, a ciertas características particulares de esta operación, el ciclo de desarrollo tiene una duración corta, de una o dos semanas. Este periodo comprende tareas, tales como el análisis de requisitos, la integración de los cambios necesarios en los diferentes archivos que componen los diagnósticos y la validación y verificación de los cambios efectuados. Cada nuevo componente es entregado a fábrica y las nuevas órdenes son procesadas utilizando estos insumos.

Evidentemente, como en todo ciclo de producción de software, existe la posibilidad de que se generen, tanto desavenencias relacionadas con el proceso de especificación de

requerimientos, como errores de sintaxis o de estructura de los archivos modificados por los ingenieros.

Por supuesto, se han establecido procedimientos para garantizar que las modificaciones realizadas a los componentes, correspondan a los nuevos requerimientos establecidos para ese ciclo de desarrollo, y para verificar y validar que las nuevas versiones de componentes estén libres de errores. Estos procedimientos de validación se llevarán a cabo, tanto durante la fase de desarrollo, como durante la fase de validación final de los componentes.

Los equipos de cómputo, tales como los servidores, son productos complejos. Están constituidos de una amplia gama de periféricos y tienen un gran número de características personalizables: distintas versiones de BIOS, diferentes versiones de firmware, los discos duros pueden configurarse en arreglos de diferentes niveles, y muchas otras opciones más. El equipo de trabajo sigue protocolos específicos y utiliza herramientas para asegurar la calidad de los componentes entregados, pero aun así, se producen errores que implican costos innecesarios y atrasos en los cronogramas de trabajo.

Las restricciones de tiempo que implica el modelo de fabricación por pedido, la naturaleza, ya de por sí compleja de la manufactura de servidores, y la complejidad estructural y sintáctica de los componentes dificulta la tarea de validar los cambios efectuados durante el ciclo de desarrollo. La información pertinente a una sola orden puede estar dispersa en uno o varios de los archivos que constituyen el componente, dependiendo de las características específicas de cada orden, y cada componente de diagnóstico puede estar conformado por cientos de archivos en distintos formatos, lo que dificulta comprender

las características de cada orden particular y, prácticamente, imposibilita una visión global y clara de las características pertinentes a todas las órdenes incluidas en un diagnóstico.

Todos estos factores ocasionan una serie de problemáticas que afectan, tanto a desarrolladores, como a los clientes que utilizan estos componentes, de las cuales se destacan:

- Errores encontrados durante el procesamiento de órdenes que no fueron encontrados durante la fase de validación, producto de un inadecuado manejo de requerimientos, incorrectamente interpretados u omitidos. Este problema afecta directamente las órdenes que están siendo procesadas en fábrica, y obliga a los desarrolladores a crear parches de emergencia o a incluir nuevos requisitos que serán introducidos en un nuevo lanzamiento del componente.
- No existe información oportuna y precisa de la cantidad y características de las órdenes soportadas por cada componente de software. Ante cualquier duda, la única forma de conocer qué ocurre exactamente al procesar una orden es analizar cada archivo del componente, lo cual podría hacerlo solamente quien comprenda la estructura y funcionalidad de los componentes.
- Aunque existe la posibilidad de hacer un análisis sintáctico de los scripts, o la validación de la estructura de archivos en formatos, tales como XML, con lo cual disminuyen las posibilidades de errores en tiempo de ejecución durante las pruebas, la forma en como la información está dispersa en los componentes dificulta la validación de estos. Con información en un formato más apropiado podrían efectuarse análisis estadísticos y comparaciones entre distintas versiones de un

mismo componente, con la finalidad de controlar de mejor forma los cambios efectuados de una versión a otra.

Si se acordara hacer un esfuerzo por crear un documento o base de datos con las características de todas las órdenes contenidas en un componente, una vez que este componente sea alterado, lo cual podría ocurrir hasta una vez cada semana, la información devendría obsoleta, a menos que cada ingeniero involucrado se encargara de actualizar la información de forma precisa cada vez que haga un cambio, lo cual ya ha demostrado ser ineficiente: el ciclo de desarrollo es muy corto y es usual que haya cambios de requisitos por parte de los clientes, lo cual hace que el desarrollo tenga una dinámica muy particular y complica el mantenimiento de documentación de este tipo.

Justificación

Se ha considerado la posibilidad de crear un sistema de análisis automatizado, basado en software, que tenga la capacidad de analizar los componentes, los cuales tienen una estructura bien definida y utilizan formatos conocidos. Este sistema podría, en segundos, extraer la información pertinente y modelarla de forma tal que facilite su interpretación. Cada versión de un componente puede modelarse como un inventario de órdenes y sus características.

El sistema de validación propuesto vendría a solventar varias de las problemáticas discutidas con anterioridad. Para empezar, el sistema estaría en capacidad de hacer un análisis de todos los archivos que conforman un componente de diagnóstico, lo que implica una revisión de la estructura y sintaxis de estos documentos, lo cual en principio ayudaría a disminuir los errores de este tipo. Por otro lado, los ingenieros de diagnósticos tendrían acceso a información estadística valiosa, generada automáticamente, lo que haría que esta

información directamente extraída de cada componente sea más precisa que si fuera generada a mano, y que podría ser utilizada en el quehacer diario del equipo. Tal información no existe en este momento.

Además, las consultas por parte de terceros con respecto a las características de una u otra orden son constantes. Actualmente, existen dos fuentes de información disponibles:

- Los documentos de intención de cliente, provistos por la fábrica con una definición escrita de las características de una orden,
- Las solicitudes de cambios, las cuales son creadas para solicitar modificaciones determinadas a los desarrolladores cuando es requerido.

Estas fuentes puede que no reflejen el estado real de los componentes en un momento determinado y no son fáciles de interpretar.

Es por esto que se ha considerado, también, el crear un repositorio de datos donde esta información pueda almacenarse, con el fin de poder contar con una referencia histórica de cada una de las versiones de cada componente enviado a los clientes, información que puede ser utilizada para comparar dos versiones diferentes consecutivas del mismo producto. Esto permitiría tener un mayor control sobre cada uno de los cambios realizados durante el ciclo de desarrollo.

Viabilidad técnica.

El lenguaje de programación que se ha escogido para el desarrollo del prototipo es Java, el cual es un lenguaje de alto nivel de propósito general, orientado a objetos: "El principal atractivo de Java es que se trata de un lenguaje seguro y portable que soporta las construcciones de la programación moderna orientada a objetos" (Allen, 2000, p. 24)

El entorno de desarrollo integrado, o sea, la aplicación que proporciona servicios integrales para facilitar el desarrollo de software que se ha escogido para el desarrollo de este proyecto es NetBeans, el cual requiere el entorno de desarrollo integrado de Java o "Java SE Development Kit" versión 7 actualización 10 o cualquier otro más reciente. Este software está disponible en línea y ha sido utilizado con anterioridad por el grupo. La licencia del software es libre, lo cual se explica con más detalle adelante, en la sección viabilidad legal.

A continuación, se presentan las recomendaciones mínimas y recomendadas de hardware estipuladas en las notas de lanzamiento de NetBeans 8.2 (Oracle Corporation, 2016)

Requerimientos mínimos.

- Microsoft Windows Vista SP1/Windows 7 Professional
- Procesador: 800MHz Intel Pentium III o equivalente
- Memoria: 512 MB
- Espacio en disco: 750 MB de espacio libre en disco

Requerimientos recomendados.

- Microsoft Windows 7 Professional/Windows 8/Windows 8.1
- Procesador: Intel Core i5 o equivalente
- Memoria: 2 GB (32-bit), 4 GB (64-bit)
- Espacio en disco: 1.5 GB de espacio libre en disco.

Las computadoras con que dispone todo el equipo de desarrollo cuentan con las siguientes características:

- Windows 7 Enterprise (Service Pack 1)
- 8 GB de memoria RAM
- Procesadores Core i5-3320M @2.60Ghz o más recientes

Estas cumplen a cabalidad con las recomendaciones mínimas y recomendadas de hardware estipuladas en las notas de versión de NetBeans 8.1

El departamento cuenta con la infraestructura necesaria para alojar, tanto el prototipo como la base de datos una vez que la aplicación sea implementada. La infraestructura cuenta con varios servidores industriales, con altas capacidades de almacenamiento los cuales son utilizados actualmente para alojar los diferentes productos de software que elaboran los distintos equipos de desarrolladores de la división.

Las especificaciones recomendadas de un servidor utilizado con el fin de alojar, tanto la aplicación como la base de datos podría consistir en un ProLiant DL380Gen9 con un procesador Intel E5-2650 v3 (Deca-core, 2.30 Ghz), con 16 GB de memoria DDR4-2133. En cuanto al almacenamiento interno, estos servidores tienen una capacidad máxima que comprende desde los 52 TB hasta los 150 TB, los cuales sobrepasan cualquier estimación considerando los alcances de este prototipo.

Dado lo anterior, se concluye que el prototipo tiene viabilidad técnica.

Viabilidad operativa.

Se ha considerado la estructura del equipo que utilizará esta aplicación, así como los procedimientos concernientes al desarrollo y validación de los componentes, y no se encontraron obstáculos operativos. Una vez finalizado el desarrollo de la aplicación, se tendrían que introducir o actualizar ciertos procedimientos para incluir el uso de esta

herramienta oficialmente y contribuir a los objetivos establecidos, lo cual tampoco representa ningún obstáculo operativo.

El equipo está conformado exclusivamente de ingenieros de software e ingenieros en informática con amplio dominio en las herramientas relacionadas con el proyecto. El equipo utiliza un número de herramientas de distintas características y complejidad, por lo que no se requiere ningún entrenamiento o capacitación. La estructura organizativa del área involucrada tampoco requiere modificación alguna.

Considerando todo lo anterior, el prototipo tiene la viabilidad operativa para ser desarrollado.

Viabilidad económica.

Según el Ministerio de Trabajo y Seguridad Social el salario mínimo de un bachiller universitario es de 524,477.85 colones por mes, dividido entre 4 semanas y 48 horas deja un total de 2,731.65 colones por hora. Se ha establecido un plazo de 4 meses, trabajando 400 horas en total para el desarrollo de la aplicación. Este plazo comprende todas las actividades de especificación, desarrollo y validación del software por lo que el costo total sería de 1, 092,660 colones. No existen gastos relacionados a equipo de hardware o licencias. La empresa considera oportuna la inversión.

Cuadro 1: Detalle de costos estimados

Concepto	**Cantidad/horas**	**Costo Unitario/colon**	**Costo Total/colones**
Especificación de Software	100	₡2.731,65	₡273.165,00
Diseño e implementación de Software	150	₡2.731,65	₡409.747,50
Validación de Software	150	₡2.731,65	₡409.747,50
Total	400	₡2.731,65	₡1.092.660,00

Fuente: Propia

No se requiere ninguna otra inversión, ni en software, redes ni telecomunicaciones, dado que la empresa cuenta con todos los elementos requeridos. Dado lo anterior, se considera que el prototipo tiene la viabilidad económica para ser desarrollado.

Viabilidad legal.

En noviembre 13 del 2006, Sun Microsystems puso a disposición sus implementaciones de Java bajo la licencia Pública General de GNU, la cual garantiza a sus usuarios la libertad de usar, estudiar, compartir y modificar el software al declarar que el software es libre y protegerlo de intentos de apropiación que restrinjan esa libertad. Esta licencia fue creada originalmente por Richard Stallman fundador de la Free Software Foundation (FSF) para el proyecto GNU.

La aplicación cumple con la ley de protección de la persona Frente al Tratamiento de sus Datos Personales (Ley No 8968) ya que no existe ningún dato personal involucrado en la programación de esta aplicación.

La ley de Procedimientos de Observancia de los Derechos de Propiedad Intelectual (Ley No 8039) es respetada, ya que todas las herramientas por utilizar son gratuitas o están propiamente licenciadas por la empresa.

Ley de Derechos de Autor y Derechos Conexos (Ley No 6683), también, será observada, ya que no se utilizará código propiedad de terceros y se mencionará en las respectivas citas bibliográficas y referencias cualquier texto de terceros utilizado en esta obra. Además, el prototipo de software producido no será distribuido sin consentimiento del desarrollador y será utilizado solamente para los fines por los cuales será desarrollado.

Dado lo anterior, el prototipo cumple con la viabilidad legal requerida.

Objetivos de la Investigación

Objetivo general.

Desarrollar un prototipo funcional para la validación de componentes de software en el departamento de Aseguramiento de Calidad de la división de servicios de ingeniería global de Hewlett Packard.

Objetivos específicos.

1. Obtener, analizar, especificar y validar los requerimientos del sistema necesarios para resolver los objetivos establecidos.
2. Diseñar la arquitectura, los componentes, la interfaz gráfica y las estructuras de datos de acuerdo con la especificación del sistema.
3. Desarrollar el prototipo funcional en conformidad con la arquitectura y las especificaciones establecidas.
4. Verificar el prototipo con el fin de determinar que éste cumple con las especificaciones establecidas.

Alcances

Este proyecto tiene como objetivo la creación de un prototipo funcional que permita la extracción de los datos contenidos en varios componentes de software, cuya complejidad estructural dificulta su análisis, con el fin de que estos datos sean transformados en una estructura comprensible y sean presentados a los ingenieros de forma tal que facilite la validación y comprensión del contenido de estos.

También, se contemplan las tareas concernientes a la especificación de software, el cual consiste en comprender, definir, documentar y validar los requerimientos del prototipo de acuerdo con las necesidades del equipo de ingenieros. Una vez obtenidos y analizados

los requerimientos estos serán especificados en un documento y validados. Se considera, además la etapa de diseño del prototipo, en la que se procederá a hacer una descripción de la arquitectura del software y los principales componentes que lo conformarán.

Además, se determinarán las estructuras de datos relevantes o los algoritmos que vayan a ser requeridos, el diseño de la interfaz gráfica de la aplicación junto a las tareas concernientes a la validación del software, con el fin de garantizar que el sistema cumpla con las especificaciones establecidas.

Al considerarse de un prototipo funcional, no se contempla ni la etapa de implementación final del software, ni el entrenamiento al personal relevante, ni la documentación de las guías de usuario.

Alcance funcional.

A continuación, se describen los módulos que constituirán el prototipo propuesto:

Módulo de validación.

Será utilizado para asegurar que la estructura de los datos primarios esté libre de errores. Este módulo debe ser capaz de analizar la sintaxis de archivos en distintos formatos, los cuales incluyen archivos de texto y archivos en formato XML. Debe verificar qué scripts en lenguajes de alto nivel como Perl estén libres de errores de compilación. El usuario utilizará un control para apuntar al directorio que contiene componentes y usará un algoritmo recursivo para validar la estructura de los archivos contenidos. Esta primera verificación (validación sintáctica) pretende garantizar la adecuada estructura de los archivos que serán comprobados durante la etapa de análisis de componente, donde se ejecutará una validación más profunda.

Módulo de análisis.

Permite al usuario seleccionar la ruta de un componente para su análisis. La aplicación procederá a analizar la estructura de los componentes y extraer los datos requeridos (usando árboles de decisión diseñados específicamente para cada caso y buscando la presencia de valores atípicos o ausencia de datos o valores nulos), utilizará una estructura de datos adecuadas para almacenar estos datos dispersos para proceder, entonces, con un análisis estadístico, una apropiada clasificación y reestructuración de datos en un formato comprensible y adecuado para su análisis.

Este módulo permite visualizar la bitácora de todo el proceso, la bitácora de errores y permite, además, exportar los datos en distintos formatos, así como su almacenamiento en el almacén de datos de la aplicación. Contiene opciones para guardar los resultados en tres formatos diferentes: texto, HTML o XML. Además, dispondrá de opciones para filtrar los resultados obtenidos usando diferentes criterios.

Módulo de exploración.

Permite la exploración de los datos obtenidos por el módulo de análisis de componentes con el fin de interpretar y evaluar los datos obtenidos. La información debe desplegarse en una interface gráfica que represente de forma clara la información obtenida incluyendo resúmenes y gráficos de ser necesario.

Debe poseer herramientas para filtrar la información usando distintos criterios. Esta información será representada como un conjunto de entidades relevantes, que tienen propiedades o atributos relacionados a esta entidad, y podrá ser extraída en diferentes formatos, según sea requerido (texto, HTML o XML).

Módulo de comparación.

Este módulo genera, a partir de la información extraída, un modelo que permita analizar dos versiones distintas del mismo componente. La aplicación se encargará de comparar los atributos de las entidades contenidas en ambas versiones e identificar incongruencias. Además, debe representar ambas entidades de forma gráfica. De esta forma, no solo será posible verificar la integridad de los componentes en cualquier punto del ciclo de desarrollo, sino que se podrá controlar qué cambios se han realizado en cada uno de ellos. Se podrá mantener un archivo histórico con la información de todos los componentes disponibles, facilitando diversas tareas, tanto de los desarrolladores, como de los agentes de calidad del equipo.

Módulo de consultas.

Permite la exploración de la biblioteca de componentes construida a partir de los resultados obtenidos al analizar las diferentes versiones de componentes. Las librerías construidas serán categorizadas de acuerdo con la plataforma a la que pertenecen, y el usuario podrá elegir cada librería para explorarla o validarla contra otra librería.

Módulo de seguridad.

Incluye opciones para crear y administrar usuarios de forma que solo usuarios debidamente identificados puedan utilizar la aplicación.

Módulo de reportes.

Será utilizado para generar e imprimir reportes relevantes, tales como bitácoras de proceso, bitácoras de errores y resultados, en los formatos que se consideren oportunos.

Módulo de mantenimiento.

Permite configurar variables globales que faculta ajustar el comportamiento de la aplicación, la selección de la carpeta de trabajo y la ruta de resultados predeterminadas.

Alcance metodológico.

Este proyecto incluye actividades de proceso de software con el fin de especificar, diseñar y probar el prototipo propuesto. Las actividades básicas de proceso consisten en: proceso de especificación de software, diseño de software y validación de software. Durante el desarrollo del prototipo se elaborarán distintos tipos de documentos los cuales contendrán la información que se produce durante cada etapa del proyecto.

Especificación de software.

La especificación de software, o la ingeniería de requerimientos, consiste en el proceso de comprender y definir qué servicios se requieren del sistema. Se enfoca en producir un documento de requerimientos convenido que especifique los requerimientos de los interesados. Como afirma Sommerville "Consiste en la actividad de transcribir la información recopilada durante la actividad de análisis, en un documento que define un conjunto de requerimientos." (Sommerville, 2011, p. 38)

Entre las actividades contempladas durante la fase de especificación de software se contempla un estudio de factibilidad, el cual debe ser rápido y barato y debe servir para estimar si las necesidades pueden cubrirse con las tecnologías disponibles, o si el sistema propuesto tiene un costo-beneficio para el equipo (Sommerville, 2011).

Otra actividad consiste en la obtención y análisis de requerimientos. Este es el proceso de derivar los requerimientos del sistema mediante discusiones con los usuarios, análisis de procedimientos y otras herramientas que permitan entender a fondo la necesidad del equipo.

Una vez obtenidos los requerimientos se procede a una especificación formal de estos, actividad llamada especificación de requerimientos y que consiste en la tarea de transcribir la información recopilada durante el análisis. Esta etapa finaliza con la validación de requerimientos, tarea que consiste en verificar que los requerimientos sean realistas, coherentes y completos (Sommerville, 2011).

Diseño del software.

El diseño del software se refiere al proceso de convertir una especificación del sistema en un sistema ejecutable. Incluye procesos de diseño y programación de software, aunque, también, puede involucrar la corrección de la especificación de software, si se utiliza un enfoque incremental de desarrollo (Sommerville, 2011).

Un diseño de software es una descripción de la estructura del software que se va a implementar, los modelos y las estructuras de datos utilizados por el sistema, las interfaces entre componentes del sistema y, en ocasiones, los algoritmos usados. Con frecuencia, los desarrolladores no llegan inmediatamente a una creación terminada, sino que, conforme avanza el proyecto, desarrollan el diseño de manera iterativa, agregan detalles o corrigen diseños anteriores (Sommerville, 2011, p. 38).

Existen cuatro actividades en el proceso de diseño, la primera consiste en el diseño arquitectónico: aquí se identifica la estructura global del sistema, los principales componentes (llamados en ocasiones subsistemas o módulos), sus relaciones y cómo se distribuyen.

Otra actividad es el diseño de interfaz, que consiste en la definición de las interfaces entre los componentes de sistemas. Con una interfaz precisa es factible usar un componente sin que otros tengan que saber cómo se implementó. Una vez que se acuerdan las

especificaciones de interfaz, los componentes se diseñan y se desarrollan de manera concurrente.

La tercera actividad del proceso de diseño consiste en el diseño de componentes: en él se toma cada componente del sistema y se diseña cómo funcionará. Esto puede ser un simple dato de la funcionalidad que se espera implementar o, alternativamente, una lista de cambios por realizar sobre un componente que se reutiliza o sobre un modelo de diseño detallado. La actividad final de la etapa de diseño consiste en el diseño de base de datos, donde se diseñan las estructuras del sistema de datos y cómo se representarán en una base de datos (Sommerville, 2011).

Validación de software.

La validación de software consiste en la demostración de que un sistema cumple, tanto con las especificaciones establecidas como con las expectativas del cliente.

"Dos propósitos guían el aseguramiento de la calidad. El primero es que el usuario del sistema de información es el factor individual más importante en establecer y evaluar su calidad. El segundo es que es mucho menos costoso corregir los problemas en sus fases iniciales…" (Kendall & Kendall, 2005, p. 581).

Las pruebas del programa, donde el sistema se ejecuta a través de datos de prueba simulados, son la principal técnica de validación. Las etapas en el proceso de prueba son las pruebas de desarrollo, en el cual las personas que desarrollan el software ponen a prueba los componentes que constituyen el sistema. Cada componente se prueba de manera independiente (Sommerville, 2011).

A continuación, se efectúan las pruebas del sistema donde los componentes del sistema se integran para crear un sistema completo. Este proceso tiene la finalidad de

descubrir errores que resulten de interacciones no anticipadas entre componentes y problemas de interfaz de componente, así como mostrar que el sistema cubre sus requerimientos funcionales y no funcionales (Sommerville, 2011).

La etapa de pruebas de aceptación es la etapa final en el proceso de pruebas, antes de que el sistema se acepte para uso operacional. El sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados. Las pruebas de aceptación revelan los errores y las omisiones en la definición de requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba (Sommerville, 2011).

Alcance tecnológico.

El objetivo de este proyecto consiste en el diseño de un prototipo funcional correspondiente a una aplicación gráfica de escritorio basada en Java, la cual deberá ser utilizada en cualquier computadora que tenga Microsoft Windows Vista SP1 o cualquier sistema operativo más reciente, un procesador Intel Pentium III o alguno más reciente, al menos 512 MB de memoria y 750 MB de espacio en disco. Se recomienda que el sistema operativo sea Windows 7 Profesional, o Windows 8/8.1, un procesador i5 o equivalente, memoria RAM de 2GB a 4GB y 1.5GB de espacio libre en disco. La computadora requiere del entorno de ejecución Java o JRE.

La aplicación va a requerir acceso a una base de datos externa, pero con el fin de que los ingenieros puedan utilizar la aplicación en escenarios donde el acceso a la red no sea posible, algunas funcionalidades esenciales estarán disponibles sin acceso a ésta. El sistema gestor de base de datos que utilizará el prototipo consiste en MySQL Community Server

versión 5.7, la cual corresponde a la versión de código abierto y licencia pública general de MySQL Server.

La funcionalidad principal del prototipo consiste en la lectura y creación de archivos que utilizan el formato XML. XML, siglas en inglés que significan “Lenguaje de Marcado Extensible” es un lenguaje que permite definir lenguajes de marcas o etiquetas. El formato es generalmente utilizado para almacenar estructuras de datos arbitrarias (su estructura es libre) en forma humanamente legible

Limitaciones

No existen limitaciones significativas para el desarrollo de este trabajo.

Entre las limitaciones no significativas identificadas hasta el momento, se encuentran las siguientes:

- Como ocurre con cualquier otro lenguaje de programación, Java tiene ciertas desventajas. Ya que este corresponde a un lenguaje interpretado, puede darse una diferencia de velocidad desfavorable con respecto a otros lenguajes de programación. Este aspecto ha sido considerado a la hora de programar las rutinas que implican mayor cantidad de cálculos, de forma que el rendimiento no sea disminuido por este aspecto. Además, con el hardware y software recomendado, se espera que este sea lo suficientemente rápido incluso para las rutinas más complejas.
- Las rutinas encargadas de analizar archivos de texto son susceptibles a un número de excepciones relacionadas, tanto a la lectura (y permisos de acceso) de los archivos que están siendo analizados, como a errores en el

formato XML de éstos. Por lo cual, estas excepciones deben ser manejadas minuciosamente para evitar errores en la información recolectada.

Antecedentes

Un equipo de Ingenieros de Software y Sistemas, el cual es parte de la división de Servicios de Ingeniería Global de Hewlett Packard Enterprise, se encarga del desarrollo y la validación de componentes de software, también llamados diagnósticos, utilizados por un sistema de automatización durante la fabricación de dichos servidores.

Estos componentes son desarrollados usando un modelo de desarrollo iterativo y creciente, por lo cual, durante cada ciclo de desarrollo, el equipo incluye nuevos requerimientos en cada nueva versión de un componente, con la finalidad de que estos puedan ser utilizados para procesar nuevos tipos de órdenes.

Dada la naturaleza de los productos fabricados y las tareas llevadas a cabo durante su manufactura, estos componentes de software son de una complejidad importante. Consisten, por lo general, en scripts, o archivos de procesamiento por lotes, los cuales contienen instrucciones que definen las distintas acciones que se ejecutarán en el sistema durante su fabricación, y archivos XML que incluyen información pertinente a esas acciones o actividades.

Durante el proceso de fabricación de cada orden de servidores, cada unidad es ensamblada y conectada a una red que le permite el acceso a los diagnósticos y a los recursos necesarios durante esta etapa de su manufactura.

Cada orden elaborada en fábrica es identificada por un código y será procesada, según la información contenida en los componentes de diagnóstico. Muchos de estos procesos

varían dependiendo del modelo del producto que se está fabricando o de los componentes que lo conforman.

El documento de intención de cliente, es el documento oficial que describe los componentes de hardware, software y procesos que determinan cada orden. Este puede indicar no solo el hardware y las versiones de firmware que una orden debe incluir, sino además, la configuración de BIOS y de otros componentes de hardware personalizable, tales como la posición en la debe ser conectado un componente PCI, sistema operativo que debe ser instalado en el sistema y otros detalles.

Como se mencionó con anterioridad, tal y como ocurre en todo ciclo de producción de software, existe la posibilidad de que se generen, tanto discrepancias relacionadas con el proceso de especificación de requerimientos, como errores de sintaxis o de estructura de los archivos modificados por los ingenieros. Existen procedimientos para garantizar que las modificaciones realizadas a los componentes correspondan a los nuevos requerimientos establecidos para ese ciclo de desarrollo, y para verificar y validar que las nuevas versiones de componentes estén libres de errores.

Este proyecto tiene como objetivo la creación de un prototipo funcional de una herramienta que soporte estos procesos de validación. Al tratarse de una herramienta a la medida, diseñada especialmente para soportar un proceso específico, y ajustarse a la estructura ya determinada de estos componentes, no se cuenta con opciones en el mercado ni dentro de la empresa que realicen una función similar.

Beneficios

En el caso de que el prototipo funcional se llegue a implementar, se espera lograr los siguientes beneficios directos:

- Se espera lograr una reducción inmediata y tangible de los defectos en los componentes reportados por nuestros clientes, los cuales son introducidos durante el desarrollo de nuevas versiones de estos. Esto, gracias a que la herramienta podrá hacer una validación automática de la estructura y la sintaxis de la mayoría de los archivos que conforman los diagnósticos, garantizando la adecuada integridad estructural de estos.
- Habilidad de diseñar reportes con las especificaciones del contenido de cada componente. Con eso será posible tener una visión global del estado actual del componente y entender de mejor manera las características de las órdenes existentes, lo cual representará una ventaja estratégica inexistente en la actualidad. Esta información será valiosa en el momento de procesar nuevas órdenes y validar que éstas cumplen los requerimientos estipulados.
- Habilidad para obtener información clara y precisa que pueda servir de apoyo durante todo el ciclo de desarrollo de los componentes. La información será moderada de forma que permita hacer una comparación entre dos versiones diferentes del mismo componente, lo que habilitará al equipo a identificar si se introdujo algún cambio no programado durante el desarrollo de una nueva versión.

Referente Institucional

Hewlett Packard Enterprise Company, Hewlett Packard Enterprise o HPE solamente, es una compañía multinacional dedicada a las tecnologías de la información. Fue fundada el 1ero de noviembre del 2015 como el resultado de la separación de la compañía Hewlett-

Packard. Su división, Enterprise Group, se enfoca en soluciones de tecnología avanzada, tales como servidores, almacenamiento y redes. El equipo de trabajo está constituido por ingenieros con distintos grados en Ciencias de la Computación, Informática o Electrónica con amplio conocimiento en sistemas operativos, arquitectura de computadoras y desarrollo de software.

Misión.

Proveer productos, servicios y soluciones de la más alta calidad y brindar más valor a nuestros clientes que ganen su respeto y lealtad.

Visión.

Ver el cambio en el mercado como una oportunidad para crecer, usar nuestras ganancias y habilidades para desarrollar y producir productos innovadores, servicios y soluciones que satisfagan las necesidades emergentes de los clientes.

Valores.

- Confianza y respeto.
- Logros y contribución.
- Resultados a través del trabajo en equipo.
- Innovación significativa.
- Integridad inquebrantable.

CAPÍTULO I: DIAGNÓSTICO

Durante este capítulo se presentan los detalles del diagnóstico FODA aplicado con el fin de determinar la situación del proyecto en cuestión. Una vez definido qué es un diagnóstico FODA y sus variables, se procede a presentar un resumen de los resultados del análisis.

El diagnóstico FODA

La planeación formal, dentro del ámbito de la administración estratégica, es "un esfuerzo administrativo que sirve para prever condiciones futuras tomando decisiones presentes" (Ramírez, 2012, p. 1). Aunque muchas empresas están habituadas a desarrollar planes, éstas suelen enfrentar ciertos problemas para recopilar y ordenar la información relativa a las condiciones de operación y los recursos del negocio, lo que afecta de manera directa la adopción de estrategias competitivas y, por tanto, la elaboración y ejecución de un plan (Ramírez, 2012, p. 1).

Una herramienta que ayuda en el proceso de análisis de la operación de una empresa es el análisis situacional - también conocido como diagnóstico FODA por sus siglas (fortalezas, oportunidades, debilidades y amenazas) - ya que es la herramienta apropiada para conocer las condiciones reales de actuación de una empresa, que facilita un buen diagnóstico y evaluación en el proceso de planeación estratégica (Ramírez, 2012, p. 1). El análisis FODA consiste en una evaluación de los factores fuertes y débiles (fortalezas y debilidades) que en su conjunto diagnostican la situación interna de una organización, así como su evaluación externa, es decir, las oportunidades y amenazas.

Variables del Análisis FODA

Antes de abordar los pasos del procedimiento del análisis, es conveniente establecer los conceptos de las variables fundamentales que se utilizan, a saber: fortalezas, oportunidades, debilidades y amenazas.

Se inicia con los conceptos de las variables internas y luego con las externas, por razón de agrupar los conceptos dentro de su misma categoría (Ramírez, 2012, p. 55).

Fortalezas.

Es algo en lo que la organización es competente, se traduce en aquellos elementos o factores que estando bajo su control, mantiene un alto nivel de desempeño, generando ventajas o beneficios presentes y claros, con posibilidades atractivas en el futuro (Ramírez, 2012, p. 55).

Las fortalezas pueden asumir diversas formas como: recursos humanos maduros, capaces y experimentados, habilidades y destrezas importantes para hacer algo, activos físicos valiosos, finanzas sanas, sistemas de trabajo eficientes, costos bajos, productos y servicios competitivos, imagen institucional reconocida, convenios y asociaciones estratégicas con otras empresas, etcétera (Ramírez, 2012, p. 55).

En otras palabras, una fortaleza es una función que la organización realiza de forma correcta. Son todos los recursos que se controlan o las capacidades y habilidades que se desarrollan positivamente.

Debilidades.

Significa una deficiencia o carencia, algo en lo que la organización tiene bajos niveles de desempeño y, por tanto, es vulnerable, denota una desventaja ante la competencia, con posibilidades pesimistas o poco atractivas para el futuro. Constituye un obstáculo para la

consecución de los objetivos, aun cuando está bajo el control de la organización (Ramírez, 2012, p. 55).

Al igual que las fortalezas éstas pueden manifestarse a través de sus recursos, habilidades, tecnología, organización, productos, imagen, etcétera (Ramírez, 2012, p. 55). Las debilidades son, entonces, los factores que provocan una posición desfavorable frente a la competencia. Son recursos de los que se carece, habilidades o capacidades que no se poseen, actividades que no se desarrollan positivamente, etcétera.

Oportunidades.

Las oportunidades son las variables que resultan positivas, favorables, explotables y que se deben descubrir en el entorno externo en donde actúa la empresa ya que permiten obtener ventajas competitivas con respecto de otras (Ramírez, 2012, p. 56).

Son aquellas circunstancias del entorno que son potencialmente favorables para la organización y pueden ser cambios o tendencias que se detectan y que pueden ser utilizados ventajosamente para alcanzar o superar los objetivos. Las oportunidades pueden presentarse en cualquier ámbito, como el político, económico, social, tecnológico, etcétera, dependiendo de la naturaleza de la organización, pero en general, se relacionan principalmente con el aspecto mercado de una empresa (Ramírez, 2012, p. 56).

Amenazas.

Las amenazas son situaciones que provienen del entorno y que pueden llegar a atentar contra la permanencia de la organización. Difícilmente, se puede incidir en éstas, sin embargo, es esencial conocerlas para minimizar la capacidad que tienen de afectar.

Son factores del entorno que resultan en circunstancias adversas que ponen en riesgo el alcanzar los objetivos establecidos, pueden ser cambios o tendencias que se presentan

repentinamente o de manera paulatina, las cuales crean una condición de incertidumbre e inestabilidad en donde la empresa tiene muy poca o nula influencia, las amenazas, también, pueden aparecer en cualquier sector como en la tecnología, competencia agresiva, productos nuevos más baratos, restricciones gubernamentales, impuestos, inflación, etcétera (Ramírez, 2012, p. 56).

Diagnóstico FODA del Prototipo

Cuadro 2: Diagnóstico FODA del prototipo.

Fortalezas	**Oportunidades**
F1. Acceso a todas las versiones de componentes.	O1. Promoción de proyectos relacionados.
F2. Documentos de especificación de requerimientos.	O2. Reducción de tiempos.
F3. Procedimientos de validación actuales.	O3. Formalización de los documentos de requerimientos.
F4. Diseño de software orientado a objetos.	O4. Desacoplamiento de componentes.
F5. Apoyo e interés genuino del equipo.	O5. Infraestructura adecuada para la implementación.
F6. Herramienta de desarrollo especializada.	
F7. Disponibilidad de una base de datos.	
F8. Procesamiento de componentes por lote.	
F9. Interfaz de múltiples documentos.	
Debilidades	**Amenazas**

D1. La aplicación requiere que los componentes tengan el formato	A1: La aplicación corre riesgo de devenir obsoleta a corto plazo.
D2. Software de propósito específico.	A2: Poca experiencia con el lenguaje Java.
D3. Interface gráfica optimizada para una resolución específica.	
D4. Los procesos de análisis consumen importantes cantidades de memoria.	

Fuente: Propia

Fortalezas.

F1. Acceso a todas las versiones de componentes.

Para la elaboración del prototipo, se ha obtenido acceso a la totalidad de componentes de software que han sido creados hasta la fecha. Estos componentes corresponden a varias plataformas o modelos de servidores diferentes y contienen soporte para cientos de órdenes con características diferentes, lo cual será una ventaja al diseñar los procedimientos para extraer la información requerida de los archivos que conforman los componentes y durante la etapa de prueba de dichos procedimientos.

F2. Documentos de especificación de requerimientos.

Se cuenta con varios ejemplares de los documentos de especificación de requerimientos utilizados durante la integración y validación de componentes, los cuales contienen todos los aspectos que deben ser considerados referentes a una orden o configuración. Estos constituyen una referencia importante durante la etapa de diseño del prototipo y al generar las clases y tipos de datos que representen estos aspectos.

F3. Procedimientos de validación actuales.

Se cuenta con información detallada de la totalidad de procedimientos utilizados en la actualidad para validar cada componente de software, la cual será considerada para que el diseño de la aplicación se acople de la mejor manera a las necesidades del equipo de validación. Durante el diseño de los módulos de validación, análisis, exploración, comparación y consultas se considerarán estos procedimientos para ofrecer la funcionalidad adecuada que facilite las tareas de los validadores de componentes.

F4. Diseño de software orientado a objetos.

El diseño orientado a objetos permite enfocar el desarrollo en términos de objetos al planificar el código y la definición de sus interacciones para proveer la funcionalidad requerida, lo cual es adecuado ya que la información contenida en los componentes, relacionada con la manufactura de servidores, es de una naturaleza muy heterogénea. Esta información debe ser extraída y almacenada en estructuras de datos adecuadas para su análisis y transformación.

F5. Apoyo e interés genuino del equipo.

A lo largo de la elaboración del proyecto, se cuenta con la asistencia de la totalidad del equipo de validación y desarrollo en caso de que se tenga alguna consulta referente a los procedimientos utilizados durante la validación, claro está, mientras la carga de trabajo lo permita. Este equipo tiene un interés genuino por la creación de herramientas que ayuden a lo largo del proceso de validación de estos componentes. Se estima que este apoyo por el éxito del proyecto será de vital importancia durante la elaboración del prototipo.

F6. Herramienta de desarrollo especializada.

La herramienta de desarrollo Java incluye soporte nativo para la validación y análisis de archivos XML, la cual será de utilidad, pues la validación y análisis de componentes requieren de la lectura de una considerable cantidad de archivos XML. Se requiere, además, de la adecuada lectura y creación de una cantidad importante de archivos de texto por lo que su soporte nativo representa una importante ventaja.

F7. Disponibilidad de una base de datos.

Mientras, en la actualidad, se cuenta con archivos desconectados y con datos redundantes, el proyecto permitirá que todos los datos estén integrados en una base de datos única que permita, evitar redundancia de datos, por medio de la normalización de sus tablas, y permitirá, además, proteger la integridad de los datos. Gracias a ésta el usuario dispondrá de información precisa y oportuna de apoyo para lograr sus objetivos.

F8. Procesamiento de componentes por lote.

El prototipo incluirá funcionalidad para analizar un conjunto importante de componentes sin requerir ninguna interacción por parte del usuario, lo que permitirá manejar grandes cantidades de trabajo óptimamente y sin tiempos de inactividad, en vez de tener que procesar cada componente de forma individual.

F9. Interfaz de múltiples documentos.

Se espera diseñar la interface gráfica como una interfaz de documentos múltiples, en la cual las diferentes ventanas de la aplicación residen dentro de una ventana madre, la cual contiene un menú. Esto es especialmente útil porque los usuarios necesitarán utilizar varias ventanas al mismo tiempo, las cuales pueden ser minimizadas y maximizadas de forma independiente.

Debilidades.

D1. La aplicación requiere que los componentes tengan el formato adecuado.

La aplicación incluye funcionalidad para extraer la información requerida de los archivos que constituyen los componentes; si algún archivo no tiene el formato o la estructura adecuada, la aplicación estará en capacidad de desplegar un mensaje de error y continuar analizando los demás ficheros, pero no podrá obtener la totalidad de la información hasta que este archivo sea reparado por un desarrollador.

D2. Software de propósito específico.

El software cumple una función muy particular, por lo cual no existen aplicaciones anteriores, ni internas ni comerciales que puedan utilizarse como referencia. Además, la aplicación solo puede extraer información del tipo de componentes específico generados por este equipo de trabajo, y no puede ser utilizado para validar componentes producidos por otros equipos dentro de la misma división.

D3. Interface gráfica optimizada para una resolución específica.

Se planea optimizar la interface gráfica para ser utilizada en una resolución de pantalla de 1920 * 1080 pixeles. Esto no quiere decir que la aplicación no pueda ser utilizada en otras resoluciones de pantalla, pero los controles serán desplegados de forma óptima solamente utilizando la resolución recomendada.

D4. Los procesos de análisis consumen importantes cantidades de memoria.

Dada la complejidad de los algoritmos que extraen la información dispersa en los componentes y la cantidad importante de archivos que los constituyen, se *estima* que la aplicación consumiría una cantidad de memoria. Los equipos del departamento incluyen 8 GB de memoria RAM, lo cual debería ser más que suficiente para manejar la carga de

trabajo, pero debe considerarse de todas formas que el desempeño del equipo puede verse comprometido si otros procesos pesados se ejecutan junto al análisis.

Oportunidades.

O1. Promoción de proyectos relacionados.

Varias de las necesidades que llevaron a considerar la posibilidad de iniciar este proyecto, también, han generado otros esfuerzos relacionados con el mejoramiento del proceso de validación de software, tales como una revisión de los procedimientos utilizados para su producción y verificación. Una vez completado el prototipo, tanto los modelos de diseño, como los reportes generados por la aplicación, puede ser empleada por otras iniciativas similares diferentes de mejoramiento de procedimientos.

O2. Reducción de tiempos.

Se prevé que la implementación de este software va a reducir de forma significativa el tiempo de validación y producción de los componentes, gracias a las características de procesamiento por lotes y validación automática de la herramienta, considerando además, que el proceso de verificación actual es un proceso difícil y lento. Una reducción en el tiempo invertido implica una optimización en el uso de los recursos y se traduce en mayor competitividad.

O3. Formalización de los documentos de requerimientos.

Los procesos de diseño que se llevarán a cabo durante la elaboración del presente proyecto incluyen el modelado de los documentos de requerimientos utilizados, actualmente, por el equipo de desarrollo de la empresa. La estructura de estos documentos no está especificada formalmente, lo que ocasiones frecuentemente problemas de interpretación.

Parte de la especificación de software puede ser utilizada para crear un documento que describa las reglas para generar e interpretar documentos de requerimientos.

O4. Desacoplamiento de componentes.

Dentro del diseño de la aplicación se contempla el diseño de componentes independientes que cumplen funciones específicas. De esa forma, es un solo módulo el encargado de comunicarse con la base de datos, mientras son otras las encargadas de analizar la información. Por lo tanto, si en el futuro se decide utilizar otro motor de base de datos, o diseñar una interface web en lugar de la aplicación de escritorio, solo se requiere cambiar los módulos correspondientes.

O5. Infraestructura adecuada para la implementación.

La división, encargada principalmente del diseño de software, cuenta con la infraestructura ideal para garantizar el buen funcionamiento de la aplicación, una vez implementada, y en un ambiente de producción real. Esta estructura incluye servidores de estándar industrial, con tolerancia a fallos y con enormes capacidades de procesamiento y almacenamiento. La infraestructura goza de constante mantenimiento y mejoras.

Amenazas.

A1: La aplicación corre riesgo de devenir obsoleta a corto plazo.

Las estrategias de diagnóstico varían frecuentemente, además es común tener que ampliar los diagnósticos para incluir soporte, tanto a nuevos productos como a nuevas opciones de hardware y software, por lo cual la estructura de los archivos que constituyen los componentes puede tener variaciones importantes frecuentemente. Es importante que el equipo de validación disponga de tiempo y recursos para brindar el mantenimiento

apropiado al software, de forma que se pueda adecuar su funcionalidad a los cambios requeridos y evitar que la herramienta se vuelva obsoleta con el tiempo.

A2: Poca experiencia con el lenguaje Java.

El equipo de trabajo está conformado por ingenieros experimentados en lenguajes de programación utilizados en procesamiento por lotes, tales como PERL y PYTHON. La programación orientada a objetos y el desarrollo de interfaces gráficas son paradigmas relativamente diferentes a las actividades comúnmente desempeñadas por parte del equipo de trabajo. Lo que implica un esfuerzo extra a la hora de asumir la responsabilidad de darle el debido mantenimiento a la herramienta.

CAPÍTULO II: MARCO TEÓRICO

En este capítulo, se procede a presentar y definir varios conceptos relacionados con el planteamiento del problema central que ha motivado la elaboración del presente trabajo.

Se han incluido tópicos afines a la temática del hardware y de los servidores, conceptos profundamente relacionados al proyecto. Además, se tratan los conceptos afines a *los lenguajes de programación*, incluyendo HTML, XML, PERL y particularmente a Java, el cual será el lenguaje utilizado durante el desarrollo del prototipo.

El prototipo

Un prototipo es una versión inicial de un sistema de software que puede utilizarse, entre otras cosas, para demostrar conceptos, discutir opciones de diseño y obtener más información sobre el problema que pretende resolver y sus posibles soluciones.

El prototipo permite a los usuarios comprobar en qué grado el sistema es conveniente y de qué forma los apoya de mejor manera en las tareas de su trabajo. También, sirve para comprobar la factibilidad del diseño propuesto, por ejemplo, puede verificarse si las estructuras de datos manejan de forma eficiente la información o si la interface gráfica es intuitiva y fácil de usar (Sommerville, 2011, p. 45).

Algoritmos

Un algoritmo es un método para resolver un problema. Y consiste en el enunciado de las reglas, paso a paso, para obtener una solución. Realizar un buen programa requiere el diseño de algoritmos y estructuras de datos correctas. Un algoritmo debe ser preciso e indicar el orden de realización de cada paso (Joyanes, 2006, pp. 17-19).

Lenguajes de programación

Cuando el procesador es una computadora, el algoritmo se ha de expresar en un formato que se denomina lenguaje de programación, ya que el pseudocódigo o el diagrama de flujo no son comprensibles por la computadora. Las operaciones que conducen a expresar un algoritmo en forma de programa se llaman programación. Así pues, los lenguajes utilizados para escribir programas de computadoras son los lenguajes de programación. Los programadores son, por lo tanto, los escritores y diseñadores de programas (Joyanes, 2006, p. 35).

Java

Para el desarrollo de este proyecto se ha seleccionado el *lenguaje de programación* Java, el cual es un lenguaje de alto nivel, de propósito general, orientado a objetos, multiplataforma, creado por James Gosling para Sun Microsystem (Deitel & Deitel, 2008). El principal atractivo de Java es que se trata de un lenguaje seguro y *portable* que soporta las construcciones de la programación moderna orientada a objetos (Allen, 2000, p.24).

Java es considerado una de las principales plataformas de desarrollo del mundo, con nueve millones de desarrolladores alrededor del mundo, tres mil millones de teléfonos y 97% de las computadoras empresariales corriendo Java (Beneke & Wieldt, 2013). En la actualidad, Java se utiliza para desarrollar aplicaciones empresariales a gran escala, para mejorar la funcionalidad de los servidores web, para proporcionar aplicaciones para los dispositivos domésticos y para muchos otros propósitos" (Deitel & Deitel, 2008, p. 8).

Entorno de ejecución Java

El entorno de ejecución Java (Java Runtime Environment o JRE) es lo que se obtiene al descargar el software de Java. JRE está formado por la máquina virtual Java (JVM),

clases del núcleo de la plataforma Java y bibliotecas de soporte. JRE es todo lo que un sistema necesita para ejecutar Java en el explorador web (Oracle Corporation, 2017).

Python

Python es un lenguaje de programación interpretado, de gran escala, utilizado en el procesamiento por lotes, con altas prestaciones para la manipulación de texto y archivos. Es un lenguaje orientado a objetos con una plataforma robusta para el soporte de programación de red (comunicación de procesos a través de una red) (Jones & Drake, 2002).

Perl

Perl, o lenguaje práctico para la extracción e informe, es un lenguaje de propósito general, de alto nivel e interpretado creado por Larry Wall en 1987 con el objetivo de facilitar la creación de reportes. Desde entonces, ha sufrido muchos cambios y revisiones. Provee poderosas facilidades para el procesamiento de texto facilitando la manipulación de archivos de este tipo. Otras de sus principales características son facilidad de uso, soporta, tanto la programación estructurada como la programación orientada a objetos y dispone de una enorme colección de módulos (Schwartz, et al., 2011).

HTML

HTML o lenguaje de marcado de hipertexto es un lenguaje de marcas utilizado por los exploradores web, tales como Internet Explorer o Mozilla Firefox para especificar la forma como una página web es desplegada (Deitel & Deitel, 2008, p. 78).

Junto con el texto, estos documentos incorporan elementos que contienen información adicional acerca de su estructura y formato. La mayoría de los elementos de HTML se delimitan mediante pares de etiquetas. El formato puede especificar en qué lugar de la

página web se despliega un elemento y el tipo y color de letra que debe ser utilizado (Deitel & Deitel, 2008, p. 121).

XML

XML o lenguaje de marcado extensible es un metalenguaje diseñado para almacenar datos, de fácil lectura, tanto para humanos como para máquinas, optimizado para su utilización en Internet. (Jones & Drake, 2002). Cada documento XML tiene una estructura física y lógica. Físicamente, cada documento está compuesto de unidades llamadas entidades. Cada documento comienza con una raíz de entidad de documento. Las estructuras lógicas y físicas deben estar anidadas apropiadamente (Bray & Paoli, 2008).

XML se está convirtiendo en el estándar de facto para la comunicación de datos en la industria del software y como el medio primario de intercambio de información entre negocios (Connolly & Begg, 205, p. 2).

Entorno integrado de desarrollo

Un entorno integrado de desarrollo es una aplicación que proporciona servicios integrales para facilitar el desarrollo de software. Este puede incluir un editor de texto o editor de código fuente, con funciones que faciliten la producción de software destacando sintáctica y semánticamente código fuente y servicios de auto-completado de código (Oracle Corporation, 2016).

NetBeans

NetBeans es un entorno de desarrollo integrado, modular, escrito en el lenguaje de programación Java. Este proyecto consiste en un ambiente de desarrollo con funcionalidad completa y de código abierto con una rica plataforma de aplicaciones cliente, que puede ser utilizada como un entorno de trabajo para la construcción de cualquier tipo de aplicación

(Oracle Corporation, 2016). NetBeans ha sido escogido como el entorno integrado de desarrollo para este proyecto.

Interfaz de programación de aplicaciones

Interfaz de programación de aplicaciones o API es un conjunto de definiciones de subrutinas, protocolos y herramientas utilizadas para construir aplicaciones de software. En términos generales, consiste en un conjunto claramente definido de métodos de comunicación entre varios componentes de software diseñados para facilitar la producción de software al facilitar bloques de construcción de código que son ensamblados por el programador (Deitel & Deitel, 2008).

Base de datos

Una base de datos consiste en una colección de datos relacionados. Un sistema de gestión de bases de datos (DBMS) es el software que gestiona y controla el acceso a la base de datos. Un sistema de base de datos es, entonces, un programa que interactúa con la base de datos en algún punto de su ejecución (Connolly & Begg, 205, p. 4).

Una base de datos es un repositorio de datos único, posiblemente extenso, que puede ser utilizado simultáneamente por muchos departamentos y usuarios. En lugar de utilizar archivos desconectados con datos redundantes, todos los datos son integrados con un mínimo de duplicación (Connolly & Begg, 205, p. 15).

Oracle

Oracle, u Oracle Corporation, es una compañía multinacional de software, localizada en California, que se especializa en el desarrollo bases de datos, sistemas de gestión de bases de datos y software empresarial (Oracle Corporation, 2012).

Microsoft

Microsoft Corporation es una compañía multinacional de tecnología localizada en Washington. Desarrolla, fabrica y brinda soporte a software, computadoras personales y soporte técnico. Sus productos más conocidos son Microsoft Windows, Microsoft Office e Internet Explorer (Balena, 2006).

MySQL

MySQL es una base de datos de código abierto, la tercera más utilizada después de Oracle y Microsoft SQL Server. Su facilidad de uso ha sido un objetivo del diseño desde sus inicios y un factor importante en su adopción y popularidad.

MySQL es la base de datos de código abierto más popular del mundo con más de 15 millones de instalaciones estimadas y decenas de miles de nuevas descargas cada día. Fue adquirida por Oracle Corporation en 2010. (Oracle Corporation, 2012).

MySQL Workbench

Es una herramienta de modelado y diseño de base de datos visual, un editor SQL y una herramienta administrativa. MySQL Workbench permite crear y validar el esquema para nuevas bases de datos MySQL y optimizar las bases de datos físicas existentes con ingeniería inversa (Oracle Corporation, 2012).

Programación orientada a objetos

El acercamiento de la programación orientada a objetos empieza al identificar los objetos involucrados en un problema y los mensajes a los que estos deben responder. El programa resultante es una colección de objetos, cada uno con sus propios datos y su propio conjunto de responsabilidades. (Bruegge & Allen, 2002, pp. 11-12).

Al contrario que la programación procedimental, que se enfoca en los algoritmos, ésta se enfoca en los datos. La idea es diseñar formatos de dato que se correspondan con las características esenciales del problema al combinar en una única unidad o módulo, tanto los datos como las funciones que operan sobre esos datos. Tal unidad se llama un objeto (Joyanes, 2006, p. 22).

Análisis de requerimientos orientado a objetos

En el análisis de requerimientos orientado a objetos, se modelan entidades del mundo real usando clases de objetos. Se puede crear diferentes tipos de modelos de objetos, que muestren cómo se relacionan mutuamente las clases de objetos, cómo se agregan objetos para formar otros objetos, cómo interactúan los objetos entre sí, etcétera. Cada uno de éstos presenta información única acerca del sistema que se especifica (Sommerville, 2011, p. 129).

La ingeniería de requerimientos es el proceso de aplicar un método de análisis estructurado, tal como el análisis orientado a objetos, con el fin de analizar el sistema y desarrollar un conjunto de modelos gráficos de éste, como los modelos de caso de uso, que luego sirven como especificación del sistema. El conjunto de modelos describe el comportamiento del sistema y se anota con información adicional (Sommerville, 2011, p. 100).

Documento de requerimientos de software

El documento de requerimientos de software (llamado algunas veces especificación de requerimientos de software o SRS) es un comunicado oficial de lo que deben implementar los desarrolladores del sistema. Incluye, tanto los requerimientos del usuario para un sistema, como una especificación detallada de los requerimientos del sistema (Sommerville, 2011, p. 91).

Modelado

El modelado es el proceso utilizado para desarrollar modelos abstractos de un sistema, donde cada modelo presenta una visión o perspectiva diferente de dicho sistema. En general, el modelado de sistemas se ha convertido en un medio para representar el sistema usando algún tipo de notación gráfica, que ahora casi siempre se basa en notaciones en el Lenguaje de Modelado Unificado (UML) (Sommerville, 2011, p. 119).

El aspecto más importante de un modelo del sistema es que deja fuera los detalles. Un modelo es una abstracción del sistema a estudiar, y no una representación alternativa de dicho sistema. De manera ideal, una representación de un sistema debe mantener toda la información sobre la entidad a representar. Una abstracción simplifica y recoge deliberadamente las características más destacadas (Sommerville, 2011, p. 138).

Abstracción

La abstracción es la propiedad de los objetos que consiste en tener en cuenta sólo los aspectos más importantes desde un punto de vista determinado y no tener en cuenta los restantes aspectos. En programación, se refiere al hecho de diferenciar entre las propiedades externas de una entidad y los detalles de la composición interna de dicha entidad. Es la abstracción la que permite ignorar los detalles internos de un dispositivo complejo (Joyanes, 2006, p. 24). Durante el proceso de diseño se utilizará la abstracción para describir la arquitectura del prototipo.

Clases

En programación orientada a objetos, los objetos son miembros de clases, las cuales son un tipo de dato, como cualquier otro, pero que contiene, tanto datos como funciones. Una clase debe ser definida, aunque su definición no implica creación de objetos (Joyanes,

2006, p. 28). A diferencia de los tipos de dato abstractos, las clases pueden definirse en términos de otras clases al usar la generalización (Bruegge & Allen, 2002, p. 39).

Atributos

Las funciones de un objeto, también llamada función miembro o métodos, son el único medio para acceder a sus datos. Estos datos se conocen, también, como atributos o variables de instancia.

Si se desea leer datos de un objeto, se llama a una función miembro del objeto, se accede a los datos y se devuelve un valor. No se puede acceder a estos directamente. Los datos están ocultos, de modo que están protegidos de alteraciones accidentales.

Los datos y las funciones están encapsulados en una única entidad. El encapsulamiento de datos y la ocultación de los datos son términos claves en la descripción de lenguajes orientados a objetos (Joyanes, 2006, p. 22).

Diagramas de clase

Los diagramas de clase pueden usarse cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases. Una clase de objeto se considera como una definición general de un tipo de objeto del sistema. Una asociación es un vínculo entre clases, que indica que hay una relación entre dicha clases. En consecuencia, cada clase puede tener algún conocimiento de esta clase asociada (Sommerville, 2011, p. 129).

Diagramas UML

UML es un lenguaje gráfico para modelado de programas de computadoras. Modelado significa, como su nombre indica, crear modelos o representaciones de algo, como un plano de una casa o similar. UML proporciona un medio de visualizar la

organización de alto nivel de los programas sin fijarse con detenimiento en los detalles del código real (Joyanes, 2006, p. 43).

Los diagramas de clase en el UML pueden expresarse con diferentes niveles de detalle. Cuando se desarrolla un modelo, la primera etapa con frecuencia implica buscar en el mundo, identificar los objetos esenciales y representarlos como clases. La forma más sencilla de hacer esto es escribir el nombre de la clase en un recuadro. También, puede anotar la existencia de una asociación dibujando simplemente una línea entre las clases (Sommerville, 2011, p. 129).

Tipos de diagramas UML

Entre los diversos tipos de diagramas UML se pueden citar los siguientes:

- Diagramas de actividad, que muestran las actividades incluidas en un proceso o en el procesamiento de datos.
- Diagramas de caso de uso, que exponen las interacciones entre un sistema y su entorno.
- Diagramas de secuencias, que muestran las interacciones entre los actores y el sistema, y entre los componentes del sistema.
- Diagramas de clase, que revelan las clases de objeto en el sistema y las asociaciones entre estas clases.

(Sommerville, 2011, p. 120)

Casos de uso

Los casos de uso son una técnica de descubrimiento de requerimientos que ha convertido en una característica fundamental del modelado de lenguaje unificado. En su

forma más sencilla, un caso de uso identifica a los actores implicados en una interacción, y nombra el tipo de interacción. Entonces, esto se complementa con información adicional que describe la interacción con el sistema. Los actores en el proceso, que pueden ser individuos u otros sistemas, se representan como figuras sencillas (Sommerville, 2011, p. 125).

Tipos de datos

Un tipo de datos es una abstracción en el contexto de los lenguajes de programación. Un tipo de datos tiene un nombre único que lo distingue de otros tipos de datos. Los tipos de datos son utilizados en los lenguajes de programación para asegurar que solo operaciones válidas sean aplicadas a tipos de datos específicos (Bruegge & Allen, 2002, p. 38).

Módulo

Un módulo es un componente de un sistema más grande que interactúa con el resto del sistema de forma sencilla y bien definida. Los detalles de que contiene el módulo no son importantes para el sistema, mientras éste cumpla su objetivo correctamente. Esto se llama ocultamiento de información y es uno de los principios más importantes de la ingeniería de software (Bruegge & Allen, 2002, p. 11).

Herencia

En orientación a objetos, el mecanismo que implementa la propiedad de generalización se denomina herencia. La herencia permite definir nuevas clases a partir de otras clases ya existentes, de modo que presentan las mismas características y comportamiento de éstas, así como otras adicionales (Ramírez, 2012, p. 29).

Al crear una clase, en vez de declarar miembros completamente nuevos, el programador puede designar que la nueva clase herede los miembros de una clase existente.

Esta clase existente se conoce como superclase, y la nueva clase se conoce como subclase. Una subclase, generalmente, agrega sus propios campos y métodos. Por lo tanto, una subclase es más específica que su superclase y representa a un grupo más especializado de objetos (Deitel & Deitel, 2008, p. 417).

Durante el desarrollo del presente proyecto, se ha considerado oportuno utilizar herencia para el desarrollo de varios elementos del prototipo.

Computadora

Una computadora es una máquina electrónica compleja, constituida por una variedad de dispositivos diferentes (Eck, 2014). Existen en diversos formatos dependiendo de la forma y función de los componentes que las constituyen, tales como la computadora de escritorio o notebooks. Muchas computadoras pueden ser utilizadas como estaciones de trabajo conectadas a una red, pero no pueden ejecutar la función de un servidor de red (Brooks, et al., 2006).

Hardware

Una computadora está compuesta por diversos dispositivos conocidos como hardware. Los costos de estos componentes de hardware han disminuido de manera espectacular en años recientes, al punto en que las computadoras personales se han convertido en artículos domésticos (Deitel & Deitel, 2008, p. 4). Los sistemas de servidor suelen incluir tarjetas madres, procesadores, memorias de acceso aleatorio, discos duros, fuentes de poder y tarjetas periféricas (Brooks, et al., 2006).

Software

El enfoque central de este trabajo consiste en la validación de componentes de *software*. Estos componentes, junto con otros componentes similares, conforman los diagnósticos utilizados durante la fabricación o ensamblaje de *servidores*.

Las computadoras procesan los datos bajo el control de conjuntos de instrucciones llamadas programas de cómputo (Deitel & Deitel, 2008, p. 4). A estos programas, los cuales incluyen aplicaciones de diversa naturaleza, tales como sistemas operativos, aplicaciones de ofimática, exploradores web, etcétera, se les denomina software" (Deitel & Deitel, 2008, p. 27).

Sistema Operativo

Un sistema operativo es tal vez la parte más importante del software del sistema y es el software que controla y gestiona los recursos del computador. En la práctica, el sistema operativo es la colección de programas de computador que controla la interacción del usuario y el hardware del computador.

El sistema operativo es el administrador principal del computador, y por ello a veces, se le compara con el director de una orquesta ya que este software es el responsable de dirigir todas las operaciones del computador y gestionar todos sus recursos. El sistema operativo asigna recursos, planifica el uso de recursos y tareas del computador, y monitoriza a las actividades del sistema informático. Estos recursos incluyen memoria, dispositivos de entrada/salida), y la unidad central de proceso (Joyanes, 2006, p. 33).

Interfaz de documento único

A diferencia de las interfaces de documento múltiple (SDI) una aplicación de documento único permite un solo documento abierto a la vez. Es, además, una manera de

organizar una interface gráfica de forma que cada ventana individual es manejada por separado por el sistema operativo, apareciendo ésta como una entrada individual en la barra de tareas del sistema operativo (Microsoft, 2016).

Interfaz de múltiples documentos

Las interfaces de múltiples documentos o MDI, permiten que múltiples formularios o documentos sean abiertos al mismo tiempo en la misma instancia de una aplicación. Una aplicación MDI contiene una ventana principal, dentro de la cual residen múltiples ventanas hijo (Microsoft, 2016).

Arquitectura del sistema.

Según Sommerville, el diseño arquitectónico se interesa por entender cómo debe organizarse y cómo tiene que diseñarse la estructura global de un sistema. El diseño arquitectónico es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. La salida del proceso de diseño arquitectónico consiste en un modelo arquitectónico que describe la forma como se organiza el sistema como un conjunto de componentes en comunicación (Sommerville, 2011, p. 148).

Windows

Microsoft Windows o simplemente Windows, es una familia de sistemas operativos gráficos desarrollado y distribuido por Microsoft. En la actualidad, la versión más reciente de Windows para computadoras personales, tabletas y teléfonos inteligentes es Windows 10 (Pérez, 2016).

Servidores

Los servidores son computadoras con características especiales, adecuados para su uso en redes comerciales. Pueden ser utilizados como servidores de red, proporcionando servicios como proxy, firewall, servidor de bases de datos y otros. A los programas que se ejecutan en una computadora se les denomina software" (Deitel & Deitel, 2008, p. 27). Los servidores que sostienen estas redes están constituidos por configuraciones físicas que son distintas de las computadoras comerciales, frecuentemente utilizando múltiples procesadores y arreglos de discos (Brooks, et al., 2006).

Hardware de servidores

Un chasis típico de servidor incluye los siguientes componentes de hardware:

- Una tarjeta madre, la cual puede utilizar múltiples microprocesadores (de 2 a 32), grandes cantidades de memoria de acceso aleatorio y sensores integrados.
- Múltiples fuentes de poder redundantes, las cuales proveen tolerancia a fallos y evitan que el servidor se apague en caso de fallo de una de las fuentes de poder.
- Múltiples discos duros, configurados en arreglos de discos, proveen tolerancia a fallos de alta velocidad para el subsistema de discos duros.
- Múltiples tarjetas de interface de red, utilizadas para proveer tolerancia a fallos para la conexión de red.
- Múltiples ventiladores de enfriamiento, para evitar sobrecalentamiento.

(Brooks, et al., 2006)

BIOS

EL BIOS, de las iniciales en inglés de Sistema Básico de Entrada/Salida es una aplicación de configuración embebida en la memoria de solo lectura del sistema. Este se encarga de un amplio rango de actividades de configuración entre las que se incluyen: configuración de dispositivos de sistema y opciones instaladas, despliegue de información de sistema, selección de controlador de arranque primario, etcétera (Hewlett Packard, 2015).

Los procesos de software

Según Sommerville, los procesos de software "son la serie de actividades relacionadas que conduce a la elaboración de un producto de software y un modelo es una representación simplificada de este modelo". Además, al enfoque sistemático que se usa en la ingeniería de software se le conoce en ocasiones como proceso de software. Un proceso de software es una secuencia de actividades que conducen a la elaboración de un producto de software (Sommerville, 2011, p. 9).

Las actividades fundamentales que son comunes a todos los procesos de software son: especificación de software, desarrollo de software, validación de software y evolución de software (Sommerville, 2011, p. 9).

Validación de software

La validación de software o, más generalmente, su verificación y validación (V&V), se crea para mostrar que un sistema cumple, tanto con sus especificaciones como con las expectativas del cliente.

Las pruebas del programa, donde el sistema se ejecuta a través de datos de prueba simulados, son la principal técnica de validación. Esta última, también, puede incluir procesos de comprobación, como inspecciones y revisiones en cada etapa del proceso de

software, desde la definición de requerimientos del usuario hasta el desarrollo del programa (Sommerville, 2011, p. 41).

Apache Subversion

Apache Subversion, abreviado frecuentemente como SVN, es un sistema de control de versión gratuito y de código libre. Esto significa que SVN maneja archivos y directorios, y los cambios hechos a estos, a lo largo del tiempo. Esto permite recuperar viejas versiones de los datos o examinar el historial de cómo estos datos ha cambiado.

SVN puede operar sobre redes, lo que permite que sea usado por varias personas en computadoras diferentes (Collins-Sussman, et al., 2011, p. 13).

CAPÍTULO III: MARCO METODOLÓGICO

El siguiente capítulo incluye la presentación de las técnicas, herramientas, instrumentos y procedimientos utilizados durante la elaboración de este trabajo.

Se explica el tipo y el método de investigación seleccionado y las fuentes de información utilizadas. Además, se incluye un cuadro de variables, el cual ha sido elaborado con el fin de describir las distintas variables, tanto conceptuales, operacionales e instrumentales, que determinan el desarrollo del proyecto.

Al finalizar, se incluye un análisis de la población de interés, la definición de una muestra significativa de esta población y la interpretación de los resultados obtenidos a partir de los datos recolectados.

Método cuantitativo

El enfoque de la investigación cuantitativo "usa la recolección de datos para probar hipótesis, con base en la medición numérica y el análisis estadístico, para establecer patrones de comportamiento y probar teorías" (Hernández, et al., 2010, p. 4).

El enfoque cuantitativo utiliza el método hipotético-deductivo. Se delinean teorías y de ellas se derivan hipótesis o suposiciones acerca de una realidad, se diseña un plan para ponerlas a prueba, se miden los conceptos incluidos en las hipótesis (variables) y se transforman los resultados en valores numéricos o datos cuantificables (Hernández, et al., 2003).

"Si los resultados corroboran las hipótesis o son consistente con éstas, se aporta evidencia en su favor. Si se refutan, se descartan en busca de mejores explicaciones e hipótesis. Cuando los resultados de diversas investigaciones aportan evidencia en favor de

las hipótesis, se genera confianza en la teoría que las sustenta o apoya. Si no es así, se descartan la hipótesis y, eventualmente la teoría." (Hernández, et al., 2003, p. 13).

Método cualitativo

El método cualitativo "utiliza la recolección de datos sin medición numérica para descubrir o afinar preguntas de investigación en el proceso de interpretación antes, durante o después de la recolección y el análisis de datos" (Hernández, et al., 2010, p. 7). Con frecuencia se basa en métodos de recolección de datos sin edición numérica, como las descripciones y las observaciones" (Hernández, et al., 2003, p. 10).

Este busca, principalmente, expandir los datos mientras que el enfoque cuantitativo pretende intencionalmente delimitar la información; en términos generales, implica la recolección de datos utilizando técnicas que no pretenden medir ni asociar las mediciones con números, tales como observación no estructurada, entrevistas abiertas, revisión de documentos, discusión en grupo, evaluación de experimentos personales y otros.

Los datos cualitativos son descripciones detalladas de eventos, personas, y conductas observadas y su manifestación. Esta directamente involucrado con las personas que se estudian y con sus personales (Hernández, et al., 2003, p. 15).

Método utilizado

Durante la elaboración de este proyecto se utilizará una metodología de investigación cuantitativa, pues su enfoque consiste en la obtención de requisitos de sistemas medibles y comprobables, de forma que pueda constatarse su exitosa implementación. Estos serán analizados y documentados.

Posteriormente, se diseñará un plan para validar que el sistema cumple con los requisitos estipulados, midiendo el grado en que estos objetivos se cumplen. Los resultados

de estas mediciones se transforman en valores numéricos o datos cuantificables, tal y como se describe en el cuadro de variables al final del presente capítulo.

Tipo de investigación descriptiva

Con frecuencia, la meta del investigador consiste en describir fenómenos, situaciones, contextos y eventos; esto es, detallar cómo son y se manifiestan. Los estudios descriptivos buscan especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis. Es decir, únicamente pretenden medir o recoger información de manera independiente o conjunta sobre los conceptos o las variables a las que se refieren, esto es, su objetivo no es indicar como se relacionan éstas (Hernández, et al., 2010, p. 80).

"Así como los estudios exploratorios sirven de fundamentos para descubrir y prefigurar, los estudios descriptivos son útiles para mostrar con precisión los ángulos o dimensiones de un fenómeno, suceso, comunidad, contexto o situación" (Hernández, et al., 2010, p. 80).

Tipo de investigación exploratoria

Los estudios exploratorios se realizan cuando el objetivo es examinar un tema o problema de investigación poco estudiado, del cual se tienen dudas o no se ha abordado antes. Es decir, cuando la revisión de la literatura reveló que tan solo hay guías no investigadas e ideas vagamente relacionadas con el problema de estudio, o bien, si se deseara indagar sobre temas y áreas desde nuevas perspectivas (Hernández, et al., 2010, p. 79).

Sirven para familiarizarse con fenómenos relativamente desconocidos, obtener información sobre la posibilidad de llevar a cabo una investigación más completa respecto

de un contexto particular, investigar nuevos problemas, identificar conceptos o variables promisorias, establecer prioridades para investigaciones futuras, o sugerir afirmaciones y postulados (Hernández, et al., 2010, p. 79).

Tipo de investigación explicativa

Los estudios explicativos van más allá de la descripción de conceptos o fenómenos o del establecimiento de relaciones entre conceptos; es decir, están dirigidos a responder por las causas de los eventos físicos o sociales. Como su nombre lo indica, su interés se centra en explicar por qué ocurre un fenómeno y en qué condiciones se manifiesta, o por qué se relacionan dos o más variables (Hernández, et al., 2010, pp. 82-83).

Tipo de investigación utilizado

La investigación requerida para cumplir con los objetivos planteados por esta investigación es, principalmente, de carácter descriptivo. Un diseño de software consiste en una descripción de la estructura del sistema que va a ser implementado, de los modelos y las estructuras de datos que serán utilizados, las interfaces entre los componentes, etcétera.

La obtención de información detallada y precisa de los servicios requeridos para resolver el problema de estudio planteado es de vital importancia. Mucha de esta información será recolectada durante la fase de especificación de requerimientos, por medio de discusiones con los usuarios, análisis de procedimientos y otras herramientas que permitan entender a fondo la necesidad del equipo.

Fuentes de información

Fuentes primarias.

Las referencias o fuentes primarias proporcionan datos de primera mano, pues se trata de documentos que incluyen los resultados de los estudios correspondientes. Ejemplos de

éstas son: libros, antologías, artículos de publicaciones periódicas, monografías, tesis y disertaciones, documentos oficiales, reportes de asociaciones, trabajos presentados en conferencias o seminarios, artículos periodísticos, testimonios de expertos, documentales, videocintas en diferentes formatos, foros y páginas en Internet, etcétera (Hernández, et al., 2006, p. 66).

Entre las fuentes primarias contempladas en la elaboración de este trabajo, se considera el uso de las minutas de entrevistas cerradas o abiertas con los participantes del sistema, la inspección de la documentación interna existente con respecto de los procedimientos de validación existente y la estructura de los componentes que serán validados. Otra fuente primaria principal consistirá en un documento de requerimientos elaborado con las contribuciones de todo el equipo. Además, se espera contar con información derivada de la evaluación de experimentos personales utilizando los componentes proporcionados.

Fuentes secundarias.

Son listas, compilaciones y resúmenes de referencias o fuentes primarias publicadas en un área de conocimiento en particular. Es decir, reprocesan información de primera mano. Comentan brevemente artículos, libros, tesis, disertaciones y otros documentos. Algunas fuentes secundarias incluyen los datos de las referencias y un breve resumen de cada una de éstas (Hernández, et al., 2006, p. 66).

Para la elaboración de este proyecto, se considera el uso de varias fuentes secundarias, incluyendo libros, artículos en línea y manuales referentes a una diversidad de temas, tales como metodología de investigación, desarrollo de software, diseño de sistemas, así como

varios libros dedicados al lenguaje de programación Java, estructuras de datos y programación orientada a objetos.

Fuentes terciarias.

Se trata de documentos donde se encuentran registradas las referencias a otros documentos de características diversas y que compendian nombres y títulos de revistas y otras publicaciones periódicas, así como nombres de boletines, conferencias y simposios, sitios web, empresas, asociaciones industriales y de diversos servicios (Hernández, et al., 2006, p. 69).

Para la elaboración de este proyecto se considera el uso de varias fuentes terciarias. Varios de los libros consultados proveen una bibliografía extensa, relacionada con los distintos temas expuestos. Esta bibliografía ha sido consultada con el fin de obtener las obras más relevantes para elaborar este proyecto.

Variables

Una variable es una propiedad que puede fluctuar y cuya variación es susceptible de medirse u observarse (Hernández, et al., 2010).

Definición conceptual.

Una definición conceptual trata a la variable con otros términos, en otras palabras amplía su descripción para dar una idea más completa de su significancia y el papel que desempeña en el proyecto (Hernández, et al., 2010). Se tratan de definiciones de diccionario o de libros especializados y cuando describen la esencia o las características de una variable, objeto o fenómeno se les denomina variables reales (Hernández, et al., 2010, p. 110).

Definición operacional.

Una definición operacional constituye el conjunto de procedimientos que describe las actividades que un observador debe realizar para recibir las impresiones sensoriales, las cuales indican la existencia de un concepto teórico en mayor o menor grado. En otras palabras, especifica qué actividades u operaciones deben realizarse para medir una variable (Hernández, et al., 2010).

Definición instrumental.

La definición instrumental de una variable consiste en la forma como ésta será analizada para obtener resultados significativos que puedan ser utilizados para apoyar las conclusiones de la investigación (Hernández, et al., 2010).

Cuadro de variables

A continuación, se muestra el cuadro de variables, el cual contiene todas las variables correspondientes a cada uno de los objeticos específicos del proyecto.

Cuadro 3: Cuadro de variables

Objetivo Específico	**Variable**	**Variable Conceptual**	**Variable Operacional**	**Variable Instrumental**
Obtener, analizar, especificar y validar los requerimientos del	Especificación de requerimientos.	La actividad de transcribir la información recopilada durante la actividad de análisis, en un	Obtención y análisis, especificación y validación de requerimientos.	Elaboración y aplicación de un cuestionario de evaluación a los actores

sistema.		documento que define un conjunto de requerimientos. (Sommerville, 2011, p. 38)		del sistema. Elaboración de casos de uso y definición de requerimientos a partir de los resultados obtenidos al aplicar el cuestionario.
Diseñar la arquitectura, los componentes, la interfaz gráfica y las estructuras de datos.	Especificación de arquitectura del sistema.	Un diseño de software es una descripción de la estructura del software que se va a implementar, los modelos y las estructuras de datos utilizados por el sistema y las interfaces entre	Diseño arquitectónico, de componentes y diseño de base de datos.	La arquitectura del sistema se desarrollará a partir de los casos de uso, requerimientos y diagramas UML obtenidos. Uso de los

		componentes del sistema. (Sommerville, 2011, p. 38)		requerimientos para generar casos de prueba de desarrollo. Revisión de los resultados contra los requerimientos.
Desarrollar el prototipo funcional en conformidad con la arquitectura y las especificaciones establecidas.	Prototipo funcional.	Versión inicial de un software utilizado, para demostrar conceptos, discutir opciones de diseño y obtener información sobre el problema que pretende resolver. (Sommerville, 2011)	Desarrollo de componentes, desarrollo de interface, desarrollo de base de datos.	Desarrollado utilizando el IDE NetBeans y MySQL Server. Pruebas de sistema.

Validar y verificar el prototipo con el fin de determinar que éste cumple con las especificaciones establecidas.	Validación y verificación del sistema.	Muestra que un sistema cumple, tanto con sus especificaciones como con las expectativas del cliente. (Sommerville, 2011, p. 41)	Plan de pruebas de aceptación.	Revisión de los resultados de las pruebas de aceptación, generadas con datos suministrados por el cliente en un entorno de producción. Se diseñarán casos de prueba a partir de los requerimientos. Esto será documentado utilizando la herramienta Word.

Fuente: Propia

Población

Una población es el conjunto de todos los casos que concuerdan con una serie de especificaciones. Una vez que se define cuál es la unidad de análisis de una investigación, se procede a delimitar la población a ser estudiada y sobre la cual se pretende generalizar los resultados. Las características de esta población deben ser establecidas con claridad (Hernández, et al., 2010, p. 173).

Dado que el prototipo viene a soportar un esfuerzo de validación hecho por parte de un equipo de diez ingenieros, parte de la división de Servicios de Ingeniería Globales de Hewlett Packard Enterprise se utilizará el criterio de los involucrados en el desarrollo y validación de componentes para la generación del documento de requerimientos de sistema.

Muestreo

Para el proceso cuantitativo la muestra es un subgrupo de la población de interés sobre el cual se recolectarán los datos, y que tiene que definirse o delimitarse de antemano con precisión, éste deberá ser representativo de dicha población (Hernández, et al., 2010, p. 173).

Por lo común, en la investigación en ciencias sociales, la fórmula para calcular el tamaño que debe tener la muestra, para ser considerada significativa, y dada una población finita, se muestra a continuación:

$$n = \frac{Z^2 * N * p * q}{E^2 * (N - 1) + Z^2 * p * q}$$

En donde:

n = Es el tamaño de la muestra que se busca.

N = Tamaño de la población de interés.

Z = Valor en la tabla de distribución normal para un nivel de confianza del 95%.

E = El error máximo aceptable o porcentaje de error potencial de que la muestra no sea representativa.

p = Probabilidad de éxito o proporción esperada.

q= Probabilidad de fracaso (1 – p).

(Hernández, et al., 2010, p. 178)

Entonces, tomando en cuenta que, para un nivel de significancia de 0.05 (Z = 1.96) existe un nivel de confianza de 95% y que la población de interés en nuestro caso consiste en 10 ingenieros, para un margen de error máximo de 5%, se aplica la fórmula utilizando los siguientes valores:

Z = 1.96,

N = 10;

p = 0,5 (Criterio conservador para maximizar el tamaño de la muestra)

q = 0,5

E = 0,05

$$n = \frac{Z^2 * N * p * q}{E^2 * (N-1) + Z^2 * p * q}$$

$$\frac{1.96^2 * 10 * 0{,}5 * 0{,}5}{0{,}05^2 * (10-1) + 1{,}96^2 * 0{,}5 * 0{,}5}$$

$$\frac{9{,}604}{0.9829} = 9.7710$$

Por lo tanto, se establece que 9.7710, redondeado a 10, representa el tamaño óptimo de la muestra, que debe ser analizada para generalizar a toda la población, con un nivel de confianza de 95% y un margen de error de 5%.

Instrumentos de recolección de datos

La etapa de obtención y especificación de requerimientos consiste en el proceso de derivar los requerimientos del sistema mediante observación de los sistemas existentes, discusiones con los usuarios y proveedores potenciales, análisis de tareas, etcétera (Sommerville, 2011, p. 37).

Como parte del desarrollo del proyecto, se procederá a la elaboración de un documento de requerimientos de software, el cual consiste en un comunicado oficial de lo que debe ser implementado por los desarrolladores del sistema (Sommerville, 2011, p. 91).

Éste será elaborado con base en los resultados obtenidos al aplicar un cuestionario a los ingenieros seleccionados que representarán a la población de interés. Los cuestionarios son un instrumento de recolección de datos cuantitativos que se basan en preguntas respecto de una o más variables a medir que pueden ser cerradas o abiertas (Hernández, et al., 2010, p. 196). Las preguntas cerradas contienen categorías u opciones de respuesta que han sido previamente delimitadas. Es decir, se presentan las posibilidades de respuesta a los participantes, quienes deben acotarse a éstas (Hernández, et al., 2010, p. 217).

Cuestionario para la elaboración del prototipo funcional.

Los validadores trabajan directamente con los componentes de software, su tarea principal consiste en confirmar que las nuevas versiones de componentes estén libres de errores y que tengan las características requeridas por los clientes. Para esto, deben confirmar que las nuevas versiones de componentes incluyan los cambios requeridos por los clientes. Además, se debe verificar que la información relacionada con el resto de las órdenes no se haya visto alterada por algún cambio imprevisto.

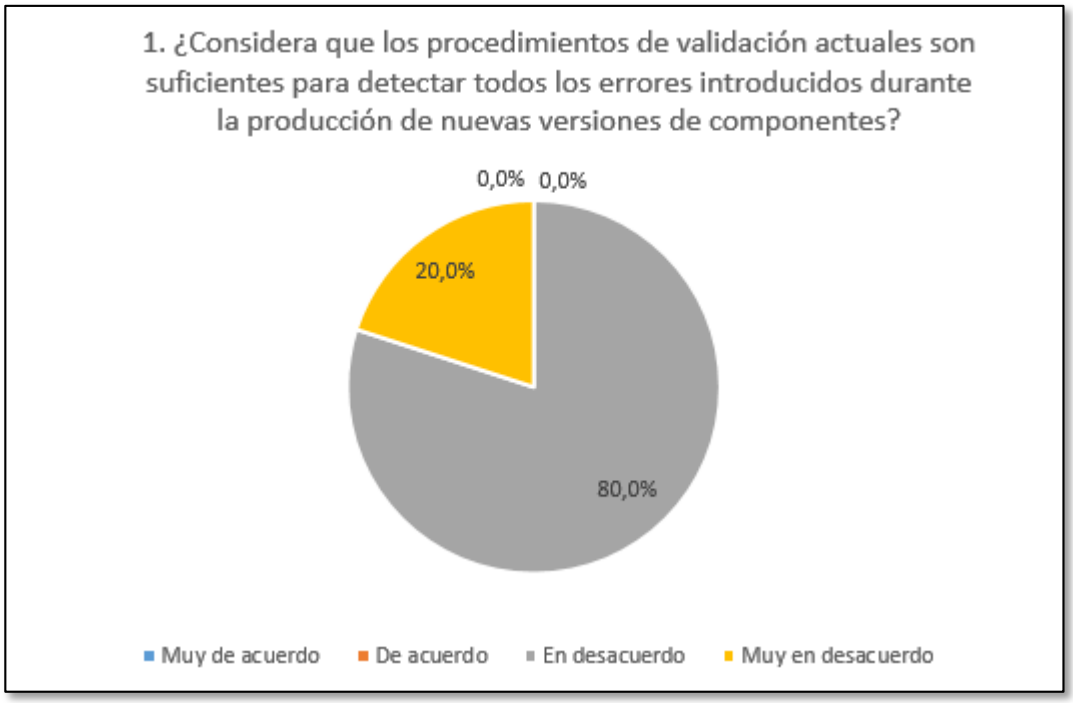
Este cuestionario ha sido aplicado con el fin de entender mejor las necesidades del equipo de validación, y el de encontrar formas más eficientes y rápidas de verificar que la estructura de los componentes esté libre de errores y corresponda a los requerimientos acordados con los clientes. Basado en estos resultados, se procederá a la elaboración del diseño de sistema que servirá de guía durante la elaboración del proyecto.

Interpretación de resultados

A continuación, se presentan los resultados obtenidos al aplicar el cuestionario diseñado como instrumento de medición.

Pregunta 1.

Gráfico 1: Resultados de la pregunta 1



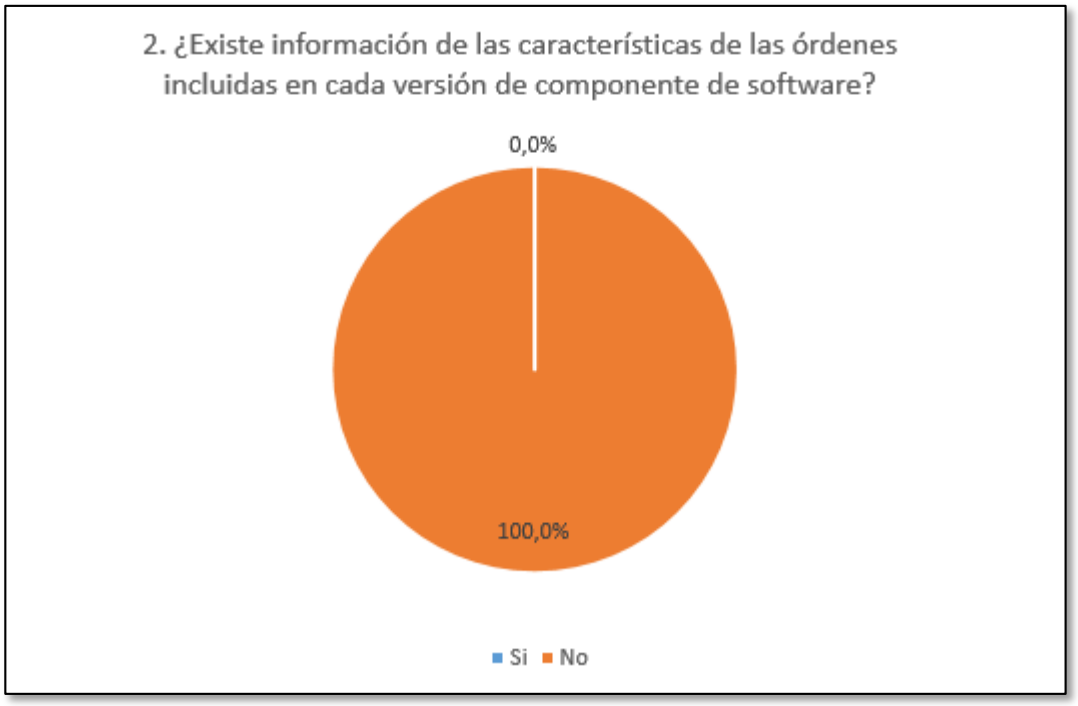
Fuente: Propia

De acuerdo con el gráfico 1, el 80% de los ingenieros consultados considera que está en desacuerdo con respecto de la afirmación de si se cuenta con procedimientos de validación suficientes para detectar todos los errores producidos durante la generación de nuevas versiones de componentes de software. El 20% restante está muy en desacuerdo con la afirmación.

Es importante que el equipo pueda garantizar que los componentes estén libres de errores que puedan comprometer la calidad de las órdenes al no cumplir con los requerimientos y compromisos establecidos. Desafortunadamente, tal y como se desprende de los resultados obtenidos en el gráfico 1, los procedimientos existentes no garantizan todavía que los componentes estén libres de errores. Este aspecto ha sido considerado durante el planteamiento del problema, el cual se ha desarrollado en el apartado "Planteamiento del problema de estudio", en el capítulo correspondiente a la introducción.

Pregunta 2.

Gráfico 2: Resultados de la pregunta 2



Fuente: Propia

Tal y como se puede apreciar en el gráfico 2, el 100% de los consultados estima que no existe información de las características de las órdenes incluidas en cada versión de componente.

Tanto para los ingenieros involucrados en el desarrollo de los componentes, como para los validadores encargados de verificar que los cambios cumplan con los requerimientos acordados, es de suma importancia contar con información precisa de referencia que permita entender el contenido de cada componente de software. Actualmente no se cuenta del todo con documentación relacionada con los componentes, lo cual ha sido considerado en el planteamiento del problema de estudio.

Al igual que la problemática anteriormente descrita este tema se ha desarrollado en el apartado “Planteamiento del problema de estudio”, en el capítulo correspondiente a la introducción.

Pregunta 3.

Gráfico 3: Resultados de la pregunta 3

Fuente: Propia

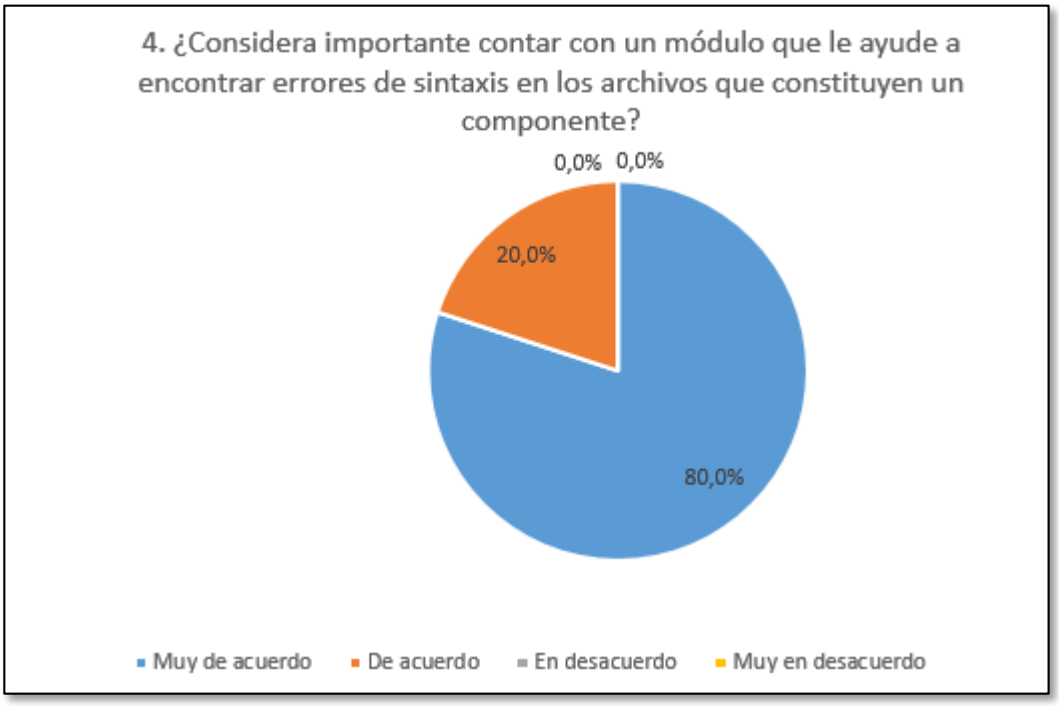
Tal y como se desprende del gráfico 3, el 100% de las personas consultadas aprecia que no cuenta con un sistema informático que le asista durante el proceso de validación de componentes.

La problemática existente requiere la búsqueda de una solución que permita, tanto disminuir o eliminar errores producidos durante la modificación de componentes, como la obtención de información precisa acerca de su contenido.

Dadas las características particulares de los componentes el diseño de un sistema específico para este fin es la solución más adecuada. Es por este motivo por el cual se ha planteado como objetivo general de este trabajo crear un sistema de análisis automatizado, basado en software, que sirva de soporte durante el proceso de validación de componentes, tal y como se describe en el apartado "Objetivo general" en el capítulo correspondiente a la introducción.

Pregunta 4.

Gráfico 4: Resultados de la pregunta 4



Fuente: Propia

El gráfico 4 muestra cómo el 80% de los ingenieros se considera muy de acuerdo con la afirmación "¿Considera importante contar con un módulo que le ayude a encontrar

errores de sintaxis en los archivos que constituyen un componente?". Un restante 20% considera estar de acuerdo con la afirmación.

Por esta razón, se ha decidido incluir el diseño de un módulo de validación dentro del alcance funcional del proyecto, el cual será utilizado por los miembros del equipo para identificar errores de sintaxis dentro de los componentes. Esto con el fin de poder capturar cualquier error de estructura o cualquier error sintáctico presente en el componente. El alcance de este módulo corresponde al apartado de alcance funcional: módulo de validación, en el capítulo correspondiente a la introducción.

Pregunta 5.

Gráfico 5: Resultados de la pregunta 5

Fuente: Propia

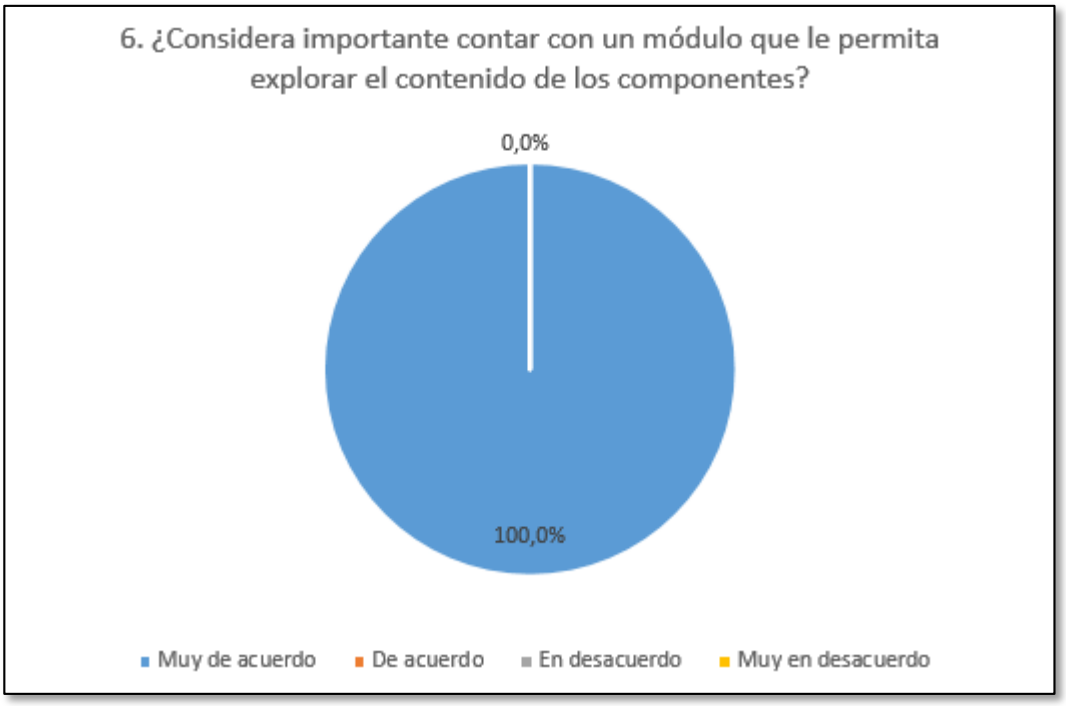
Según el gráfico 5, el 90% de los consultados se considera muy de acuerdo con la afirmación "¿Considera importante contar con un módulo que le permita analizar automáticamente el contenido de un componente?". Un restante 10% manifiesta estar de acuerdo con la afirmación.

Considerando las respuestas favorables, se ha incluido, como parte del alcance funcional del proyecto, el diseño de un módulo de análisis, el cual será utilizado por los miembros del equipo para obtener información directamente de los componentes y almacenarla, tanto en una base de datos como en archivos XML en un formato adecuado para su análisis. El proceso de análisis debería ser llevado a cabo automáticamente, extrayéndose la información directamente de los componentes en cuestión para garantizar que ésta sea fiel e inequívoca.

El alcance de este módulo corresponde al apartado de alcance funcional: módulo de análisis, en el capítulo correspondiente a la introducción.

Pregunta 6.

Gráfico 6: Resultados de la pregunta 6



Fuente: Propia

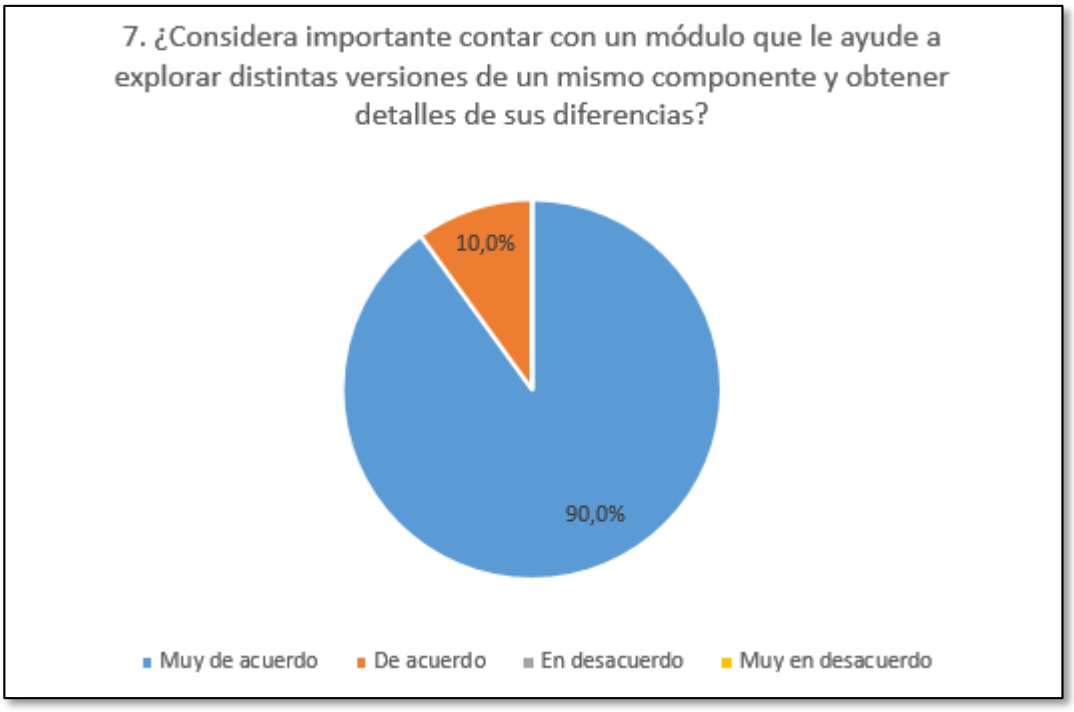
Ante la pregunta, "¿Considera importante contar con un módulo que le permita explorar el contenido de los componentes?", el 100% de los ingenieros consultados estimó estar "muy de acuerdo" con la afirmación.

En correspondencia a los resultados obtenidos, se ha incluido el objetivo de diseñar un módulo que permita explorar el contenido de los componentes de la forma que mejor se adapte a las necesidades del equipo, con el fin de que éste cuente con la información que mejor apoye el proceso de validación.

El alcance del módulo de exploración se describe en el apartado correspondiente al alcance funcional: módulo de exploración.

Pregunta 7.

Gráfico 7: Resultados de la pregunta 7



Fuente: Propia

El gráfico 7 permite apreciar cómo el 90% de los consultados se considera muy de acuerdo con la afirmación "¿Considera importante contar con un módulo que le ayude a explorar distintas versiones de un mismo componente y obtener detalles de sus diferencias?". Un restante 10% declara estar de acuerdo con la afirmación.

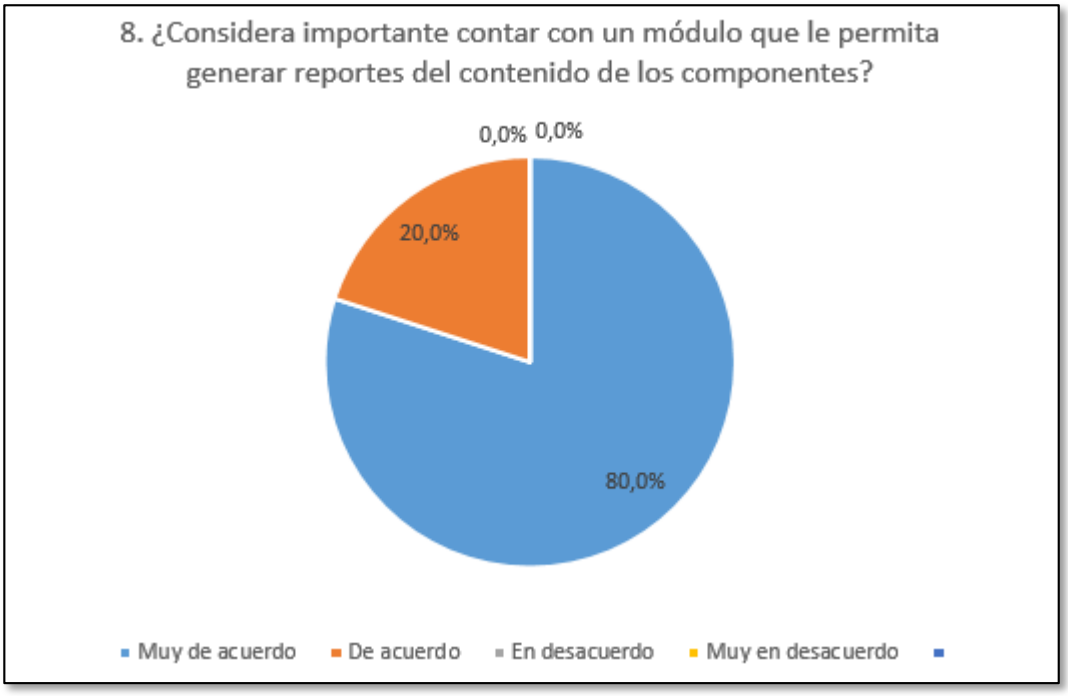
La generación de nuevas versiones de un componente implica un desarrollo incremental, en el cual un componente de software sufriría, por lo general, cambios parciales, mientras el resto del componente permanecería intacto.

El poder comparar dos componentes de diferente versión, de forma que pueda observarse con facilidad cuáles órdenes han cambiado y de qué forma le permitirá al equipo de validación tener un mejor control de los cambios que han sido introducidos durante la

generación de nuevas versiones de software. Es, por esta razón, que el diseño de dicho módulo ha sido contemplado dentro del alcance del proyecto.

Pregunta 8.

Gráfico 8: Resultados de la pregunta 8



Fuente: Propia

Como se puede derivar del gráfico 8, el 80% de los consultados se considera muy de acuerdo con la afirmación "¿Considera importante contar con un módulo que le permita generar reportes del contenido de los componentes?". Un restante 20% valora estar de acuerdo con la afirmación.

Considerando las respuestas obtenidas, se ha contemplado el diseño de un módulo de reportes, que les permita a los usuarios la generación de reportes en los formatos más convenientes, de forma que puedan distribuir información de los componentes como mejor

se considere necesario. También es importante que estos reportes puedan ser personalizados en cierto grado, de forma que puedan producirse los reportes que mejor convengan.

CAPÍTULO IV: DESARROLLO

Este capítulo contiene los apartados correspondientes a la fase de desarrollo del prototipo en cuestión.

Primeramente, se presenta el apartado del análisis, el cual incluye los diagramas de uso desarrollados a partir de la información obtenida durante la recolección de datos, un análisis detallado de aspectos tales como del hardware y software requeridos, además de descripciones detalladas de aspectos, tales como los motores de bases de datos utilizadas y aspectos del personal requerido.

Se incluyen además, los apartados relacionados con el diseño del sistema, incluyendo la arquitectura del sistema, los diseños de interfaces, así como los diseños de bases de datos y procesos relevantes del proyecto.

Análisis

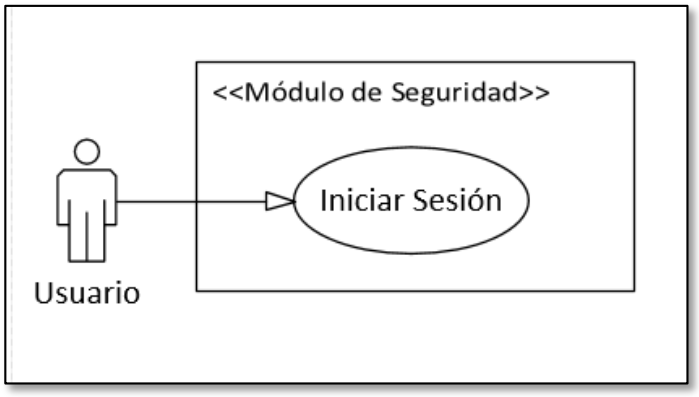
Diagramas de caso de uso.

Módulo de seguridad.

A continuación, se muestran los casos de uso más importantes, con sus respectivas tablas de descripción.

Caso de uso "Iniciar sesión".

Figura 1: Caso de uso "Iniciar Sesión".



Fuente: Propia

Cuadro 4: Descripción de caso de uso “Iniciar sesión”

Caso de uso	1. Iniciar sesión
Actor	Usuario
Módulo	Seguridad

Flujo principal:

- El usuario abre la aplicación por primera vez.
- El sistema despliega el formulario principal.
- El sistema despliega el formulario de bienvenida, el cual contiene información de la operación, el logo de la empresa y el nombre de la aplicación, y el cual se cierra automáticamente después de algunos segundos.
- El sistema despliega el formulario de inicio de sesión, el cual contiene campos de texto editables para los valores de nombre de usuario y clave.
- El usuario ingresa su usuario y clave. Y presiona el botón “iniciar sesión”.
- El sistema utiliza el módulo de seguridad para validar que la información proporcionada sea la correcta, basado en datos obtenidos de la base de datos.

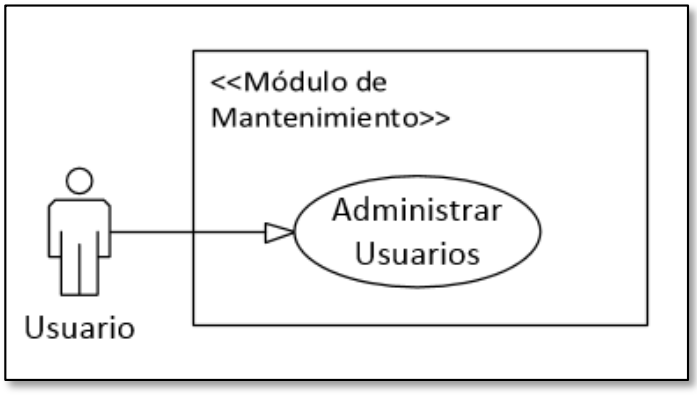
Precondiciones: El usuario abre la aplicación por primera vez.
Post condiciones: • Si la información no es adecuada se genera un mensaje de error. • Si la información es verificada exitosamente se despliega el formulario "exploración de librería de componentes". • Una vez que el usuario es autenticado, variables globales pertinentes son inicializadas, el menú del formulario principal se habilita y la librería de componentes es desplegada.

Fuente: Propia

Módulo de mantenimiento.

Caso de uso "Administrar usuarios".

Figura 2: Caso de uso "Administrar usuarios".



Fuente: Propia

Cuadro 5: Descripción de caso de uso "Administrar usuarios"

Caso de uso	2. Administrar usuarios

Actor	Usuario
Módulo	Mantenimiento

Flujo principal:

Consultar usuarios

- El usuario selecciona la opción correspondiente a “administración de usuarios” en el submenú de mantenimiento del menú principal.
- El sistema despliega el formulario de administración de usuarios. Esta ventana contiene una lista de todos los usuarios existentes, construida a partir de información obtenida del módulo de gestión de datos quien, a su vez, ha obtenido estos datos de la base de datos de la aplicación.
- El usuario selecciona un elemento de la lista
- El sistema despliega la información correspondiente a este usuario particular.

Flujos alternativos:

Editar usuario

- El usuario tiene la opción de seleccionar con el ratón un usuario de la lista.
- El sistema despliega la información correspondiente a este usuario en pantalla. Cada campo del formulario desplegará una propiedad del usuario.
- El usuario presiona el botón editar y procede a hacer los cambios que correspondan. Al finalizar presiona el botón guardar.
- El módulo de datos actualiza los cambios en la base de datos.

Crear nuevo

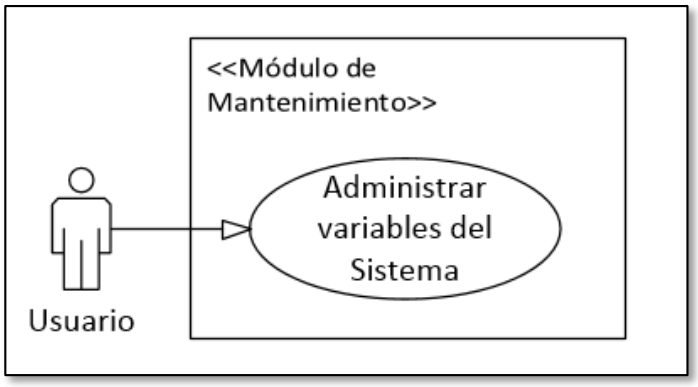
- El usuario presiona el botón “nuevo usuario”.

• El sistema habilita los campos de texto utilizados para introducir información del usuario. • El usuario completa los campos requeridos. Al presionar salvar, y si la información es adecuada, se envía al módulo de manejo de datos, el cual se encargará de insertar el nuevo usuario en la tabla correspondiente de la base de datos.
Precondiciones: Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de mantenimiento usando la opción "administración de usuarios" en el submenú de "mantenimiento" del menú principal.
Post condiciones: Si se hacen cambios a los usuarios, estos serán almacenados en la base de datos.

Fuente: Propia

Caso de uso "Administrar variables de sistema".

Figura 3: Caso de uso "Administrar variables de sistema".



Fuente: Propia

Cuadro 6: Descripción de caso de uso "Administrar variables del sistema"

Caso de uso	3. Administrar variables del sistema
Actor	Usuario
Módulo	Mantenimiento

Flujo principal:

Consultar valores

- El usuario accede al formulario “administrar variables de sistema”, el cual se encuentra en el submenú de “mantenimiento” del menú principal.
- El sistema despliega el formulario, el cual contiene campos de texto con los valores de variables globales de la aplicación. Estas incluyen opciones, tales como el directorio en el que se almacenarán los archivos XML generados durante los análisis o la ruta del directorio donde se encuentran los componentes. Así como el directorio predeterminado utilizado por la aplicación y el usuario predeterminado y su respectiva clave de acceso.

Precondiciones:

Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de “administrar variables globales”, en el submenú de mantenimiento del menú principal.

Flujos alternativos:

Editar variables del sistema

- El usuario presiona el botón de editar del formulario de administración de variables globales.
- El sistema habilita las opciones utilizadas para navegar por el sistema de archivos y seleccionar los nuevos valores de los directorios de trabajo de la aplicación y los

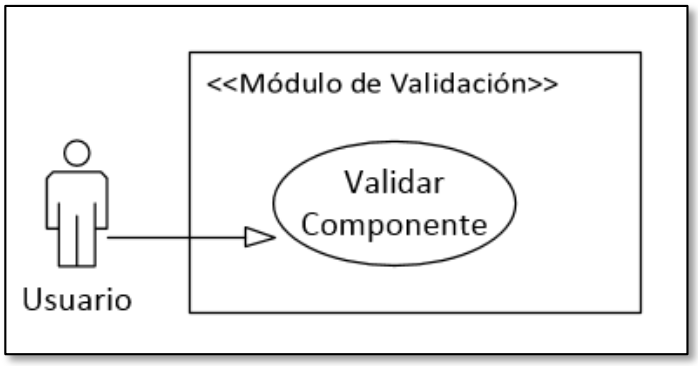
valores correspondientes al usuario y clave predeterminados. • El usuario presiona el botón guardar y los cambios son almacenados por el módulo de manejo de datos.
Post condiciones: El formulario es desplegado en pantalla mostrando la información correspondiente a las variables globales.

Fuente: Propia

Módulo de validación.

Caso de uso "Validar componente".

Figura 4: Caso de uso "Validar componente".



Fuente: Propia

Cuadro 7: Descripción de caso de uso "Validar componente"

Caso de uso	4. Validar Componente
Actor	Usuario
Módulo	Validación
Flujo principal:	

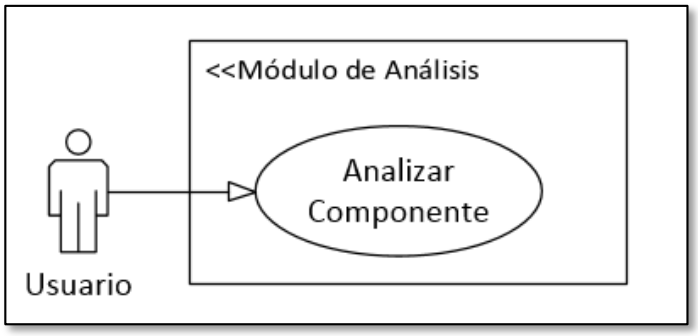
• El usuario selecciona el formulario de validación de componentes desde el menú de herramientas del menú principal. La ventana de validación se desplegará en pantalla. • El usuario utiliza el menú "validar componente" para seleccionar la ruta en disco de un componente e iniciar el proceso de validación. • El sistema procederá a explorar cada archivo contenido en el directorio, categorizará cada archivo y procederá a validar su estructura y gramática, según sea el caso. Un archivo XML sería analizado por la librería *Javax.xml.parsers.DocumentBuilder*, mientras la sintaxis de un archivo PERL sería analizada utilizando el compilador PERL, capturando las salidas correspondientes.
Precondiciones: Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de validación de componentes.
Post condiciones: Los resultados y los errores encontrados durante la validación son desplegados en pantalla.

Fuente: Propia

Módulo de análisis.

Caso de uso "Analizar componente".

Figura 5: Caso de uso "Analizar componente".



Fuente: Propia

Cuadro 8: Descripción de caso de uso "Analizar componente"

Caso de uso	5. Analizar Componentes
Actor	Usuario
Módulo	Análisis

Flujo principal:

- El usuario selecciona el formulario de análisis de componentes desde el menú de herramientas del menú principal. La ventana de análisis se desplegará en pantalla.
- El usuario utiliza la opción validar componente para seleccionar la ruta en el sistema de archivos del componente que será analizado.
- El sistema inicia el análisis de cada componente seleccionado. El análisis consiste en la validación y análisis de la estructura de archivos que constituyen el componente. Después se identifica y enlista cada una de las órdenes contenidas. A continuación, se extrae la información relevante de cada orden y se almacena en estructuras de datos diseñadas con este fin. La estructura almacena, tanto detalles de la estructura de archivos del componente, como la colección lógica de entidades que representan las órdenes y sus características. El formulario de análisis despliega cada uno de los

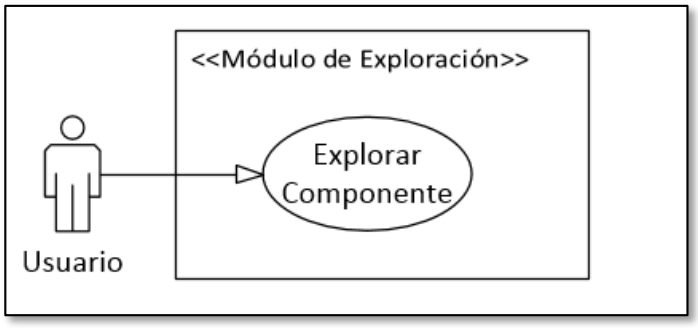
pasos ejecutados durante este proceso. • Una vez terminado el análisis de componente, la información obtenida es almacenada en una estructura de datos en memoria. Un módulo de generación de XML recibe esta estructura como parámetro y la transforma en un documento XML con una composición tal que puede, eventualmente, ser utilizada por la misma herramienta para obtener, de nuevo, los datos originales sin tener que repetir el análisis. • Si se cuenta con acceso a la base de datos, este módulo genera un registro en la tabla correspondiente para el componente analizado.
Precondiciones: Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de análisis de componentes desde el menú de herramientas.
Post condiciones: Una vez terminado el análisis de componente la información obtenida es almacenada en una estructura de datos en memoria.

Fuente: Propia

Módulo de exploración.

Caso de uso "Explorar componente".

Figura 6: Caso de uso "Explorar componente".



Fuente: Propia

Cuadro 9: Descripción de caso de uso "Explorar componente"

Caso de uso	6. Explorar componente
Actor	Usuario
Módulo	Módulo de Exploración
Flujo principal: Explorar componente • El usuario selecciona el formulario de exploración de componentes desde el menú de herramientas del menú principal. La ventana de exploración se muestra en pantalla. • El usuario utiliza la opción "leer" en el menú del formulario para seleccionar la ruta del archivo XML de componente que corresponda. • El sistema leerá el archivo XML, previamente generado por el formulario de análisis de componente, y desplegará su contenido en una ventana que facilite la exploración de las órdenes contenidas en el componente. El formulario deberá desplegar información básica del componente, tal como ruta, nombre, versión, y, además, debe desplegar estadísticas de su composición.	
Flujo alternativo:	

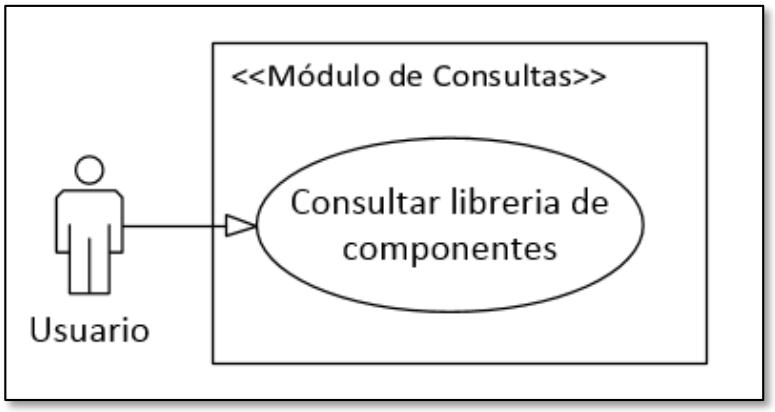
Explorar una configuración particular • Todas las configuraciones disponibles se muestran en una lista. Desde esta lista podrá desplegarse un nuevo formulario con los detalles de cada una de estas configuraciones. • Este formulario de exploración mostrará toda la información disponible de la orden. Incluyendo: información de las versiones de firmware utilizadas en la unidad, configuración del BIOS, configuración de la controladora de almacenamiento inteligente, configuración de memoria, configuración de tarjetas PCI, información de scripts de personalización, etcétera.
Precondiciones: Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de exploración de componentes que se encuentra en el submenú de herramientas.
Post condiciones: Se despliega un formulario que facilite la exploración de las órdenes contenidas en el componente. Desde esta ventana se pueden obtener los detalles de cada orden y se pueden imprimir reportes correspondientes al componente en cuestión.

Fuente: Propia

Módulo de exploración.

Caso de uso "Consultar librería de componentes".

Figura 7: Caso de uso "Consultar librería de componentes".



Fuente: Propia

Cuadro 10: Descripción de caso de uso "Consultar librería de componentes"

Caso de uso	Consultar librería de componentes
Actor	Usuario
Módulo	Consultas

Flujo principal:

- El sistema muestra, automáticamente, la librería de componentes una vez que el usuario es autenticado exitosamente por el formulario de inicio de sesión. Además, ésta puede ser accedida desde el menú de herramientas del menú principal. Esta consiste en una ventana localizada al extremo izquierdo del formulario principal. El formulario consulta el módulo de datos y obtiene los registros de los componentes disponibles en la base de datos. Estos se despliegan en un control de exploración de árbol clasificado por plataforma. Cada plataforma contiene nodos que representan las versiones de cada componente, ordenados por antigüedad (el más nuevo de primero).
- El usuario puede utilizar los controles del formulario para filtrar la lista y desplegar solamente las últimas versiones del componente. También, puede seleccionar un

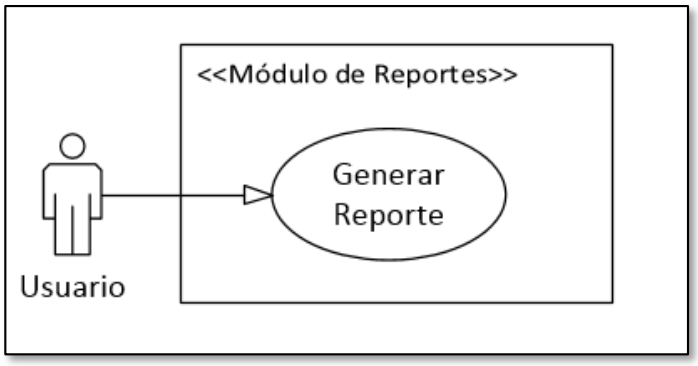
componente específico y seleccionar la opción “explorar” desde el menú. • El sistema consultará la base de datos para obtener una copia del documento de definición del componente XML, y desplegará el formulario de exploración de componente.
Precondiciones: Un usuario ha iniciado sesión en la aplicación. El formulario de librería de componentes es desplegado y muestra la información contenida en la base de datos.
Post condiciones: Una vez seleccionado el componente, se despliega un formulario de exploración con la información relevante a ese componente.

Fuente: Propia

Módulo de reportes.

Caso de uso “Generar reporte”.

Figura 8: Caso de uso “Generar reporte”.



Fuente: Propia

Cuadro 11: Descripción de caso de uso “Generar reporte”

Caso de uso	7. Generar Reporte
Actor	Usuario
Módulo	Reportes

Flujo principal:

- El usuario puede acceder a las opciones de generación de reportes desde el formulario de exploración de componentes, una vez que se haya seleccionado un componente y la información esté desplegada en el formulario.
- El sistema genera reportes basado en la información recolectada por el módulo de análisis de componente.
- El usuario tiene a su disposición herramientas para filtrar y personalizar el reporte, además de opciones para guardar el reporte generado.

Precondiciones:

Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de exploración de componentes y selecciona un componente válido.

Un usuario ha iniciado sesión en la aplicación. El usuario selecciona un componente de la librería de componentes.

Post condiciones:

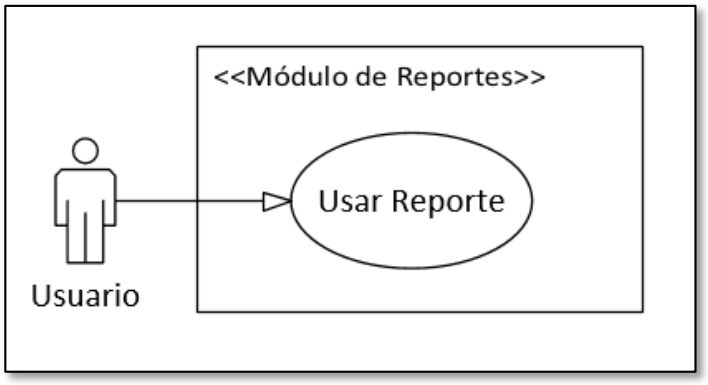
El usuario selecciona la sección de reportes y elige el formato adecuado.

El reporte es generado y se despliegan las opciones para guardar el archivo.

Fuente: Propia

Caso de uso "Usar reporte".

Figura 9: Caso de uso "Usar reporte".



Fuente: Propia

Cuadro 12: Descripción de caso de uso "Usar reporte"

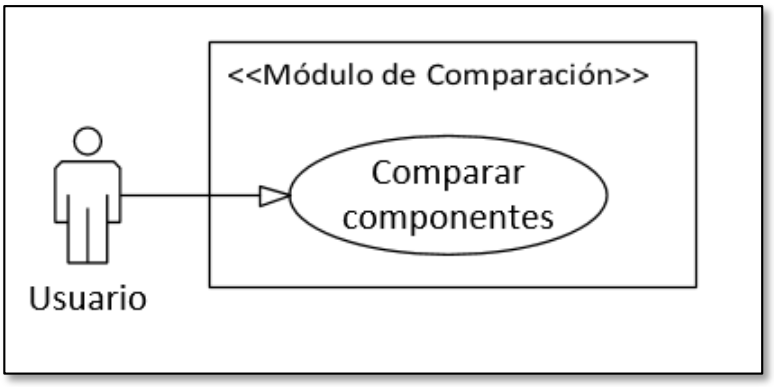
Caso de uso	8. Usar Reporte
Actor	Usuario
Módulo	Reportes
Flujo principal: • El usuario crea el reporte utilizando las opciones de generación de reportes desde el formulario de exploración de componentes, una vez que se haya seleccionado un componente y su información haya sido desplegada. • El sistema despliega un formulario con una prevista del reporte y opciones de impresión, como título y numeración de página. • El usuario puede ver, imprimir o cancelar la impresión.	
Precondiciones: Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de exploración de componentes y selecciona un componente válido. Un usuario ha iniciado sesión en la aplicación. El usuario selecciona un componente de	

la librería de componentes.
Post condiciones: El usuario selecciona la sección de reportes y elige el formato adecuado. El reporte es generado y se despliegan las opciones para guardar o imprimir el archivo.

Fuente: Propia

Caso de uso "Comparar componentes".

Figura 10: Caso de uso "Comparar componentes".



Fuente: Propia

Cuadro 13: Descripción de caso de uso "Comparar componentes"

Caso de uso	9. Comparar componentes
Actor	Usuario
Módulo	Módulo de comparación
Flujo principal: Comparar dos componentes • El usuario puede acceder al formulario de "comparación de componentes" desde la sección de "herramientas" del menú principal de la aplicación. Este consiste en un	

formulario que permite al usuario seleccionar dos componentes, correspondientes a diferentes versiones de un mismo componente. • El sistema analiza y compara los componentes, desplegando, a continuación, un formulario con los resultados de esta comparación. Esta ventana muestra la lista de órdenes que constituyen ambos componentes. Cuando la misma orden está presente en ambos componentes, ambas versiones se señalan en color negro si son idénticas o en color anaranjado si existen diferencias. • El usuario puede obtener el detalle de la comparación para inspeccionar las diferencias oprimiendo el botón explorar.
Precondiciones: Un usuario ha iniciado sesión en la aplicación. El usuario accede al menú de comparación de componentes y selecciona dos componentes válidos.
Post condiciones: Se despliega la información de ambos componentes es un formulario para su inspección.

Fuente: Propia

Análisis detallado del software desarrollado.

Formulario principal.

Consiste en una ventana de interface de múltiples documentos, la cual es desplegada en modo de pantalla completa al iniciar el programa. Este contiene un menú principal, con los submenús de “Sistema”, “Mantenimiento”, “Herramientas” y “Ayuda”. Desde estos submenús se puede acceder a todos los otros formularios de la aplicación.

“Disponibilidad sin conexión”.

Cuadro 14: “Disponibilidad sin conexión”

Número: 1
Nombre: Disponibilidad sin conexión
Descripción: La aplicación contará con un usuario predeterminado, de forma que los usuarios puedan acceder a la aplicación aunque no cuenten con conexión a la base de datos. En esta modalidad algunas funciones no estarán disponibles, pues éstas dependen de información almacenada en la base de datos. El análisis de componentes, incluyendo la capacidad de generar archivos XML de componentes y la lectura de estos, debe estar disponible aunque no se cuente con acceso a la base de datos.

Fuente: Propia

“Formulario principal y menú principal”.

Cuadro 15: “Formulario principal y menú principal”

Número: 2
Nombre: Formulario principal y menú principal
Descripción: Se diseñará un formulario principal de tipo interface de documentos múltiples, el cual contendrá un menú principal con cinco submenús: "Sistema", "Mantenimiento", "Herramientas", “Validación”, "Base de Datos", “Exploración” y "Ayuda". Al presionar cada una de las opciones contenidas en los submenús, se desplegará la ventana correspondiente, el cual estará contenido en el formulario principal.

Fuente: Propia

"Validación de datos".

Cuadro 16: "Validación de datos"

Número: 3
Nombre: Validación de datos
Descripción: Todos los formularios que requieran entrada de datos por parte del usuario deberán hacer una verificación adecuada, de forma que el usuario no pueda ingresar datos erróneos al sistema. Por ejemplo, cuando el usuario utilice el formulario de autenticar usuario, si el campo de nombre de usuario está en blanco, se desplegará el mensaje de error correspondiente.

Fuente: Propia

"Formularios internos".

Cuadro 17: "Formularios internos"

Número: 4
Nombre: Formularios internos
Descripción: Se generarán las siguientes ventanas internas: "formulario de bienvenida", "formulario de inicio de sesión", "formulario de administración de variables globales", "formulario de administración de usuarios", "formulario de validación de componentes", "formulario de análisis de componentes", "formulario de lectura de XML de componente", "librería de componentes", "formulario de comparación de componentes" y "formulario de ayuda" o "Acerca de".

Fuente: Propia

Formulario de bienvenida.

El formulario de bienvenida contiene el logo y el nombre de la empresa. No tiene controles adicionales y no interactúa con ningún otro objeto o formulario. Se despliega automáticamente por algunos segundos, antes de cerrarse automáticamente cuando la aplicación inicia, o se despliega permanentemente cuando se accede al menú de bienvenida en el submenú de Ayuda de la aplicación.

"Ventana de bienvenida".

Cuadro 18: "Ventana de bienvenida"

Número: 5
Nombre: Ventana de bienvenida
Descripción: Al iniciar la aplicación, Se presentará un formulario de bienvenida con el logo de la empresa, información de la organización y el nombre y versión de la aplicación. Esta se desplegará durante varios segundos y se cerrará automáticamente. También, podrá ser accesible desde el menú de "ayuda" "bienvenida".

Fuente: Propia

Módulo de seguridad.

Formulario de inicio de sesión.

El formulario de inicio de sesión es utilizado para permitir el uso del sistema únicamente a usuarios debidamente autenticados. Este consta de una ventana con dos cajas de texto correspondientes al nombre de usuario y a la clave de acceso. Al presionar el botón de iniciar sesión esta información es enviada al módulo de seguridad, el cual, utilizando la

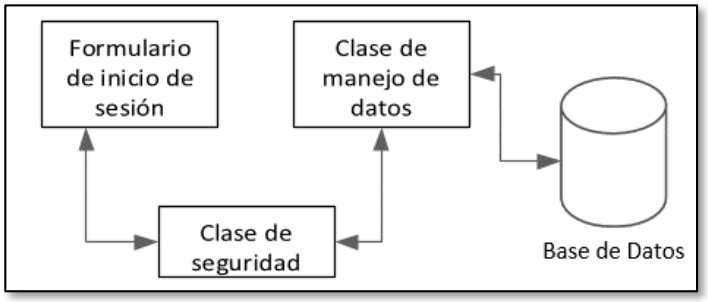
clase de manejo de datos, confirma que la información suministrada corresponda a un usuario y a una clave válida.

Este formulario se muestra automáticamente una vez que se haya desplegado la ventana de bienvenida. Además, el usuario puede acceder a este formulario desde el submenú de "sistema" utilizando la opción "ingresar".

Clase de seguridad.

La clase de seguridad es una clase utilizada por el formulario de inicio de sesión, la cual provee las funciones requeridas para validar un nombre de usuario y su respectiva clave de acceso. Además, una vez que el usuario ha sido verificado exitosamente, establece los valores globales de las variables de usuario actual, utilizadas por otros módulos de la aplicación para identificar quién es el usuario que utiliza el sistema.

Figura 11: Formulario de inicio de sesión y módulo de seguridad.



Fuente: Propia

Requerimiento "Autenticar usuario".

Cuadro 19: Requerimiento funcional "Autenticar usuario"

Número de requerimiento: 6
Nombre: Autenticar usuario

Descripción: El módulo de seguridad recibe un nombre de usuario y clave, y valida que la información suministrada corresponda a un usuario existente en la base de datos y a su clave.
Entradas: Nombre de usuario y clave.
Fuente: Datos ingresados por el usuario en el formulario de inicio de sesión.
Salida: Confirmación si el usuario y la clave son válidas.

Fuente: Propia

Requerimiento "Solo usuarios autenticados podrán utilizar la aplicación".

Cuadro 20: Requerimiento no funcional "Solo usuarios autenticados podrán utilizar la aplicación"

Número de requerimiento: 7
Nombre: Solo usuarios debidamente autenticados podrán utilizar la aplicación
Descripción: Al iniciar la aplicación, después de presentar el formulario de bienvenida, se desplegará una ventana de inicio de sesión. En este punto el menú principal estará desactivado y el usuario no podrá utilizar el sistema. Una vez que el usuario haya sido debidamente autenticado, el menú será habilitado y el usuario podrá hacer uso de la herramienta. Solo las opciones de "Sistema" – "Inicio de sesión" y la opción de "Ayuda" – "acerca de", estarán habilitadas.

Fuente: Propia

Módulo de mantenimiento.

Formulario de administración de variables globales.

El formulario de administración de variables globales es una ventana interna que contiene cuatro campos de texto, cada uno con su correspondiente botón de selección de ruta de archivo, los cuales son utilizados para desplegar los valores correspondientes a los siguientes directorios:

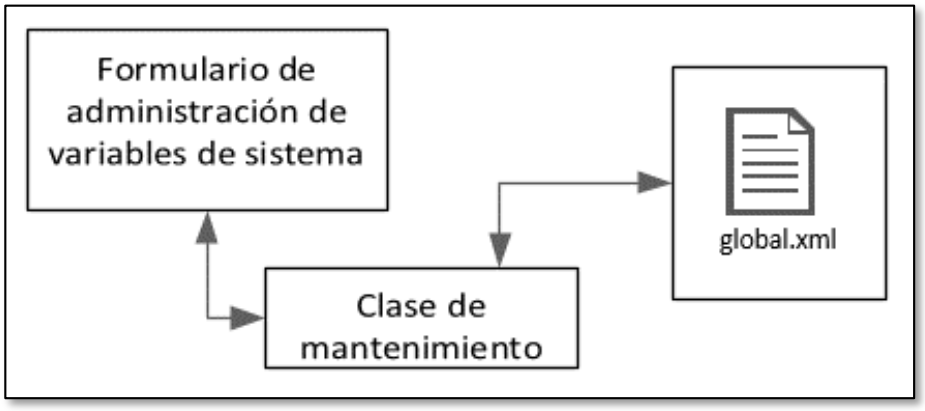
- Directorio predeterminado de la aplicación: Contiene todos los sub directorios y recursos utilizados por la aplicación.
- Directorio del Repositorio SVN: Esta ruta corresponde al directorio en disco que contiene los archivos de diagnósticos de órdenes.
- Directorio SVN de Especiales: Esta ruta corresponde al directorio en disco que contiene los archivos de diagnósticos de órdenes especiales.
- Directorio de resultados XML: Esta ruta corresponde al directorio en disco que será utilizado por la aplicación para almacenar los archivos de resultados de análisis de componente XML.

La aplicación cuenta, además, con campos para desplegar los valores actuales del usuario y de la clave de usuario predeterminados, utilizados para acceder al sistema incluso cuando no se cuenta con acceso a la base de datos.

El botón “Editar” es utilizado para habilitar el modo de edición del formulario, desde el cual es posible cambiar los valores de todos los controles mencionados anteriormente, con la posibilidad de utilizar el botón “Guardar” para enviar estos nuevos valores al módulo de mantenimiento, el cual, a su vez, los enviará al módulo de manejo de datos para su

almacenamiento en la base de datos. El botón “cancelar” puede ser utilizado para devolver el formulario a su modo original sin guardar los cambios.

Figura 12: Formulario de administración de variables globales.



Fuente: Propia

Clase de mantenimiento.

El formulario de administración de variables globales trabaja junto con la clase de mantenimiento de forma tal que los valores de las variables globales (utilizadas por todo el sistema) son almacenados en un archivo XML. La ventana de mantenimiento contiene funciones para leer y escribir variables en el archivo global.xml.

La clase de mantenimiento incluye, también, funciones utilizadas por el formulario de administración de usuarios, los cuales utilizan, a su vez, funcionalidad incluida en el módulo de manejo de datos. Estos métodos son “insertar usuario”, “actualizar usuario” y “obtener usuarios”.

Requerimiento “Obtener variables globales”.

Cuadro 21: Requerimiento funcional “Obtener variables globales”

Número de requerimiento: 8

Nombre: Obtener variables globales
Descripción: El formulario de "administración de variables globales" obtiene los valores de un conjunto de variables de sistema almacenadas en un archivo XML utilizado por el módulo de manejo de datos para este fin. La información se muestra en el formulario de administración de variables de sistema disponible en el submenú de mantenimiento.
Entradas: Ninguna.
Fuente: No aplica.
Salida: Un conjunto de objetos que contienen los valores de un conjunto de variables de sistema.

Fuente: Propia

Requerimiento "Actualizar variables globales".

Cuadro 22: Requerimiento funcional "Actualizar variables globales"

Número de requerimiento: 9
Nombre: Actualizar variables globales
Descripción: El formulario de "administración de variables globales" es utilizado para establecer o cambiar los valores de un conjunto de variables de sistema almacenadas en un archivo XML administrado por el módulo de manejo de datos. Estas variables son utilizadas para definir ciertos comportamientos de la aplicación.
Entradas: Los valores suministrados por el usuario en el formulario "administración de variables globales".
Fuente: El formulario "administración de variables globales".
Salida: Los nuevos valores son actualizados en el archivo de variables globales.

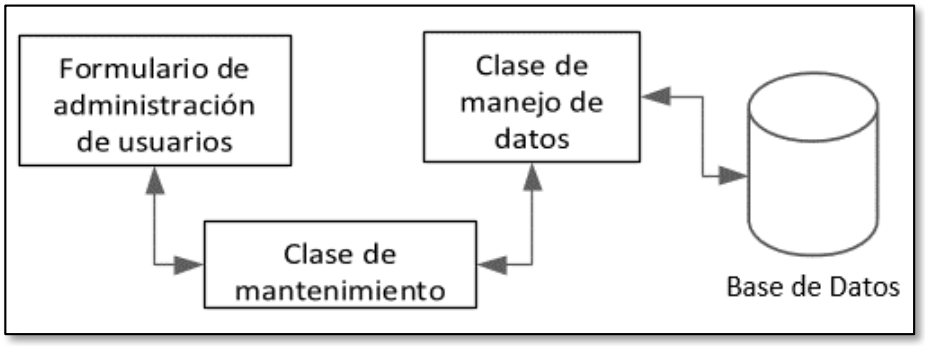
Fuente: Propia

Formulario de administración de usuarios.

El formulario de administración de usuarios muestra todos los usuarios contenidos en la base de datos. Estos se presentan en una tabla cuyas columnas corresponden al nombre, apellido, nombre de usuario y correo de cada usuario. Al seleccionar cada usuario de esta tabla, sus datos son transferidos a varios campos de texto, los cuales pueden ser editados si el usuario presiona el botón editar.

Durante la edición de un usuario, y una vez que se hayan realizado los cambios pertinentes, se puede presionar el botón "actualizar", lo cual producirá que el registro, correspondiente a dicho usuario en la base de datos, sea actualizado. Por otro lado, el usuario puede presionar el botón cancelar, lo que devolverá el formulario a su estado original, ignorando cualquier cambio hecho. Además, existe un botón "nuevo usuario", el cual puede ser utilizado para insertar un nuevo registro de usuario en la base de datos.

Figura 13: Formulario de administración de usuarios.



Fuente: Propia

Requerimiento "Consultar usuarios".

Cuadro 23: Requerimiento funcional "Consultar usuarios"

Número de requerimiento: 10
Nombre: Consultar usuarios
Descripción: El formulario de "administración de usuarios" obtiene un conjunto de objetos que contienen la información de los usuarios contenidos en la base de datos. Esta información es extraída de la base de datos por medio del módulo de manejo de datos. Los nombres de los usuarios se muestran en la lista correspondiente.
Entradas: Ninguna.
Fuente: No aplica.
Salida: Un conjunto de objetos contienen la información de los usuarios, contenidos en la base de datos, que será desplegada en el formulario "administración de usuarios".

Fuente: Propia

Requerimiento "Insertar nuevo usuario".

Cuadro 24: Requerimiento funcional "Insertar nuevo usuario"

Número de requerimiento: 11
Nombre: Insertar nuevo usuario
Descripción: El usuario utiliza el formulario "administración de usuarios" para crear un nuevo usuario. Una vez provista la información requerida, ésta es enviada al módulo de manejo de datos, el cual insertará el nuevo registro en la base de datos.
Entradas: Nombre de usuario y clave, nombre, apellido y dirección de correo del usuario.
Fuente: Datos ingresados por el usuario en el formulario de "administración de usuarios".

Salida: Confirmación de si el nuevo registro ha sido exitosamente insertado en la base de datos.

Fuente: Propia

Requerimiento "Actualizar usuario existente".

Cuadro 25: Requerimiento funcional "Actualizar usuario existente"

Número de requerimiento: 12
Nombre: Actualizar usuario existente
Descripción: El usuario utiliza el formulario de "administración de usuarios" para editar un usuario existente. Una vez provista la información requerida, ésta es enviada al módulo de manejo de datos, el cual actualizará el nuevo registro en la base de datos.
Entradas: Nombre de usuario y clave, nombre, apellido y dirección de correo del usuario.
Fuente: Datos ingresados por el usuario en el formulario de "administración de usuarios".
Salida: Confirmación si el nuevo registro ha sido exitosamente actualizado en la base de datos.

Fuente: Propia

Módulo de validación.

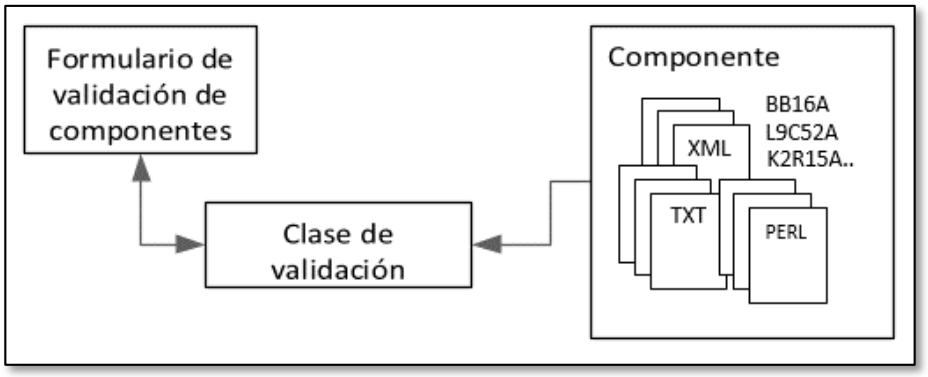
Formulario de validación de componentes.

El formulario de validación de componentes (Figura 14) puede accederse desde el submenú de herramientas del menú principal de la aplicación.

Este formulario contiene un menú interno, con el submenú “validar” en el que se encuentra el botón “componente” que despliega un buscador de archivos con el que el usuario puede seleccionar el directorio que será validado por el sistema.

El formulario de validación está dividido en dos pestañas. La primera pestaña “resultados” contiene un campo de texto “salida” donde se despliega la bitácora del proceso de validación, y un campo de texto “errores” donde se desplegaría cualquier error encontrado durante el proceso de validación de un componente. La segunda pestaña “archivos” incluye una tabla, con los campos que corresponde a: la ruta del archivo que se está analizando, el nombre del archivo, el tipo de archivo y el resultado del análisis.

Figura 14: Formulario de validación de componentes.



Fuente: Propia

Clase de validación de componentes.

La clase de validación de componentes provee las funciones requeridas para validar la estructura de los archivos XML y PERL, así como para obtener las salidas del proceso. Esta clase es utilizada exclusivamente por el formulario de validación para analizar la estructura de todos los archivos que conforman el componente, de forma que pueda garantizarse un análisis y funcionamiento exitosos.

Requerimiento "Validación de un componente".

Cuadro 26: Requerimiento funcional "Validación de un componente"

Número de requerimiento: 13
Nombre: Validación de un componente
Descripción: Un componente de validación recibe la ruta en disco donde se localiza un directorio de componente. Este proceso hace un recorrido recursivo de los archivos contenidos en el directorio, identificando cuáles corresponden a archivos conformados utilizando el lenguaje XML y cuáles archivos corresponden a scripts en el lenguaje PERL. Cada uno de estos archivos es analizado por otro proceso de validación, el cual identifica si la estructura sintáctica de estos está libre de errores.
Entradas: La ruta en disco donde se localiza un directorio de componente.
Fuente: La ruta es obtenida gracias a un selector de archivo disponible en el formulario de validación de componentes, el cual puede ser utilizado por el usuario para navegar hasta el directorio del componente y seleccionarlo para su validación.
Salida: La salida consiste en el resultado del examen de cada archivo. Y los errores generados cuando corresponda.

Fuente: Propia

Módulo de análisis.

Formulario de análisis de componentes.

El formulario de análisis de componente (Figura 18) puede ser accedido utilizando el botón "analizar componente" del submenú "herramientas" del menú principal.

El formulario tiene su propio menú, el cual incluye la opción de “análisis” que ofrece las opciones de analizar un solo componente o de hacer un análisis en lote de todas las versiones de un producto (o plataforma) determinado. También, existe la opción de hacer un análisis de todas las versiones contenidas en el repositorio de componentes. Cada opción despliega una ventana de exploración de archivos con el cual el usuario puede seleccionar la ruta del directorio correspondiente.

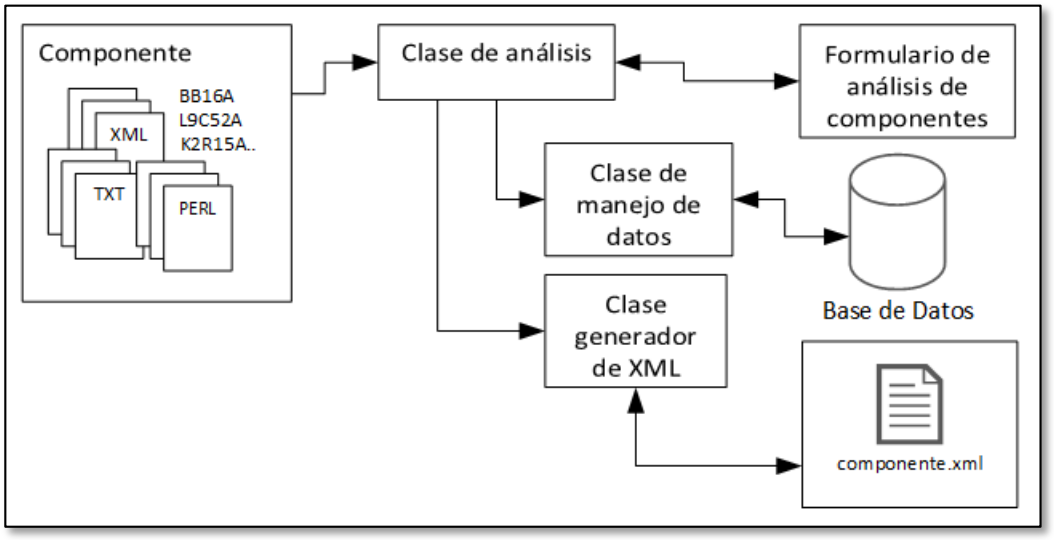
El formulario consiste de cuatro pestañas, correspondientes a “información general”, “salida”, “XML” y “procesamiento por lotes”. La ventana de “información general” incluye campos de texto para mostrar la ruta en disco del directorio del componente analizado, el nombre, la plataforma a la que corresponde y el número de versión, así como información estadística de la composición lógica del componente. Toda esta información es obtenida durante el análisis del componente.

La pestaña “salida” muestra dos campos de texto, correspondientes a la salida del análisis, el cual desplegará todos los mensajes generados durante el análisis de un documento, así como otro campo de texto correspondiente a los errores y advertencias encontradas durante el análisis.

La pestaña XML contiene un campo de texto con la salida que corresponde al documento XML generado después del análisis. Esta estructura puede ser almacenada en el sistema de archivos, utilizando el botón “salvar archivo”. Este archivo puede ser utilizado posteriormente por el formulario de lectura de XML para recuperar toda información obtenida del análisis.

Durante la definición del alcance de este módulo, se mencionó que éste dispondría de opciones para filtrar los resultados obtenidos usando distintos criterios. Estas opciones han

sido colocadas en el módulo de exploración de componente, el cual es utilizado para explorar los componentes obtenidos, donde parece ser más conveniente que se encuentre esta función.



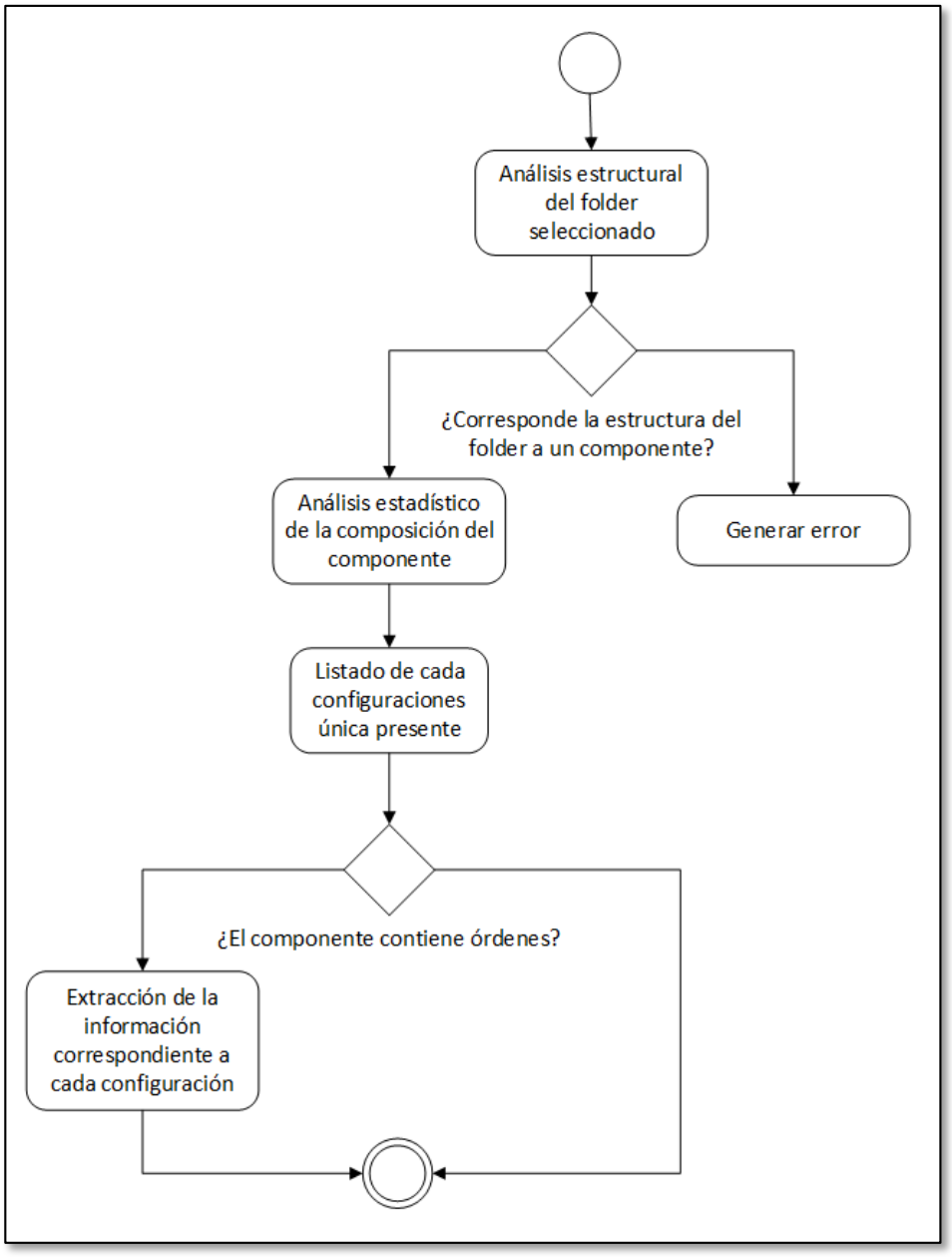
Fuente: Propia

Clase de análisis de componentes.

La clase de análisis de componente es la encargada de analizar la estructura del componente, extraer la información relacionada con las órdenes que contiene y almacenarla en las estructuras de datos especialmente diseñadas para este fin.

La clase devolvería al formulario una estructura de datos conteniendo toda la información correspondiente al componente analizado. En el caso correspondiente al análisis por lotes, la clase de análisis de componentes insertaría la información obtenida directamente en la base de datos. Creando, además, los respectivos archivos XML en el directorio predeterminado utilizado para este fin por la aplicación.

Figura 15: Diagrama básico de flujo del proceso de análisis de componente.



Fuente: Propia

Clase generador de XML.

La clase "generador de XML", recibe como parámetro una estructura de datos proveniente, por lo general, de la clase de análisis de componentes. Es a partir de la información contenida en esta estructura, que el generador crea el documento XML

correspondiente al componente original. Este documento XML de componente puede ser utilizado por otros módulos para recuperar la información obtenida durante el análisis. Además, contiene las funciones requeridas para almacenar el documento XML en el sistema de archivos.

Requerimiento "Análisis de un componente".

Cuadro 27: Requerimiento funcional "Análisis de un componente"

Número de requerimiento: 14
Nombre: Análisis de un componente
Descripción: Un módulo de análisis recibe la ruta en disco donde se localiza un directorio de componente. Este proceso recorre todos los directorios y archivos que constituyen el componente y extrae la información relevante a las órdenes. El componente de análisis retorna una estructura de datos que representa toda la información obtenida.
Entradas: La ruta en disco donde se localiza un directorio de componente.
Fuente: La ruta es obtenida gracias a un selector de archivo disponible en el formulario de validación de componentes, el cual puede ser utilizado por el usuario para navegar hasta el directorio del componente y seleccionarlo para su validación.
Salida: La salida consiste en el resultado del examen de cada archivo. Y los errores generados cuando corresponda.

Fuente: Propia

Requerimiento "Almacenamiento de componente analizado".

Cuadro 28: Requerimiento funcional "Almacenamiento de componente analizado"

Número de requerimiento: 15
Nombre: Almacenamiento de componente analizado
Descripción: Cada vez que un componente es examinado por el módulo de análisis de componente, este va a insertar un registro en la base de datos con toda la información relevante de dicho componente, utilizando el módulo de manejo de datos. Esta información incluye el archivo XML generado en el requerimiento "Generación del XML de un componente"
Entradas: La ruta en disco donde se localiza un directorio de componente.
Fuente: La ruta es obtenida gracias a un selector de archivo disponible en el formulario de validación de componentes, el cual puede ser utilizado por el usuario para navegar hasta el directorio del componente y seleccionarlo para su validación.
Salida: La salida consiste en el resultado del examen de cada archivo. Y los errores generados cuando corresponda.

Fuente: Propia

Requerimiento "Generación del XML de un componente".

Cuadro 29: Requerimiento funcional "Generación del XML de un componente"

Número de requerimiento: 16
Nombre: Generación del XML de un componente
Descripción: Un módulo generador de XML recibe una estructura de datos de componente como parámetro. Este analiza la estructura y genera un documento XML con todos los datos de dicho componente. Este archivo puede ser almacenado en el sistema de archivos de la computadora o en la base de datos, y puede, posteriormente,

ser leído por la aplicación para generar la estructura de datos original.
Entradas: Una estructura de datos de componente generada por el módulo de "análisis de componente.
Fuente: La estructura de datos de componente se genera en el formulario de análisis de sistema cuando el usuario escoge un componente para ser analizado.
Salida: La salida consiste en un archivo XML con una estructura determinada que contiene toda la información de la estructura de datos en cuestión.

Fuente: Propia

Requerimiento "Análisis de componentes en lote".

Cuadro 30: Requerimiento no funcional "Análisis de componentes en lote"

Número de requerimiento: 17
Nombre: Análisis de componentes en lote
Descripción: El módulo de análisis de componente contará con opciones para analizar un conjunto de componentes a la vez. Al seleccionar el directorio de una plataforma, éste podrá navegar por cada uno de los directorios de las distintas versiones de componentes y extraer la información requerida, transformarla y almacenarla en la base de datos, sin ninguna interacción por parte del usuario, una vez que haya iniciado el análisis.

Fuente: Propia

Módulo de exploración.

Formulario de exploración de componente.

El formulario de exploración de componente puede accederse desde el submenú de herramientas del menú principal. Este consiste en una ventana, con su respectivo menú, y

con cuatro pestañas correspondientes a las secciones “información general”, “reportes”, “XML” y “estructura de archivos”.

El menú del formulario incluye el botón “leer”, el cual, al ser presionado, despliega una ventana de exploración de archivos que puede ser utilizada por el usuario para seleccionar un archivo XML de componente que será procesado por el módulo de lectura. Este módulo puede generar la estructura de datos que utilizará el formulario para que el usuario pueda explorar el contenido del componente.

La pestaña “información general” incluye campos de texto utilizados para mostrar la ruta en disco del directorio del componente analizado, el nombre del componente, la plataforma a la que corresponde y el número de versión, así como información estadística de la composición lógica del componente. Además, esta pestaña incluye una lista de las configuraciones disponibles, desde la cual se puede seleccionar una configuración y presionar el botón explorar. A continuación, se desplegará un formulario de exploración de configuración (ver formulario de exploración de configuración a continuación).

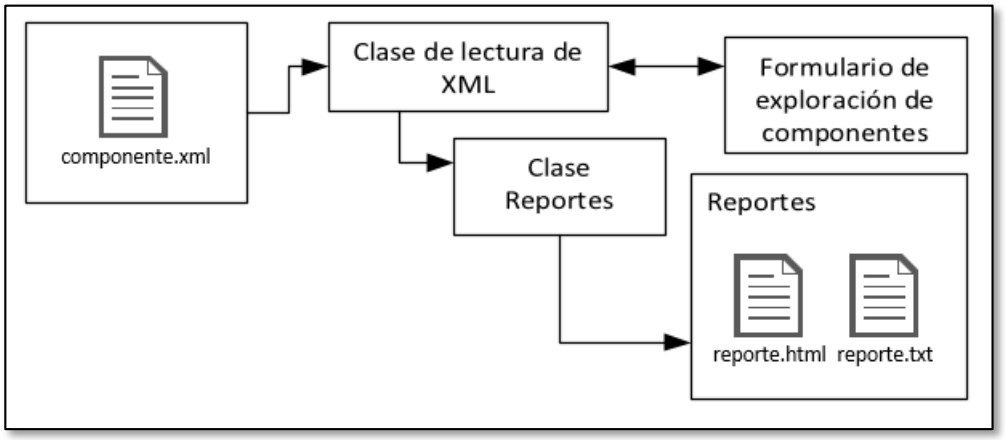
La pestaña “reportes”, ofrece una vista previa de los reportes tanto en formato HTML como en formato de texto, con opciones para mostrar solamente las configuraciones de un programa determinado y para visualizar una configuración específica.

La pestaña XML ofrece la estructura del documento leído por el formulario, y las opciones para guardar el documento en el sistema de archivos del sistema operativo.

Finalmente, la pestaña de “estructura de archivos” presenta una lista con todas las configuraciones disponibles en el componente. Al seleccionar cada una de estas configuraciones, otra lista despliega las rutas de los archivos relacionados a dicha

configuración, los cuales pueden ser accedidos por el usuario usando el botón “abrir archivo”.

Figura 16: Formulario de lectura de XML de componente.



Fuente: Propia

Clase de lectura de XML.

Esta clase contiene la funcionalidad requerida para recuperar la información que ha sido almacenada en los archivos XML de componente por la clase “generador de XML”. Esta recibe, ya sea la ruta de un archivo XML o el contenido en memoria de un documento XML cuya estructura corresponda a la establecida para contener la definición de un componente.

La información extraída es entonces almacenada en una estructura de datos en memoria que es utilizada por el formulario de exploración para desplegar su contenido en la ventana de exploración de componente.

Requerimiento “Exploración de componente”.

Cuadro 31: Requerimiento funcional “Exploración de componente”

Número de requerimiento: 18
Nombre: Lectura de archivo XML de componente
Descripción: Un módulo de lectura recibe un archivo XML de componente como parámetro. A partir de la información contenida en el archivo el módulo puede generar una estructura de datos con toda la información de ese componente. Esta información, por lo general, será presentada en un formulario de exploración de componente, el cual tiene funciones para mostrar la información disponible de la mejor manera.
Entradas: Un archivo XML de componente.
Fuente: El archivo puede provenir de la base de datos o puede provenir del sistema de archivos.
Salida: La información se presenta en un formulario de exploración de componente.

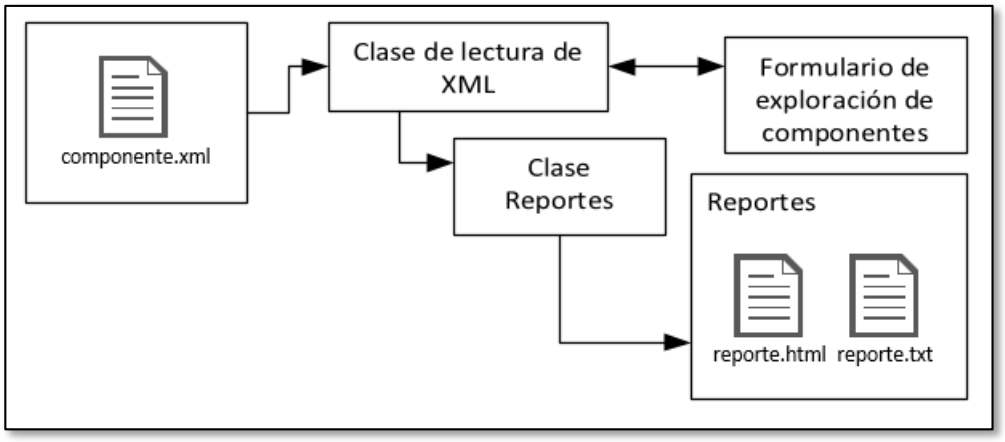
Fuente: Propia

Módulo de reportes.

La clase de reportes está encargada de generar informes escritos, en los formatos correspondientes, cada vez que un componente es visualizado en el formulario de exploración. El usuario puede acceder a este módulo desde el formulario de exploración de componentes, el cual, tiene una pestaña "reportes" la cual incluye una prevista de los informes que pueden ser generados por el sistema.

Esta clase incluye funciones para filtrar configuraciones específicas, y para modificar la estructura del reporte, según convenga.

Figura 17: Clase de reportes.



Fuente: Propia

Requerimiento "Generación de reportes de un componente".

Cuadro 32: Requerimiento funcional "Generación de reportes de un componente"

Número de requerimiento: 19
Nombre: Generación de reportes de un componente
Descripción: Un componente de generador de reportes recibe una estructura de datos de componente como parámetro. Este componente analiza la estructura de datos y genera una salida HTML o de texto con toda la información del componente en cuestión. Este reporte puede guardarse en el sistema de archivos.
Entradas: Una estructura de datos de componente generada por el módulo de "análisis de componente.
Fuente: La estructura de datos de componente se genera en el formulario de análisis de sistema cuando el usuario escoge un componente para ser analizado.
Salida: La salida consiste en un archivo HTML o en texto plano con la información requerida por el usuario.

Fuente: Propia

Módulo de consultas.

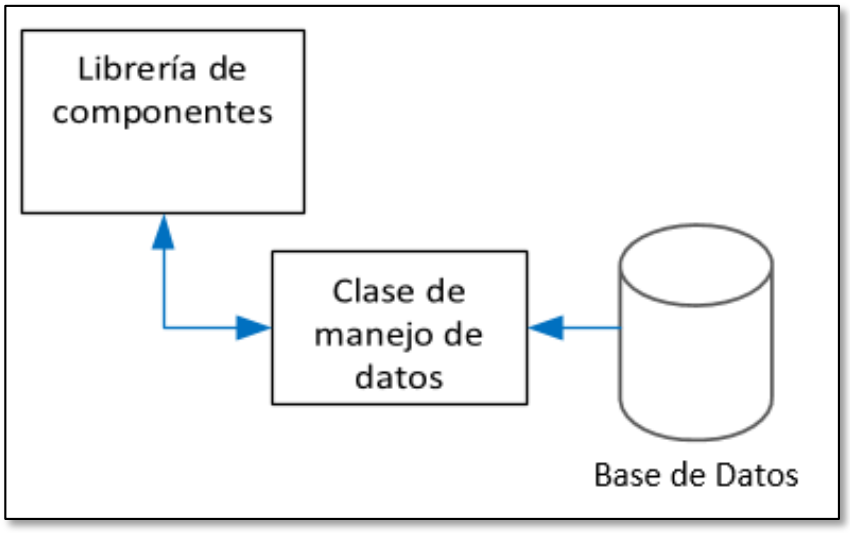
Librería de componentes.

La librería de componentes es utilizada por el usuario para consultar componentes que han sido analizados previamente, y cuyos resultados han sido almacenados en la base de datos. Este formulario clasifica los componentes por plataforma y número de versión. El usuario puede visualizar cualquier componente y configuración previamente analizado desde esta librería.

Esta librería consiste en un formulario, el cual se muestra en la parte izquierda de la interface principal y que contiene dos secciones, estas despliegan información acerca de los componentes disponibles de forma diferente.

La primera sección consiste en una vista de árbol, en la cual los componentes son agrupados de acuerdo con la plataforma o modelo del servidor a la que corresponden. Existe la opción de desplegar las últimas cinco versiones disponibles del componente o todas las versiones disponibles. La segunda sección despliega solamente la última versión disponible del componente de cada plataforma, y enlista las configuraciones disponibles en cada una de estas versiones. En esta sección el usuario puede hacer una búsqueda utilizando el nombre de la configuración.

Figura 18: Formulario de librería de componentes.



Fuente: Propia

Requerimiento "Despliegue de la librería de componentes".

Cuadro 33: Requerimiento funcional "Lectura de archivo XML de componente"

Número de requerimiento: 20
Nombre: Lectura de archivo XML de componente
Descripción: Un "componente de lectura de XML de componente" recibe un archivo XML de componente como parámetro. A partir de la información contenida en el archivo el componente puede generar una estructura de datos XML con toda la información de ese componente. Esta información, por lo general, será presentada en un formulario de exploración de componente, el cual tiene funciones para presentar de la mejor manera la información disponible.
Entradas: Un archivo XML de componente.
Fuente: El archivo puede provenir de la base de datos o puede provenir del sistema de archivos.

Salida: La información se presenta en un formulario de exploración de componente.

Fuente: Propia

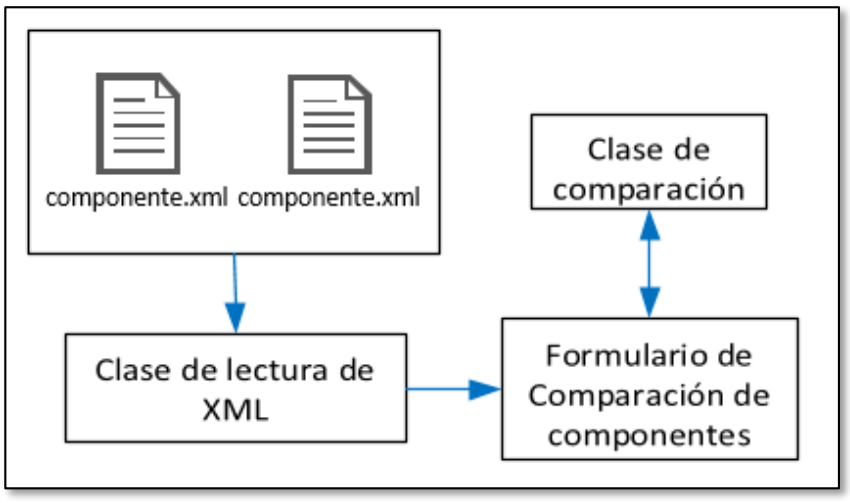
Módulo de comparación.

Formulario de comparación de componentes.

Este formulario de comparación de componentes puede ser accedido desde el submenú de herramientas en el menú principal de la aplicación. Puede ser utilizado para seleccionar dos versiones diferentes de un componente correspondiente a la misma plataforma y comparar su contenido.

Una vez seleccionados los componentes, se desplegarán dos listas. Cada una con las configuraciones correspondientes a cada componente, de forma que pueda observarse si las dos versiones de la misma configuración son idénticas o si las versiones son distintas. Desde este formulario el usuario podrá acceder a los detalles de las diferencias entre ambas versiones de la misma orden o configuración.

Figura 19: Formulario de comparación de componentes.



Fuente: Propia

Requerimiento "Comparación de dos componentes".

Cuadro 34: Requerimiento funcional "Comparación de dos componentes"

Número de requerimiento: 21
Nombre: Comparación de dos componentes
Descripción: Un "componente de lectura de XML de componente" recibe dos archivos XML de componente como parámetro. A partir de la información contenida en el archivo el componente puede generar las estructuras de datos XML correspondiente a cada componente. Esta información será enviada a un módulo de comparación que procederá a analizar los componentes, desplegando, a continuación, un formulario con los resultados de esta comparación. Esta ventana muestra la lista de órdenes que conforman ambos componentes. Cuando la misma orden está presente en ambos componentes, éstas se señalan en color azul si son idénticas o en color rojo si existen diferencias.
Entradas: Dos archivos XML de componente.
Fuente: Los archivos pueden provenir de la base de datos o pueden provenir del sistema de archivos.
Salida: La información se presenta en un formulario de comparación de componentes.

Fuente: Propia

Formulario de ayuda.

Requerimiento "Formulario de ayuda".

Cuadro 35: Requerimiento no funcional "Formulario de ayuda"

Número de requerimiento: 22

Nombre: Formulario de ayuda
Descripción: En el formulario "acerca de" se encontrará información relacionada con la aplicación, a su versión, etcétera. Además de algún logo del proyecto.

Fuente: Propia

Análisis detallado del hardware requerido.

Durante el desarrollo de la aplicación se utilizó una laptop HP ProBook 6470b. Esta computadora cuenta con un procesador Intel Core i5-3320M, con una velocidad de 2.60 GHz. Cuenta con 8,00 GB de memoria y tiene una versión de Windows 7 Enterprise de 64-bits.

Para todos los efectos, el hardware mínimo requerido para el uso de la aplicación y los requerimientos recomendados serían los siguientes:

Requerimientos mínimos.

- Microsoft Windows Vista SP1/Windows 7 Professional:
- Procesador: 800MHz Intel Pentium III o equivalente
- Memoria: 512 MB
- Espacio en disco: 750 MB de espacio libre en disco

Requerimientos recomendados.

- Microsoft Windows 7 Professional/Windows 8/Windows 8.1
- Procesador: Intel Core i5 o equivalente
- Memoria: 2 GB (32-bit), 4 GB (64-bit)
- Espacio en disco: 1.5 GB de espacio libre en disco.

Análisis detallado de los elementos relacionados con las telecomunicaciones.

Los sistemas operativos recomendados con anterioridad, tanto Microsoft Windows Vista SP1 como Windows 7 Professional, incluyen el software de red integrado para utilizar el motor de base de datos a través de TCP/IP de ser requerido. Las conexiones a un servidor remoto de base de datos requerirán del protocolo de TCP/IP. Los clientes pueden acceder al servidor por medio del puerto 3306.

Descripción detallada de base de datos.

Para la elaboración del prototipo, se utilizó el software de gestión bases de datos "MySQL Community Server", versión 5.7.18.1, disponible bajo licencia pública general de GNU, la cual garantiza al usuario final la libertad de usar el software. Se utilizó la versión correspondiente a la arquitectura de 64 bits.

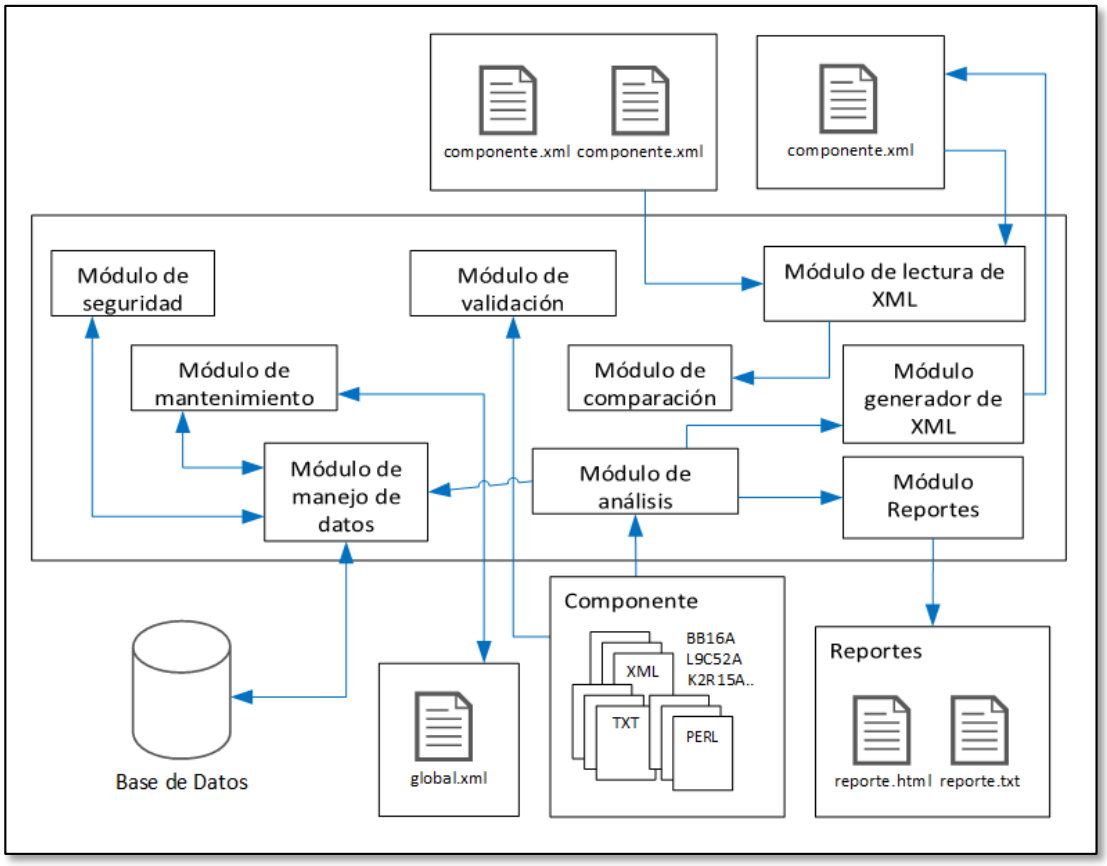
Descripción detallada del personal requerido.

Las responsabilidades de los ingenieros en automatización incluyen el diseño y soporte de soluciones de diagnósticos utilizadas por varias fábricas involucradas en la producción de servidores de rack y servidores de torre. Estas pruebas son utilizadas, entre otras cosas, para verificar la configuración de hardware, automatizar la instalación de firmware y sistemas operativos y el diagnóstico de los equipos durante el proceso de ensamblaje. Además, los ingenieros están en constante comunicación con las fábricas involucradas, brindando soporte y resolviendo problemas técnicos relacionados con los distintos componentes de diagnóstico.

Diseño

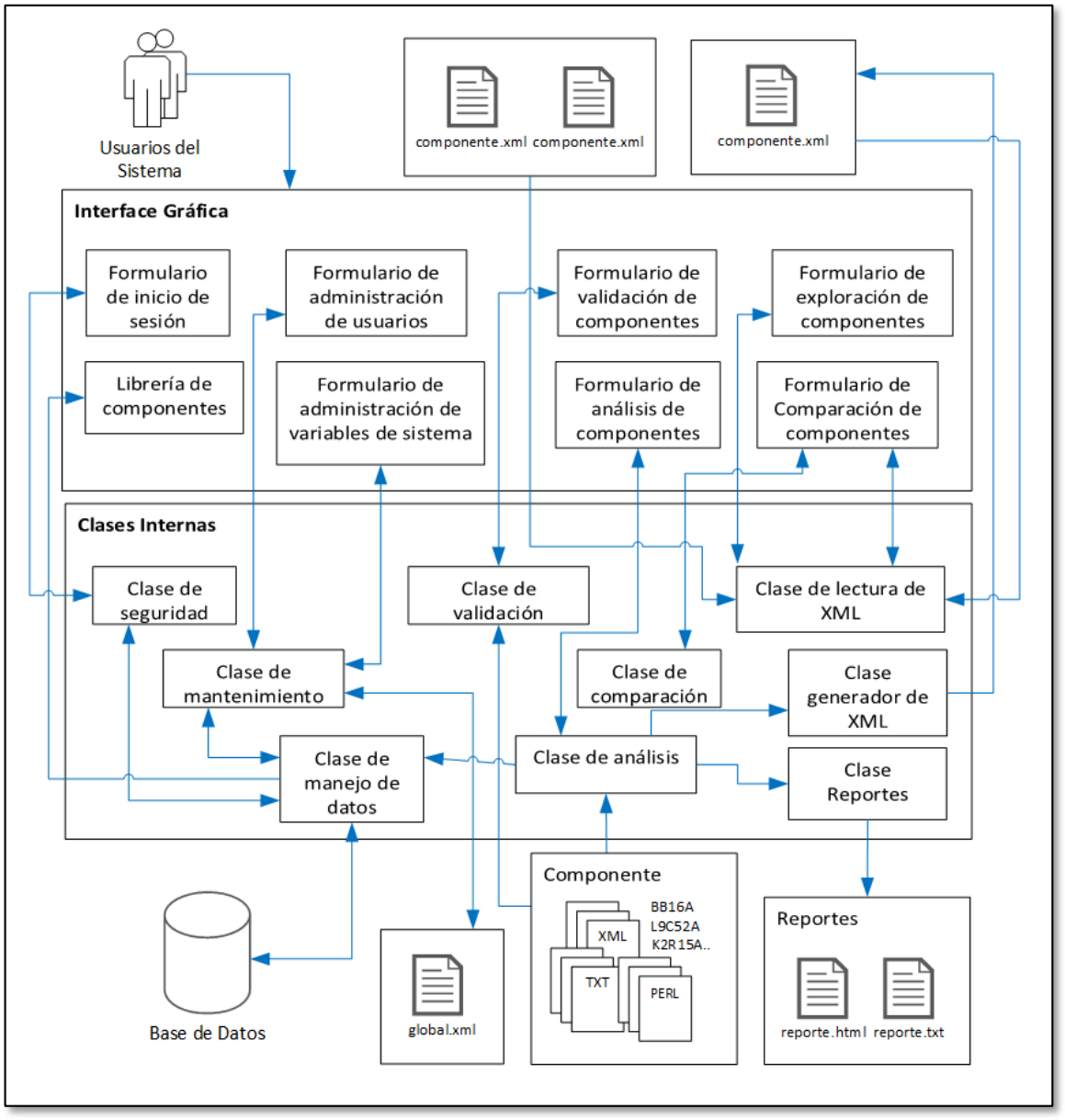
Diagrama de arquitectura del software.

Figura 20: Diagrama de la arquitectura global del software.



Fuente: Propia

Figura 21: Detalle de la arquitectura global del software.



Fuente: Propia

Prototipo a nivel general dentro de la infraestructura del cliente.

La aplicación se enfoca en los componentes de software utilizados durante la manufactura de equipos de cómputo, específicamente, servidores de estándar industrial. Estos componentes son utilizados por un sistema de automatización que, a grandes rasgos,

se encarga de ciertos aspectos de la configuración de los servidores y además, de verificar que su configuración final corresponda a la esperada por los clientes.

Documento de intención de cliente

Esta configuración está plasmada en un documento que contiene las especificaciones de las órdenes, tanto a nivel de hardware, como el software. Este indica la cantidad de unidades que deben ser fabricadas, y cada característica de la unidad está plasmada en este documento, llamado documento de intención de cliente. Es de suma importancia que el equipo este configurado exactamente como ha sido acordado con el cliente.

En la figura 22, se muestra un ejemplo de la especificación de hardware de una orden. Esta lista contiene una columna con los números de parte, los cuales identifican exclusivamente cada tipo de parte distinta, junto a una columna con la descripción correspondiente.

Figura 22: Ejemplo de la especificación de hardware de una orden.

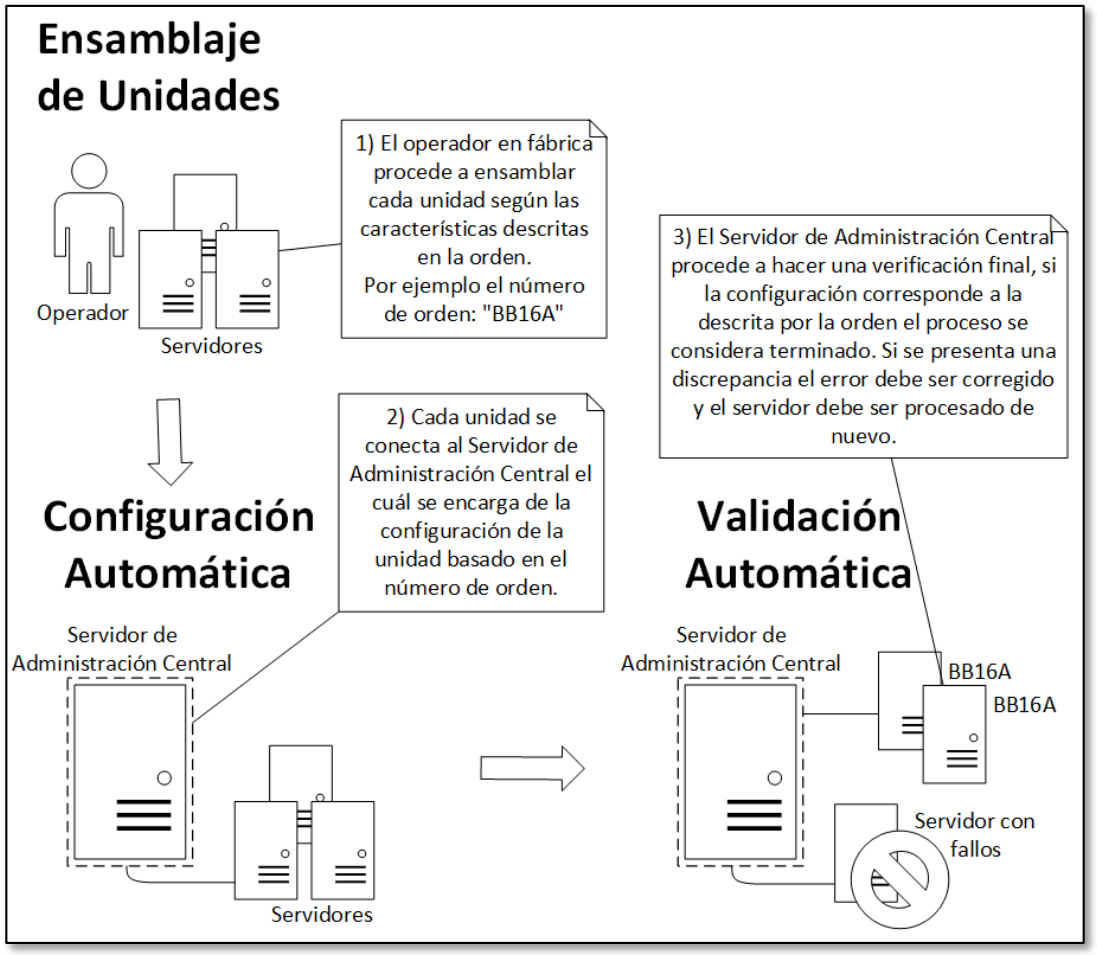
HP Model X and Option Kits	Description
754536-B21	HP ML350T09 SFF CTO Server
801232-L21	HPE ML350 Gen9 E5-2620v4 FIO Kit
805347-B21	HPE 8GB 1Rx8 PC4-2400T-R Kit
726536-B21	HP 9.5mm SATA DVD-ROM Jb Gen9 Kit
785067-B21	HP 300GB 12G SAS 10K 2.5in SC ENT HDD
749974-B21	HP Smart Array P440ar/2G FIO Controller
725878-B21	HP ML350 Gen9 Redundant Fan Kit
720478-B21	HPE 500W FS Plat Ht Plg Pwr Supply Kit
726545-B21	HP ML350 Gen9 SFF Media Cage Kit
765652-B21	HP ML350 Gen9 AROC cable kit
512485-B21	HPE iLO Adv incl 1yr TSU 1-Svr Lic

Fuente: Centro Global de Ingeniería.

Proceso de manufactura de servidores

La siguiente figura muestra, a grandes rasgos, el proceso de manufactura de servidores en fábrica. Un operario se encarga de ensamblar físicamente la unidad, utilizando los componentes de hardware adecuados, según ha sido requerido por el cliente. A continuación, el servidor es conectado a una red de pruebas, por medio de la cual un servidor de administración central toma control de la unidad para ejecutar un número de procesos encargados de su configuración. Los componentes de software son quienes contienen la información utilizada por este servidor para procesar la unidad. Finalmente, el mismo proceso automático se encarga de validar o verificar que la configuración final del servidor corresponda a lo establecido.

Figura 23: Manufactura de servidores.



Fuente: Propia

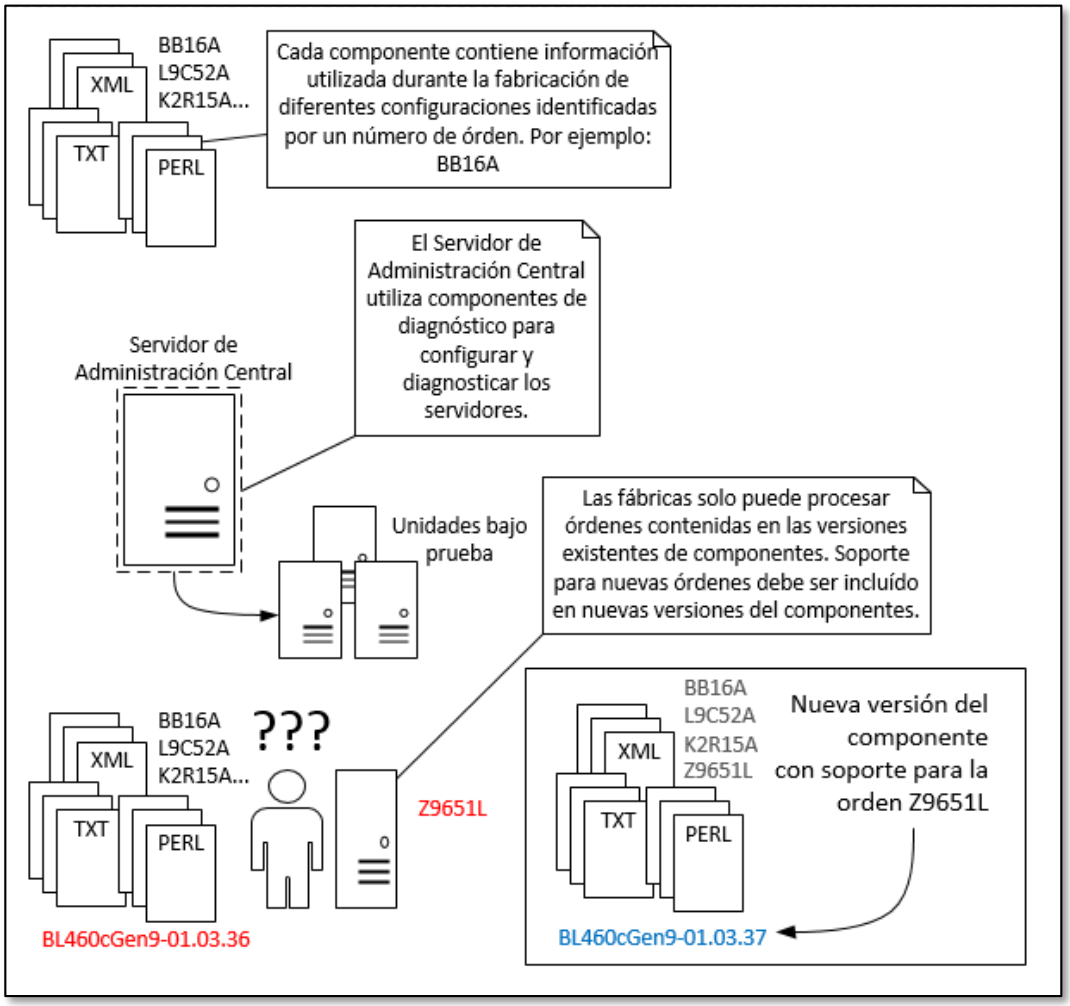
Componentes de diagnóstico

Son precisamente los componentes de diagnóstico quienes contienen la información que es utilizada por el servidor de administración central durante la configuración de los servidores.

Las fábricas tienen que procesar órdenes con configuraciones que ya han sido utilizadas en el pasado, pero también, órdenes nuevas, con combinaciones de características y requerimientos no utilizados con anterioridad. Para que estas órdenes puedan ser

procesadas por el sistema de automatización de fábrica, se deben generar nuevas versiones de los componentes, que incluyan la información necesaria para procesar las órdenes nuevas.

Figura 24: Componentes de diagnóstico.



Fuente: Propia

Diseño de Interfaces.

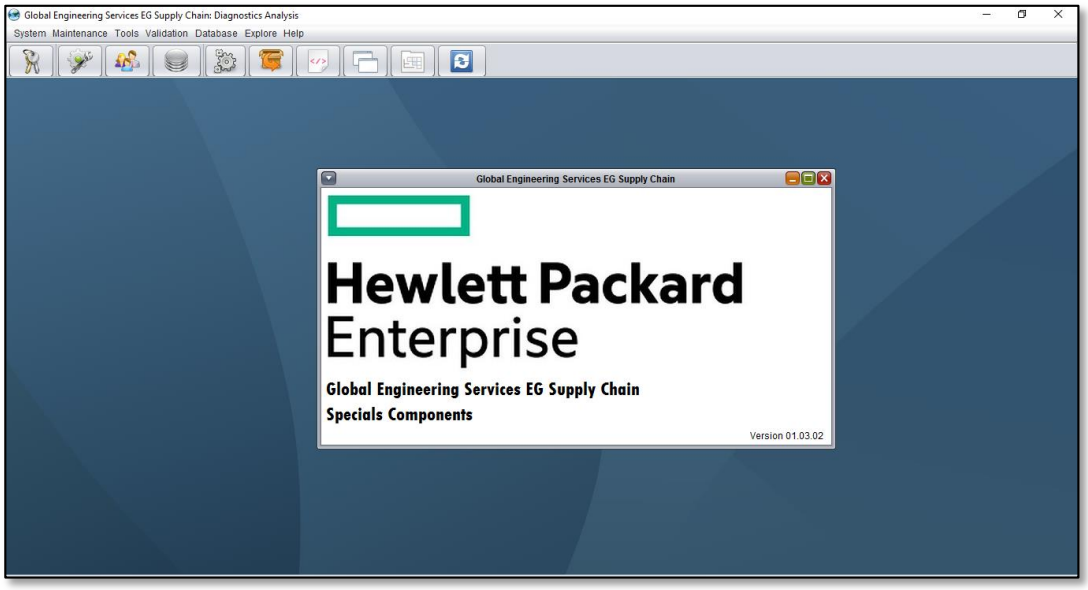
Gráficas

Formulario principal y formulario de bienvenida.

El formulario principal consiste en una interface de documento múltiple. Esto quiere decir que todas las otras ventanas de la aplicación estén contenidos dentro de este formulario. Este será desplegado en modo de pantalla completa de forma predeterminada, con opciones para cambiar su tamaño, según convenga.

El formulario de bienvenida se despliega automáticamente por algunos segundos, al iniciar la aplicación. Esta ventana tiene el logo de la empresa e información relacionada con la operación y la versión de la aplicación.

Figura 25: Formulario principal y formulario de bienvenida.

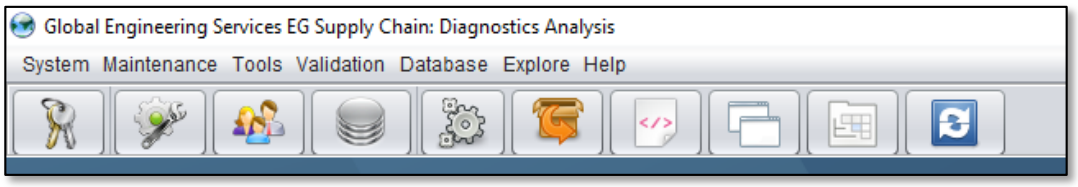


Fuente: Propia

Barra de menú y barra de herramientas.

El menú principal contiene una barra de menú y una barra de herramientas las cuales pueden ser utilizadas por los usuarios para acceder a todas las herramientas disponibles en el sistema. Estas estarán deshabilitados hasta que el usuario inicie una sesión en el sistema.

Figura 26: Barra de menú y barra de herramientas.



Fuente: Propia

Formulario de inicio de sesión.

El formulario de inicio de sesión es utilizado por los usuarios para acceder al sistema, utilizando un nombre de usuario y clave válidos. Esta ventana puede ser utilizada, además, para iniciar otra sesión con un usuario diferente.

Figura 27: Formulario de inicio de sesión.

Fuente: Propia

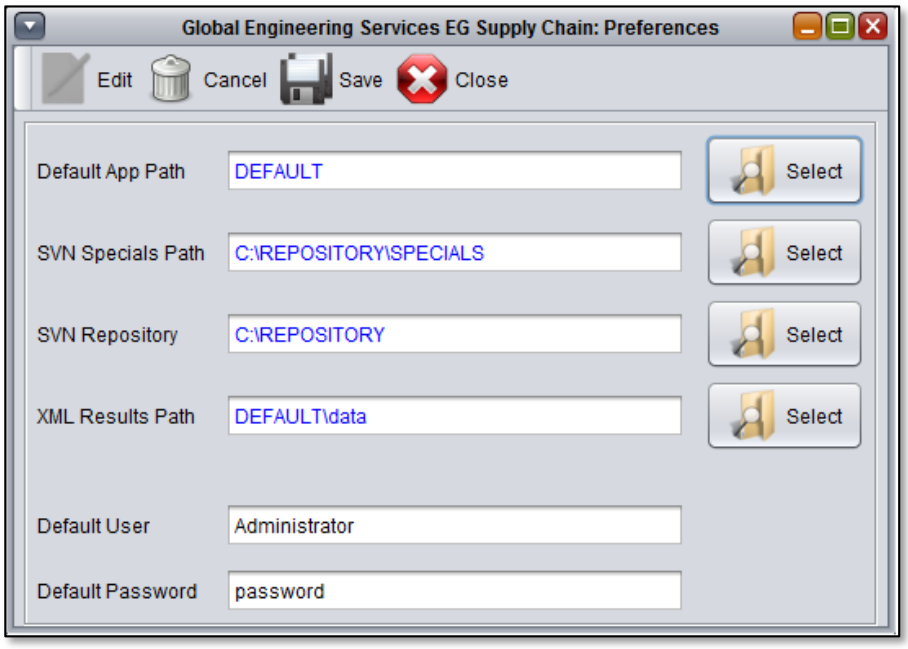
Formulario de administración de variables globales.

El formulario de administración de variables globales es utilizado para configurar algunos aspectos del sistema.

Entre estos aspectos se encuentra el directorio en el cual la aplicación y sus recursos son almacenados. También, se pueden configurar las rutas de dos directorios utilizados por el sistema de control de versión para almacenar el código fuente de los componentes.

Otra opción corresponde al directorio de trabajo, en el cual se almacenarán los archivos XML que contienen la información obtenida por el formulario de análisis de componente. Un último aspecto corresponde a la configuración el usuario y clave predeterminados de la aplicación.

Figura 28: Formulario de administración de variables globales.

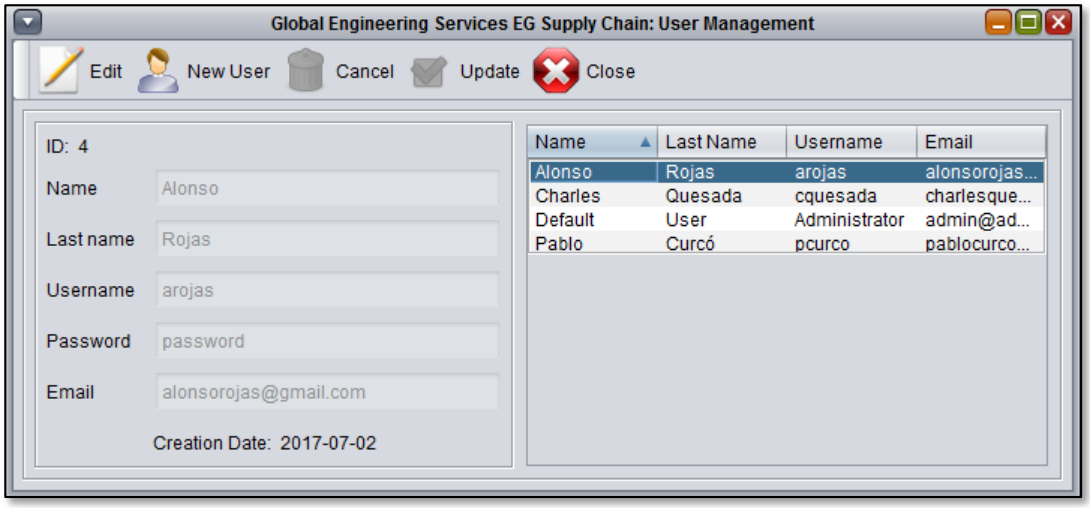


Fuente: Propia

Formulario de administración de usuarios.

Este formulario es utilizado para administrar los usuarios que utilizan el sistema. Desde esta ventana se puede consultar, crear y editar usuarios.

Figura 29: Formulario de administración de usuarios.

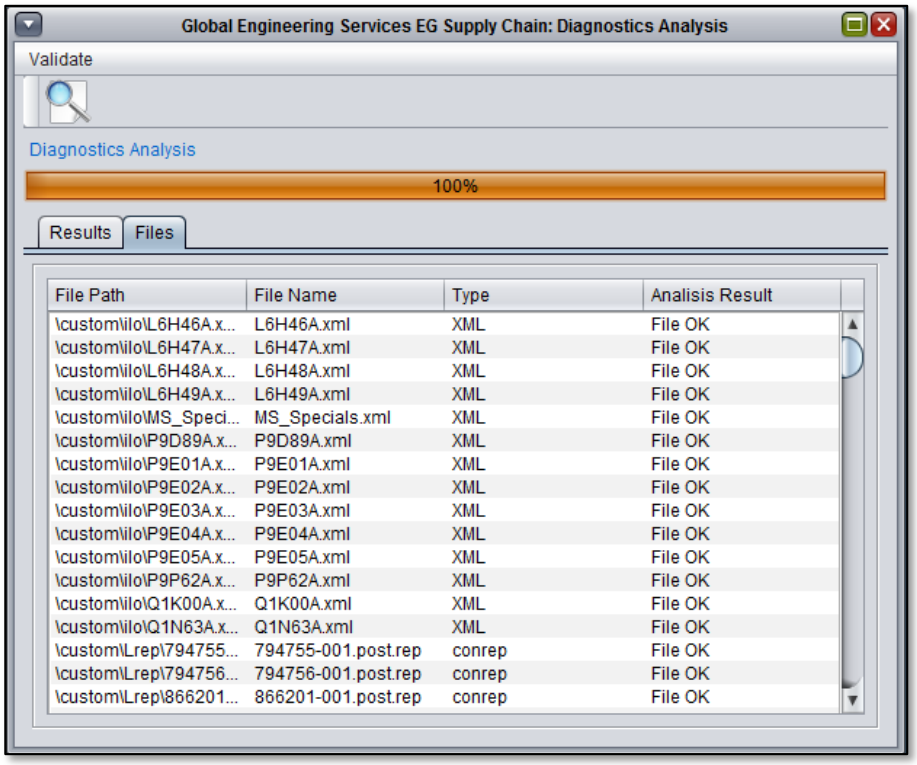


Fuente: Propia

Formulario de validación de componentes.

Este formulario es utilizado para validar los archivos que conforman un componente. Se valida que cada archivo esté libre de errores de sintaxis que pudiesen perjudicar el proceso de análisis o que pudiesen generar errores durante el procesamiento de órdenes en producción.

Figura 30: Formulario de validación de componentes.



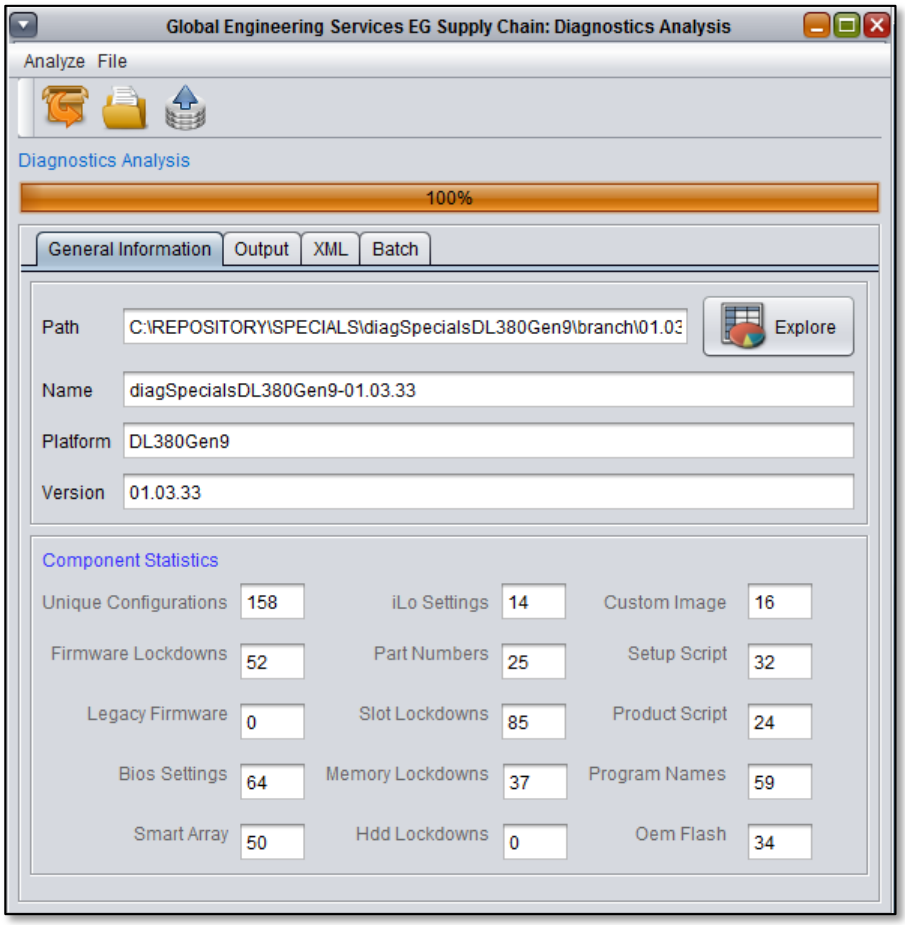
Fuente: Propia

Formulario de análisis de componentes. Información General.

Este formulario puede ser utilizado para analizar un componente de diagnóstico. Este análisis corresponde al proceso por el cual el sistema extrae y procesa la información contenida en el sistema de archivos del componente.

Una vez seleccionado el componente de interés, se ejecutará el análisis correspondiente. La pestaña de información general despliega información del componente e información estadística de su contenido.

Figura 31: Formulario de análisis de componentes. Información General.

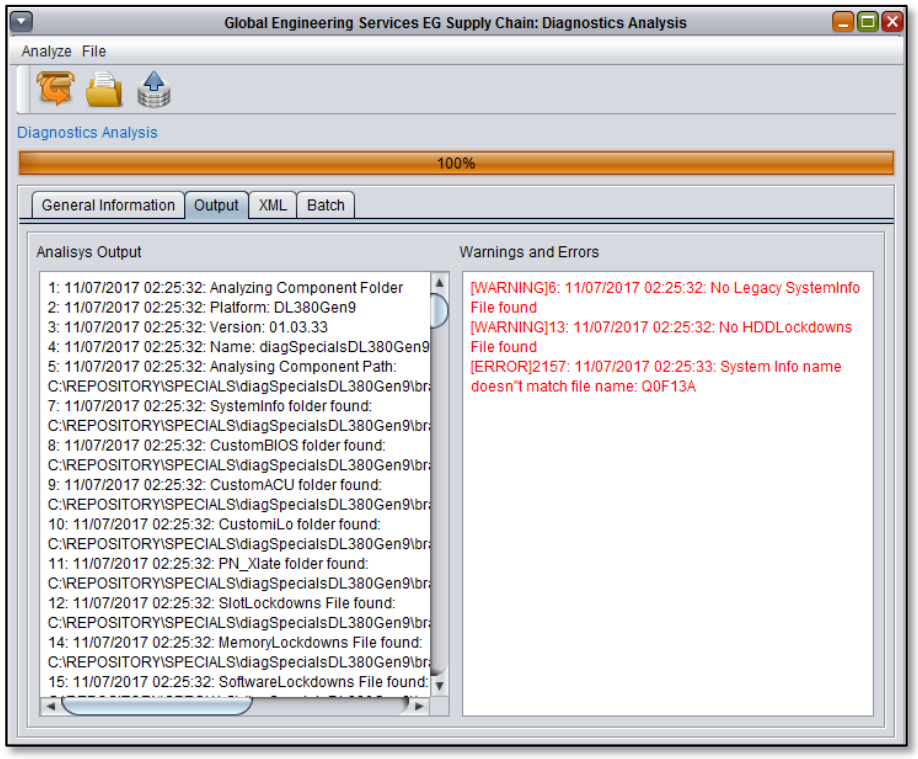


Fuente: Propia

Formulario de análisis de componentes. Salidas.

La pestaña de salidas permite leer la bitácora generada durante el análisis del componente, en la cual se detalla cada paso relevante ejecutado durante el proceso de análisis del componente. Cualquier advertencia o error es desplegado en el campo correspondiente a las salidas de advertencias y errores.

Figura 32: Formulario de análisis de componentes. Salidas.

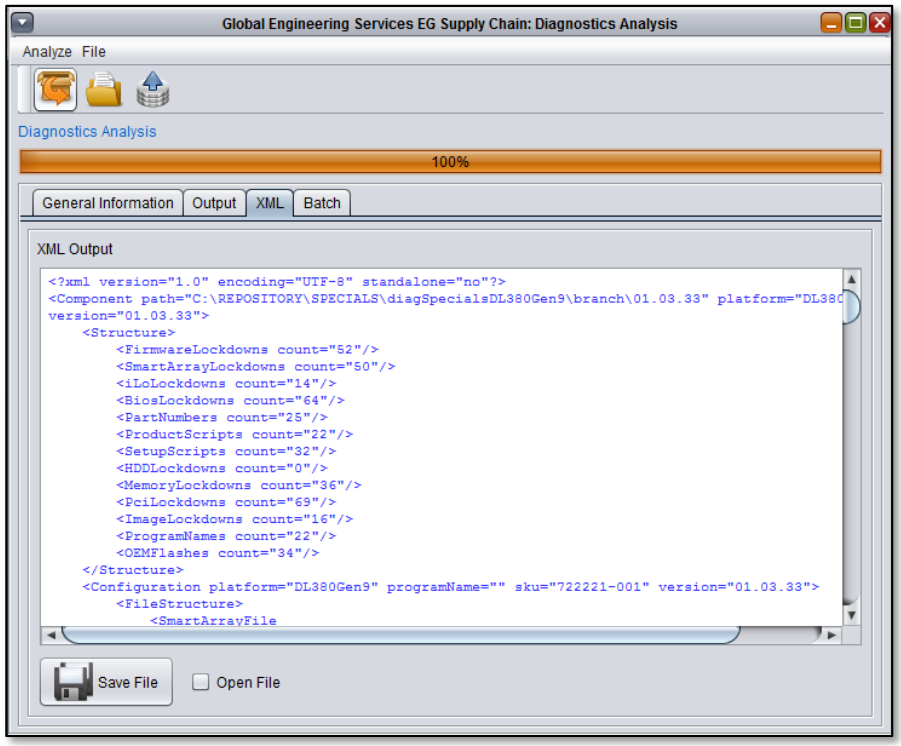


Fuente: Propia

Formulario de análisis de componentes. Salida XML.

La pestaña XML permite obtener el documento XML generado durante el análisis de componente, y el cual contiene toda la información extraída de este. Ofrece la opción de guardar el archivo generado en el sistema de archivos de la computadora.

Figura 33: Formulario de análisis de componentes. Salida XML.

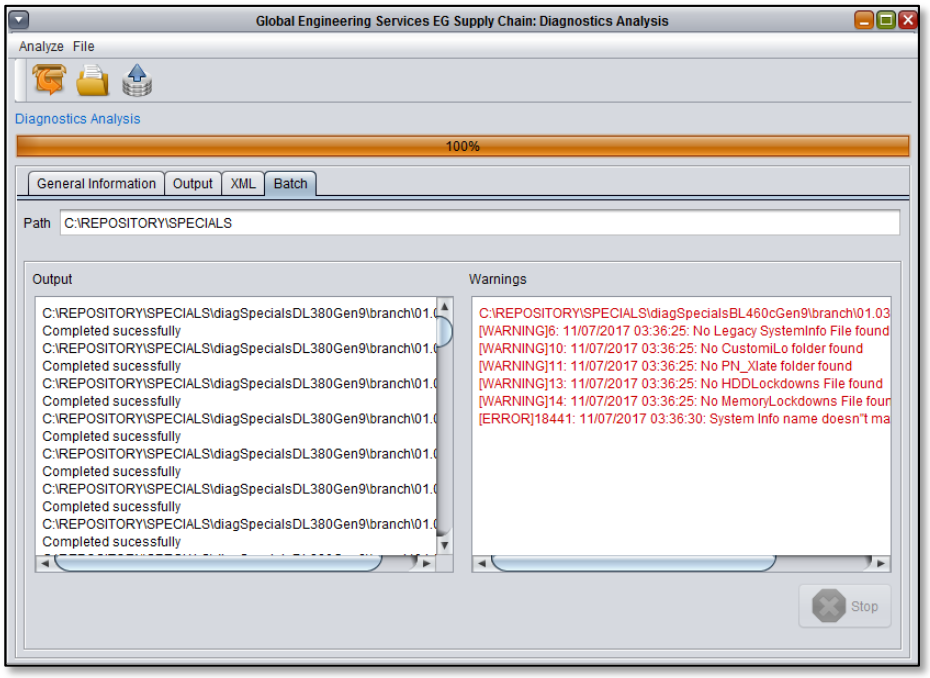


Fuente: Propia

Formulario de análisis de componentes. Procesamiento por lotes.

La pestaña de procesamiento por lotes del formulario de análisis de componentes es habilitada al analizar varios componentes utilizando las opciones correspondientes. Esta ventana despliega los detalles de los análisis y alerta al usuario de cualquier error encontrado.

Figura 34: Formulario de análisis de componentes. Procesamiento por lotes.

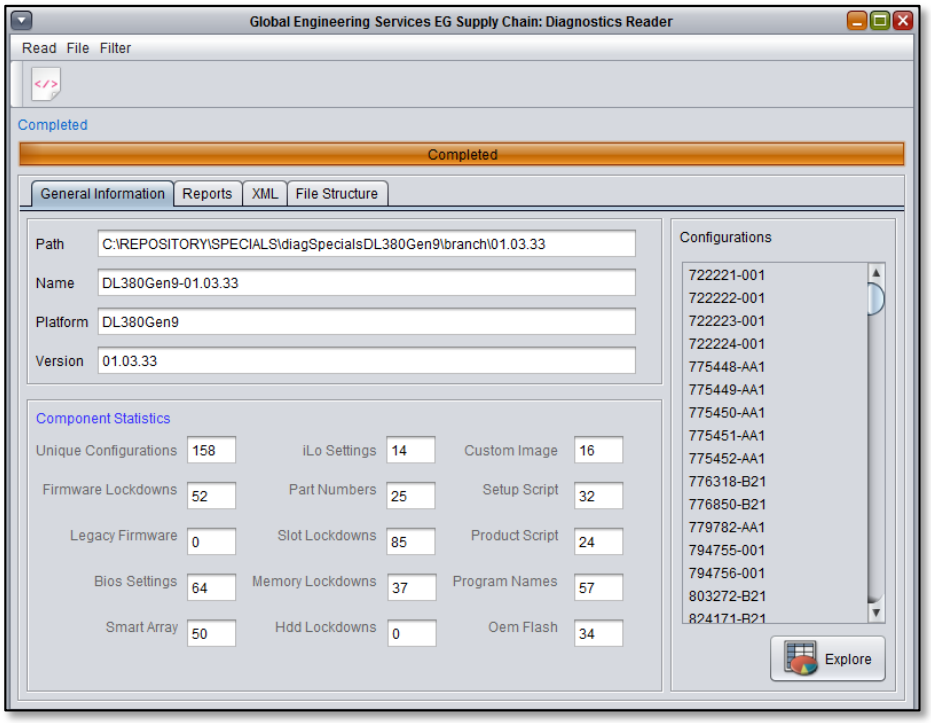


Fuente: Propia

Formulario de exploración de componentes. Información General.

El formulario de exploración de componentes puede ser utilizado para examinar con más detalle la información recolectada por el módulo de análisis. Este requiere un archivo XML producido por el módulo de análisis, generalmente proveniente de la base de datos o del sistema de archivos. La pestaña de información general despliega información del componente e información estadística de su contenido. El usuario puede desplegar el formulario de exploración de configuración desde la lista de configuraciones ubicada a la derecha de esta pestaña.

Figura 35: Formulario de exploración de componentes. Información General.

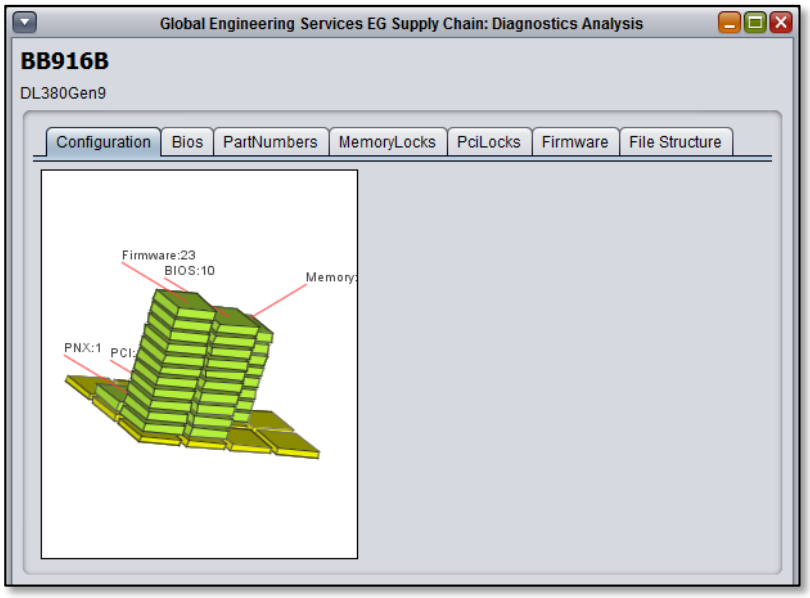


Fuente: Propia

Formulario de exploración de configuración.

El formulario de exploración de configuración permite obtener información de cada una de las configuraciones que conforman cada componente. Esta ventana tiene una pestaña habilitada por cada categoría relevante y una representación gráfica de la configuración.

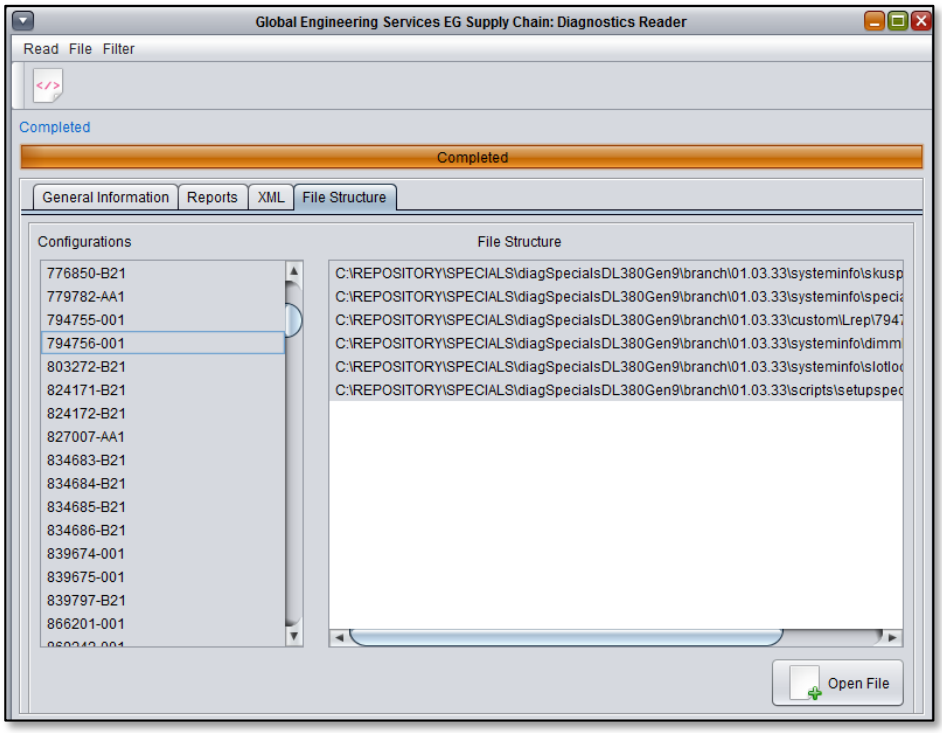
Figura 36: Formulario de exploración de configuración.



Formulario de exploración de componentes. Estructura de archivos.

La pestaña de estructura de archivos incluye una lista de las configuraciones contenidas en el componente elegido. Al seleccionar un elemento de la lista, se despliega otra lista con la ruta de cada uno de los archivos que contienen información de dicha configuración, permitiendo abrir dicho archivo para explorar su contenido.

Figura 37: Formulario de exploración de componentes. Estructura de archivos.

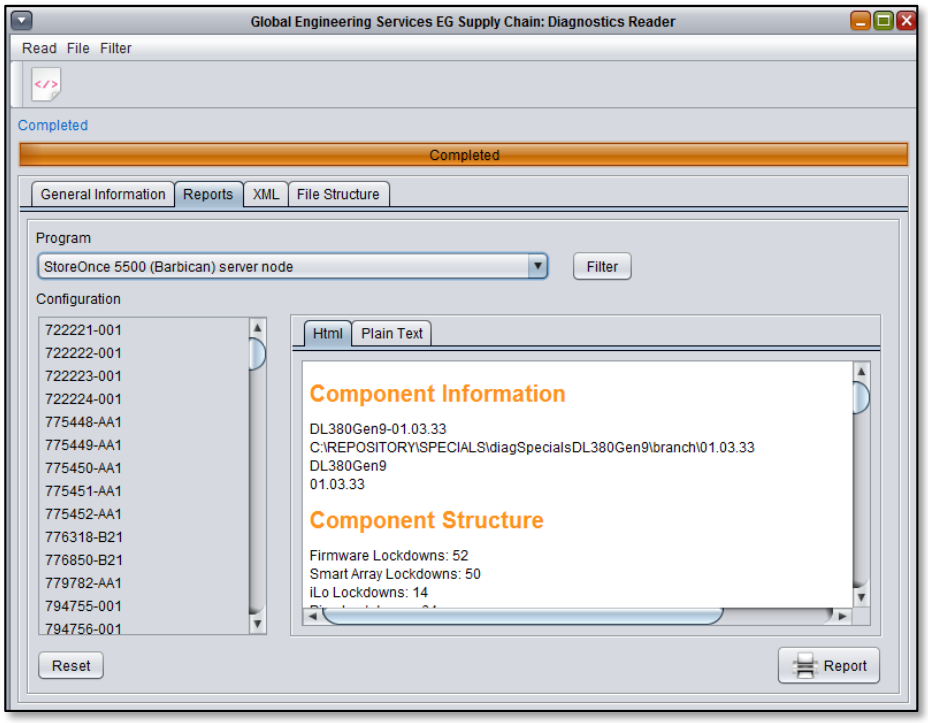


Fuente: Propia

Formulario de exploración de componentes. Reportes.

Permite al usuario diseñar reportes basados en la información obtenida durante el análisis. Una vez seleccionada la información requerida, puede accederse al formulario de impresión.

Figura 38: Formulario de exploración de componentes. Reportes.

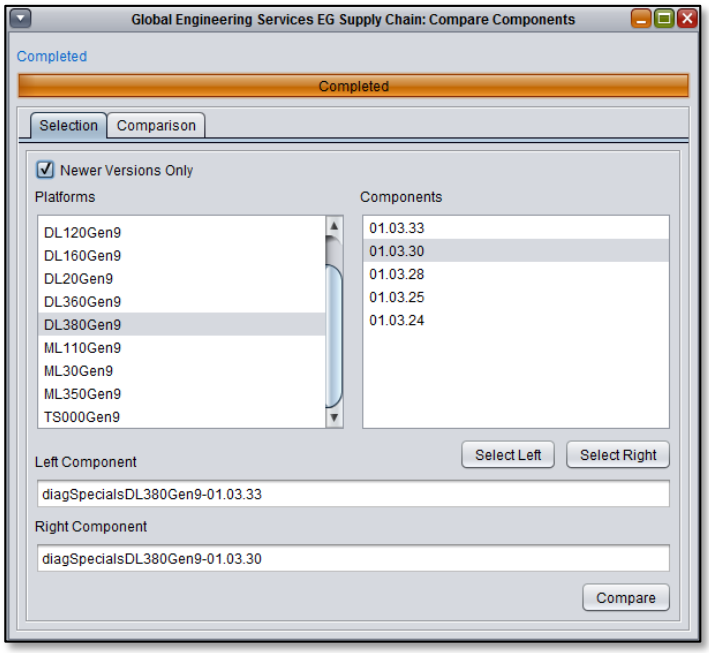


Fuente: Propia

Formulario de comparación de componentes. Pestaña de selección.

El formulario de comparación de componentes es utilizado para comparar dos versiones diferentes del mismo componente, con el fin de explorar los cambios introducidos en la versión más reciente. La pestaña de selección despliega una lista de todas las configuraciones disponibles, y permite seleccionar dos versiones de una misma plataforma para compararlas.

Figura 39: Formulario de comparación de componentes. Selección.



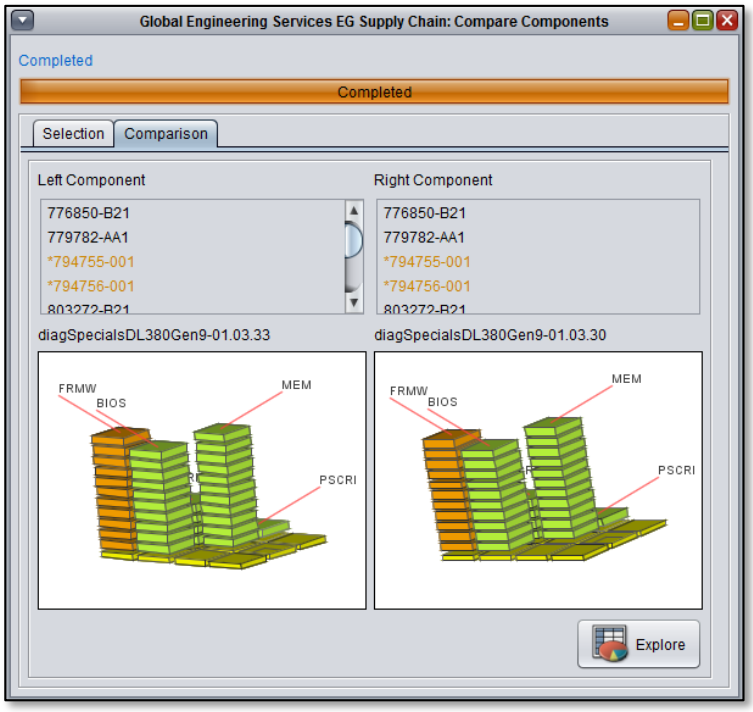
Fuente: Propia

Formulario de comparación de componentes. Pestaña de comparación.

La pestaña de comparación permite observar los resultados de la comparación de las dos versiones seleccionadas en la pestaña de selección. Consta de dos listas, las cuales muestra las configuraciones contenidas en cada componente. Si las configuraciones son idénticas en ambos componentes, el nombre de ambas versiones se mostrará en color negro. Si el sistema determina que las dos configuraciones tienen diferencias, desplegará el nombre en un color naranja.

Cada vez que se selecciona una configuración de la lista, se despliegan los gráficos correspondientes a cada versión de dicha configuración.

Figura 40: Formulario de comparación de componentes. Comparación.

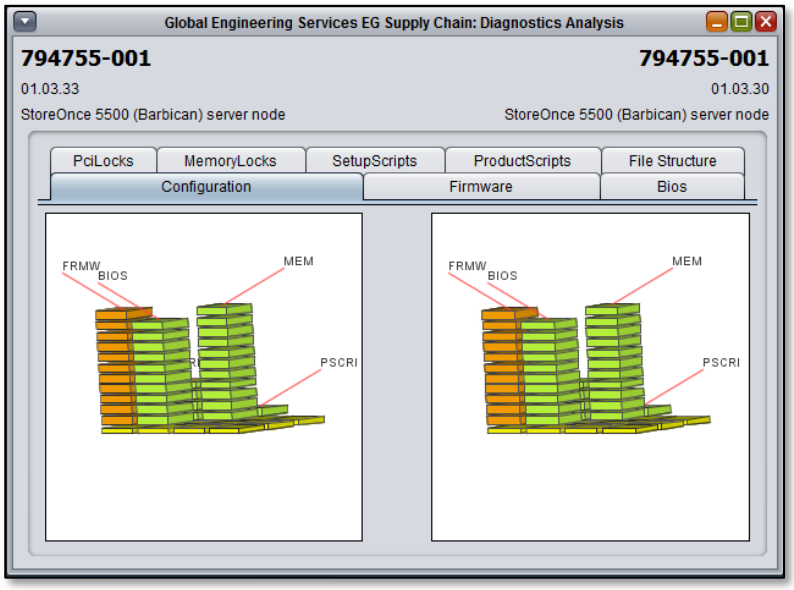


Fuente: Propia

Formulario de exploración de comparación.

Desde la pestaña “comparación” del formulario de comparación de componentes puede accederse al formulario de exploración de comparación. Desde esta ventana puede obtenerse una comparación de la información correspondiente a cada una de las versiones de la configuración seleccionada.

Figura 41: Formulario de exploración de comparación.

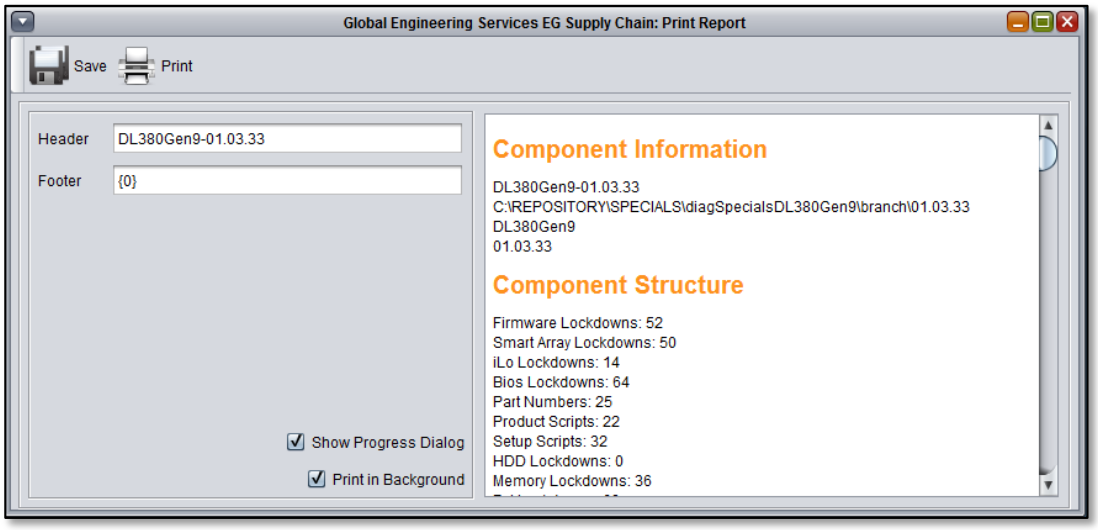


Fuente: Propia

Formulario de impresión.

El formulario de impresión permite observar la prevista de un reporte e imprimirlo de ser necesario. Existen opciones para mostrar el diálogo de progreso de impresión y opciones para ejecutar el proceso de impresión en segundo plano. Es posible, también, guardar el reporte en un archivo, ya sea en formato HTML o texto.

Figura 42: Formulario de impresión.

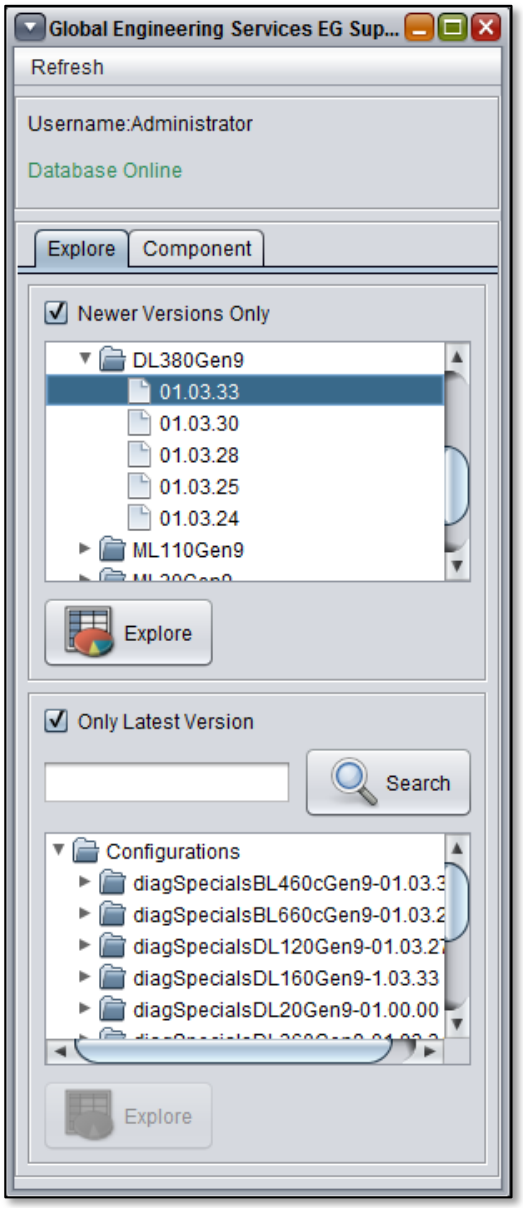


Fuente: Propia

Formulario de consultas.

Figura 43: Formulario de consultas.

Cada vez que un componente es analizado, la información obtenida es almacenada en la base de datos. El formulario de consultas despliega varios controles desde los cuales puede accederse a los componentes almacenados en la base de datos. Desde cualquiera de las listas disponibles se puede seleccionar un componente específico y presionar el botón “Explorar” para desplegar la ventana de exploración de componentes.

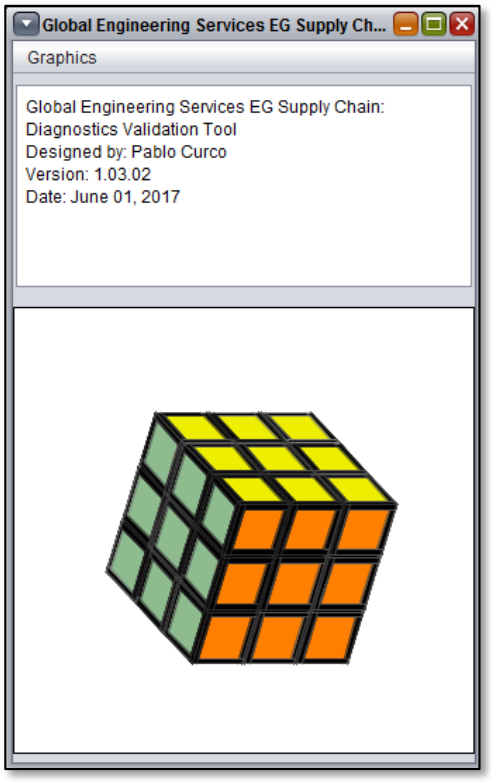


Fuente: Propia

Formulario de ayuda.

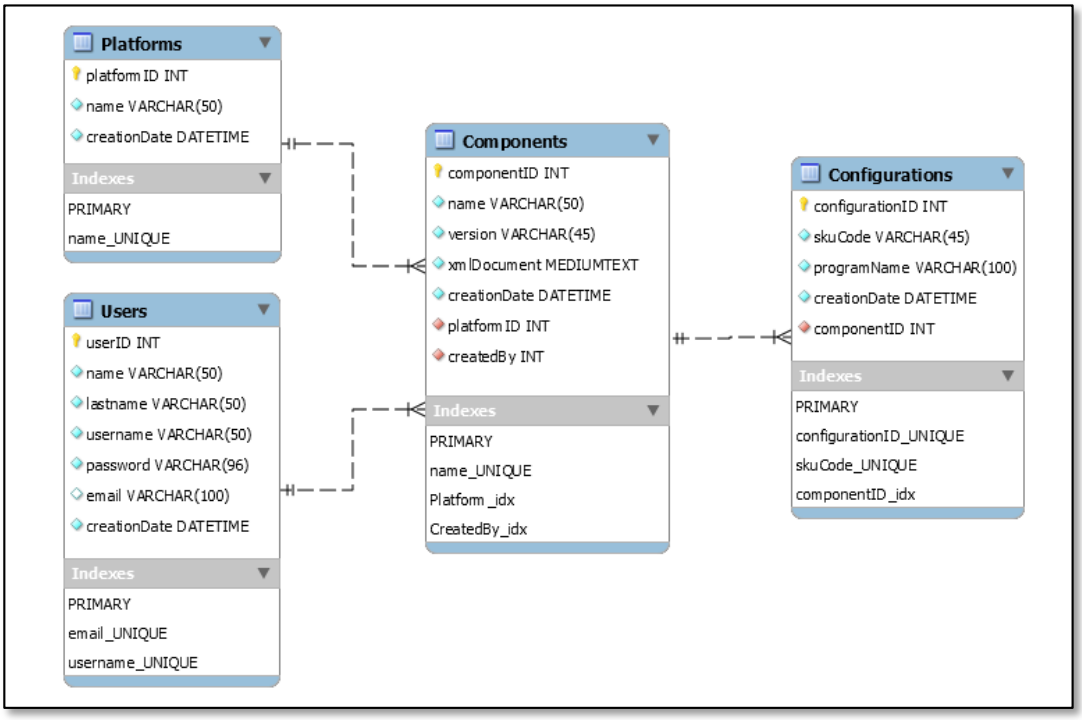
Despliega información relacionada con la operación, el nombre y versión de la aplicación entre otros datos.

Figura 44: Formulario de ayuda.



Fuente: Propia

Diseño de base de datos.



Diccionarios de datos.

Base de datos.

Tabla Platforms.

Cuadro 36: Tabla “Platforms”

Nombre de Columna	Tipo de Datos	Descripción
plataformID	INT	Un entero único auto incrementado utilizado para identificar cada registro de la tabla.
Name	VARCHAR(50)	El nombre de cada

		plataforma.
creationDate	DATETIME	La fecha y hora de la creación del registro.

Fuente: Propia

Tabla Users.

Cuadro 37: Tabla “Users”

Nombre de Columna	Tipo de Datos	Descripción
userID	INT	Un entero único auto incrementado utilizado para identificar cada registro de la tabla.
Name	VARCHAR(50)	El nombre del usuario.
lastName	VARCHAR(50)	El apellido del usuario.
userName	VARCHAR(50)	El identificador de usuario o nombre de sesión único.
password	VARCHAR(64)	El password, el cual será almacenado usando un método de encriptación SHA-2.
Email	VARCHAR(100)	El correo electrónico.
creationDate	DATETIME	La fecha y hora de la creación del registro.

Fuente: Propia

Tabla Components.

Cuadro 38: Tabla “Components”

Nombre de Columna	Tipo de Datos	Descripción
componentID	INT	Un entero único auto incrementado utilizado para identificar cada registro de la tabla.
Name	VARCHAR(50)	El nombre del componente, usualmente con el formato plataforma-versión.
version	VARCHAR(50)	La versión del componente, usualmente con el formato 01.03.26.
xmlDocument	MEDIUMTEXT	Documento XML de formato específico que contiene toda la información relevante de un componente.
platformID	INT	El identificador de la plataforma correspondiente de la tabla platforms.

createdBy	INT	El identificador correspondiente del usuario que inserto el componente.
creationDate	DATETIME	La fecha y hora de la creación del registro.

Fuente: Propia

Tabla Configurations.

Cuadro 39: Tabla “Configurations”

Nombre de Columna	Tipo de Datos	Descripción
configurationID	INT	Un entero único auto incrementado utilizado para identificar cada registro de la tabla.
skuCode	VARCHAR(45)	El nombre de identificación de cada configuración, usualmente compuesta de 6 letras y números.
programName	VARCHAR(100)	El nombre del programa de creación de la configuración.
componentID	INT	El identificador del

		componente correspondiente en la tabla componentes.
creationDate	DATETIME	La fecha y hora de la creación del registro.

Fuente: Propia

Estructura XML para definir un componente.

Nodo componente.

La estructura XML diseñada para almacenar la información de un componente determinado contiene un solo nodo “Componente”. El cual contiene, a su vez, un solo nodo “Estructura” con información de su composición.

Nodo estructura.

El nodo “Componente” tiene un único nodo hijo llamado “Estructura”. Este es utilizado para almacenar estadísticas obtenidas del componente. Cada característica personalizable de una orden es contabilizada.

Figura 45: Nodo “Componente” y nodo “Estructura”.

```
<Component path="" platform="" version="">
    <Structure>
        <FirmwareLockdowns count=""/>
        <SmartArrayLockdowns count=""/>
        <iLoLockdowns count=""/>
        <BiosLockdowns count=""/>
        <PartNumbers count=""/>
        <ProductScripts count=""/>
        <SetupScripts count=""/>
        <HDDLockdowns count=""/>
        <MemoryLockdowns count=""/>
        <PciLockdowns count=""/>
        <ImageLockdowns count=""/>
        <ProgramNames count=""/>
        <OEMFlashes count=""/>
    </Structure>
```

Fuente: Propia

Cuadro 40: Nodo “Estructura”

Nombre de Columna	Tipo de Datos	Descripción
FirmwareLockdowns count	INT	Cantidad de configuraciones con firmware personalizado.
SmartArrayLockdowns count	INT	Cantidad de configuraciones con arreglos de discos personalizados.
iLoLockdowns count	INT	Cantidad de configuraciones con configuraciones de acceso

		remoto personalizado.
BiosLockdowns count	INT	Cantidad de configuraciones con configuraciones de BIOS personalizadas.
PartNumbers count	INT	Cantidad de configuraciones con números de parte personalizados.
ProductScripts count	INT	Cantidad de configuraciones con scripts de configuración personalizada.
SetupScripts count	INT	Cantidad de configuraciones con scripts de configuración personalizada.
HDDLockdowns count	INT	Cantidad de configuraciones con almacenamiento personalizado.
MemoryLockdowns count	INT	Cantidad de

		configuraciones con memoria personalizada.
PciLockdowns count	INT	Cantidad de configuraciones con dispositivos periféricos personalizados.
ImageLockdowns count	INT	Cantidad de configuraciones con sistema operativo personalizado.
ProgramNames count	INT	Cantidad de configuraciones que pertenecen a un programa específico.
OEMFlashes count	INT	Cantidad de configuraciones con binario de BIOS personalizado.

Fuente: Propia

Nodos configuración.

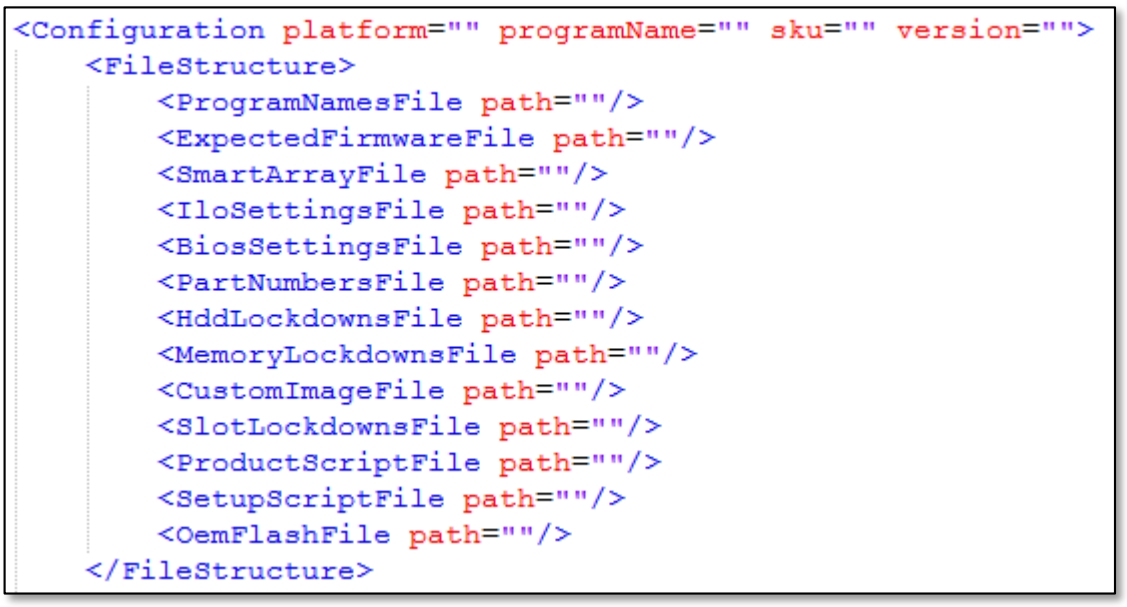
El nodo “Componente” tiene, además del nodo “Estructura”, varios nodos hijo llamados “Configuración” que contienen la información de cada configuración soportada

por el componente. Cada configuración contiene, a su vez, dos nodos únicos, “Estructura de archivos” e “Intención de cliente”.

Nodo estructura de archivos.

Cada nodo “Configuración” tiene un nodo hijo llamado “Estructura de archivos”. Este es utilizado para almacenar las rutas de los archivos que contienen información de dicha configuración.

Figura 46: Nodos “Configuración” y nodo “Estructura de archivos”.



```
<Configuration platform="" programName="" sku="" version="">
    <FileStructure>
        <ProgramNamesFile path=""/>
        <ExpectedFirmwareFile path=""/>
        <SmartArrayFile path=""/>
        <IloSettingsFile path=""/>
        <BiosSettingsFile path=""/>
        <PartNumbersFile path=""/>
        <HddLockdownsFile path=""/>
        <MemoryLockdownsFile path=""/>
        <CustomImageFile path=""/>
        <SlotLockdownsFile path=""/>
        <ProductScriptFile path=""/>
        <SetupScriptFile path=""/>
        <OemFlashFile path=""/>
    </FileStructure>
```

Fuente: Propia

Cuadro 41: Nodo “Estructura de archivos”

Nombre de Columna	Tipo de Datos	Descripción
ProgramNamesFile path	String	Ruta del archivo de nombres de programa.
ExpectedFirmwareFile path	String	Ruta del archivo de

		configuración de firmware.
SmartArrayFile path	String	Ruta del archivo de configuración de arreglo de discos.
IloSettingsFile path	String	Ruta del archivo de configuración del dispositivo de acceso remoto.
BiosSettingsFile path	String	Ruta del archivo de configuración de BIOS.
PartNumbersFile path	String	Ruta del archivo de librería de partes.
HddLockdownsFile path	String	Archivo de configuración de almacenamiento.
MemoryLockdownsFilecount	String	Ruta del archivo de configuración de memoria.
CustomImageFile path	String	Ruta del archivo de configuración de imágenes de software.
SlotLockdownsFile path	String	Ruta del archivo de configuración de periféricos.
ProductScriptFile path	String	Ruta del archivo de scripts personalizadas.
SetupScriptFile path	String	Ruta del archivo de scripts

		personalizadas.
OemFlashFile path	String	Ruta del archivo de configuración de BIOS personalizado.

Fuente: Propia

Nodo intención de cliente.

Cada nodo “Configuración” tiene, además del nodo hijo “Estructura de archivos”, un nodo hijo llamado “Intención de cliente”. Este es utilizado para almacenar todas las características de la orden.

Figura 47: Nodo “Intención de Cliente”.

```
<CustomerIntent>
    <ExpectedFirmwareList>
        <ExpectedFirmware expectedDate="" expectedVersion="" num="" type=""/>
    </ExpectedFirmwareList>
    <SmartArraySettingsList>
        <SmartArraySetting name="" value=""/>
    </SmartArraySettingsList>
    <IloSettingsList>
        <IloSetting name=" " section=" " value=""/>
    </IloSettingsList>
    <BiosSettingsList>
        <BiosSetting name="" value=""/>
    </BiosSettingsList>
    <PartNumbersList>
        <PartNumber description="" partnum name=""/>
    </PartNumbersList>
    <HarddriveLockdownsList>
        <HarddriveLockdown capacity="" driveType="" enclosureBay=""/>
    </HarddriveLockdownsList>
    <MemoryLockdownsList>
        <MemoryLockdown caption="" position="" size=""/>
    </MemoryLockdownsList>
    <CustomImageList>
        <CustomImage partnumber=""/>
    </CustomImageList>
    <SlotLockdownsList>
        <SlotLockdown entry="" key="" name="" slot=""/>
    </SlotLockdownsList>
    <SetupScriptsList>
        <SetupScript description="" name=""/>
    </SetupScriptsList>
    <OemFlashList>
        <OemFlash name=""/>
    </OemFlashList>
</CustomerIntent>
```

Fuente: Propia

Cuadro 42: Nodo “Intención de cliente”

Nombre de Columna	Tipo de Datos	Descripción
ExpectedFirmwareList	Lista de nodos XML	Lista de nodos de configuración de firmware.
SmartArraySettingsList	Lista de nodos XML	Lista de nodos de configuración de arreglo de discos.

IloSettingsList	Lista de nodos XML	Lista de nodos de configuración del dispositivo de acceso remoto.
BiosSettingsList	Lista de nodos XML	Lista de nodos de configuración de BIOS.
PartNumbersList	Lista de nodos XML	Lista de nodos de librería de partes.
HarddriveLockdownsList	Lista de nodos XML	Lista de nodos de configuración de almacenamiento.
MemoryLockdownsList	Lista de nodos XML	Lista de nodos de configuración de memoria.
CustomImageList	Lista de nodos XML	Lista de nodos de configuración de imágenes de software.
SlotLockdownsList	Lista de nodos XML	Lista de nodos de configuración de periféricos.
SetupScriptsList	Lista de nodos XML	Lista de nodos de scripts personalizadas.
OemFlashList	Lista de nodos XML	Lista de nodos de configuración de BIOS personalizado.

Fuente: Propia

Diseño de procesos.

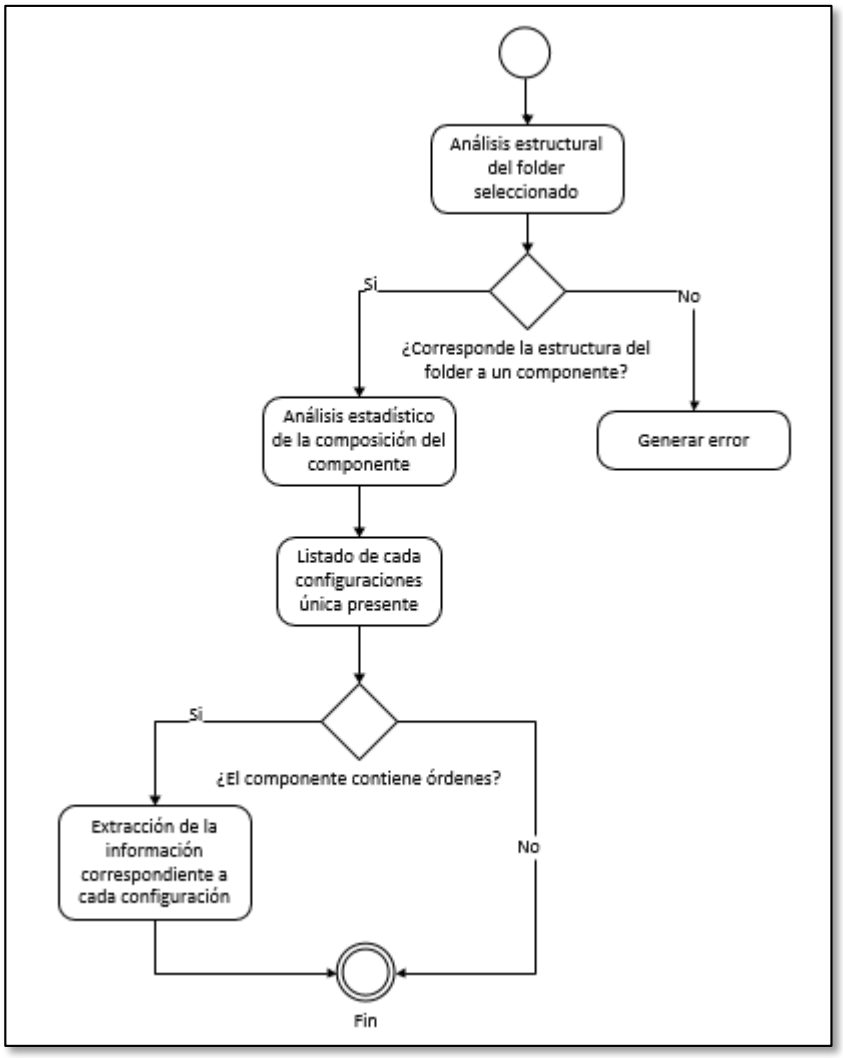
Análisis de un componente.

El módulo de análisis es el encargado de examinar cada componente y extraer la información relevante contenida, la cual será utilizada para generar el documento XML y los reportes correspondientes.

A continuación, se muestra un diagrama que muestra el proceso de análisis de un componente. Una vez establecida la existencia de recursos que contengan información de interés, se procede a hacer un análisis de la composición del componente.

Este análisis consiste en inspeccionar cada uno de los archivos que determinan cada aspecto personalizable de una orden, por ejemplo, los archivos con configuraciones personalizadas de BIOS, generando de esta forma una lista que contenga cada una de las configuraciones que tengan BIOS personalizados. Una vez examinados todos los archivos de interés, se generan una única lista de cada configuración única existente en el componente y los aspectos personalizados que presenta.

Figura 48: Diagrama del proceso de análisis de un componente.



Fuente: Propia

Extracción de la información correspondiente a cada configuración.

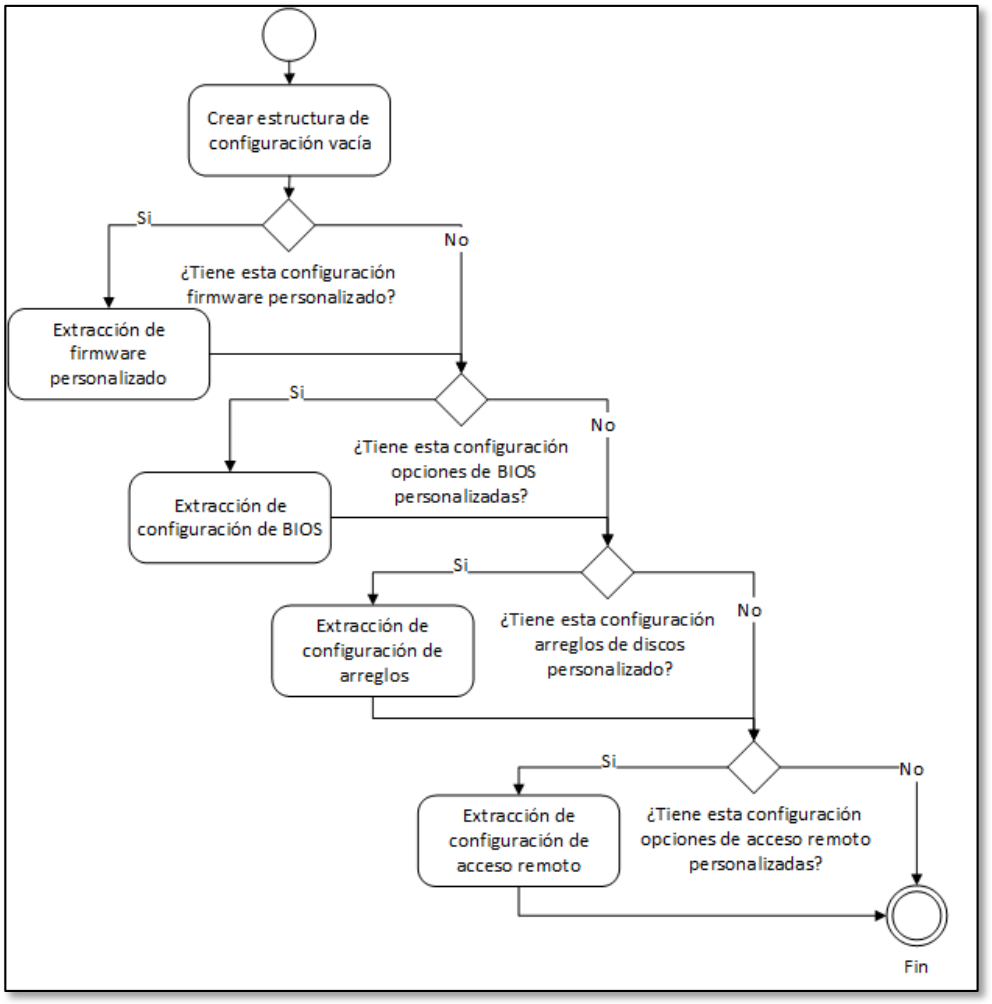
Llegado a este punto, el algoritmo conoce cada una de las configuraciones contenidas en el componente, y conoce además, los aspectos personalizables correspondientes a esta orden.

Se procede, entonces, a crear una estructura de datos por cada configuración existente, y se almacena la información correspondiente. Si la configuración incluye versiones de

firmware personalizados, se extrae, entonces, la información del archivo correspondiente y se coloca en las listas existentes en la estructura de datos para este fin. Si la estructura de datos contiene información de arreglos de discos (RAID) personalizados, se procede, entonces, a almacenar esta información en la lista correspondiente, y se sigue de igual manera con cada una de las doce características de definen una orden.

El diagrama que se presenta a continuación, muestra el proceso de extracción de información correspondiente a las primeras cuatro características del análisis.

Figura 49: Diagrama del proceso de extracción de la información de una configuración.



Fuente: Propia

Diseño de salidas.

Archivo XML componente.

A continuación, se muestra un ejemplo de un XML de componente generado utilizando el módulo de análisis de componente. En este caso específico el componente de prueba utilizado solamente cuenta con una configuración “TEST1A”, la cual solo contiene un firmware personalizado.

Figura 50: Archivo XML componente.

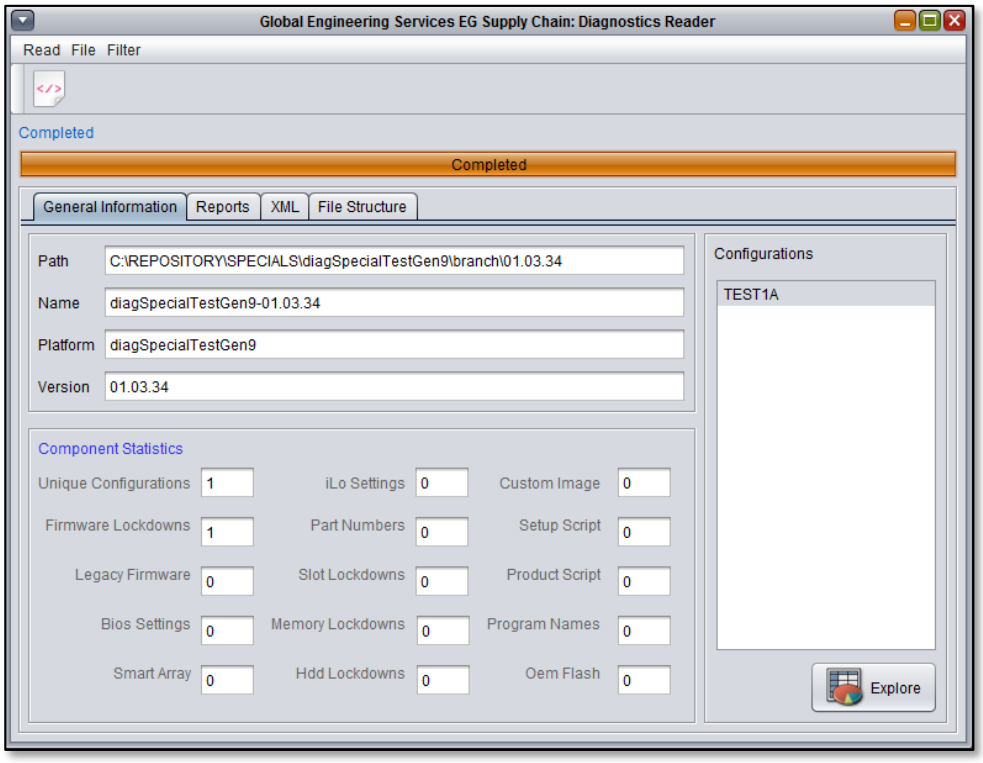
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Component path="diagSpecialTestGen9\branch\01.03.34"
                    platform="diagSpecialTestGen9" version="01.03.34">
    <Structure>
        <FirmwareLockdowns count="1"/>
        <SmartArrayLockdowns count="0"/>
        <iLoLockdowns count="0"/>
        <BiosLockdowns count="0"/>
        <PartNumbers count="0"/>
        <ProductScripts count="0"/>
        <SetupScripts count="0"/>
        <HDDLockdowns count="0"/>
        <MemoryLockdowns count="0"/>
        <PciLockdowns count="0"/>
        <ImageLockdowns count="0"/>
        <ProgramNames count="0"/>
        <OEMFlashes count="0"/>
    </Structure>
    <Configuration platform="diagSpecialTestGen9" programName=""
                                        sku="TEST1A" version="01.03.34">
        <FileStructure>
            <ExpectedFirmwareFile path="\diagSpecialTestGen9\branch
                        \01.03.34\systeminfo\specials\TEST1A.xml"/>
        </FileStructure>
        <CustomerIntent>
            <ExpectedFirmwareList>
                <ExpectedFirmware expectedDate="12/27/2015"
                            expectedVersion="" num="92" type="P89"/>
            </ExpectedFirmwareList>
        </CustomerIntent>
    </Configuration>
</Component>
```

Fuente: Propia

Exploración del componente.

Este es el formulario de exploración de componente, desplegando la información contenida en el XML generado a partir del componente de prueba anteriormente mencionado. Se puede apreciar, tanto la ruta del directorio que contiene el componente, como el nombre, la plataforma y versión correspondiente. En cuanto a estadísticas, se puede apreciar que el componente contiene una única configuración. Esta se muestra en la lista de la derecha, desde la cual podría accederse a sus detalles.

Figura 51: Exploración del componente.

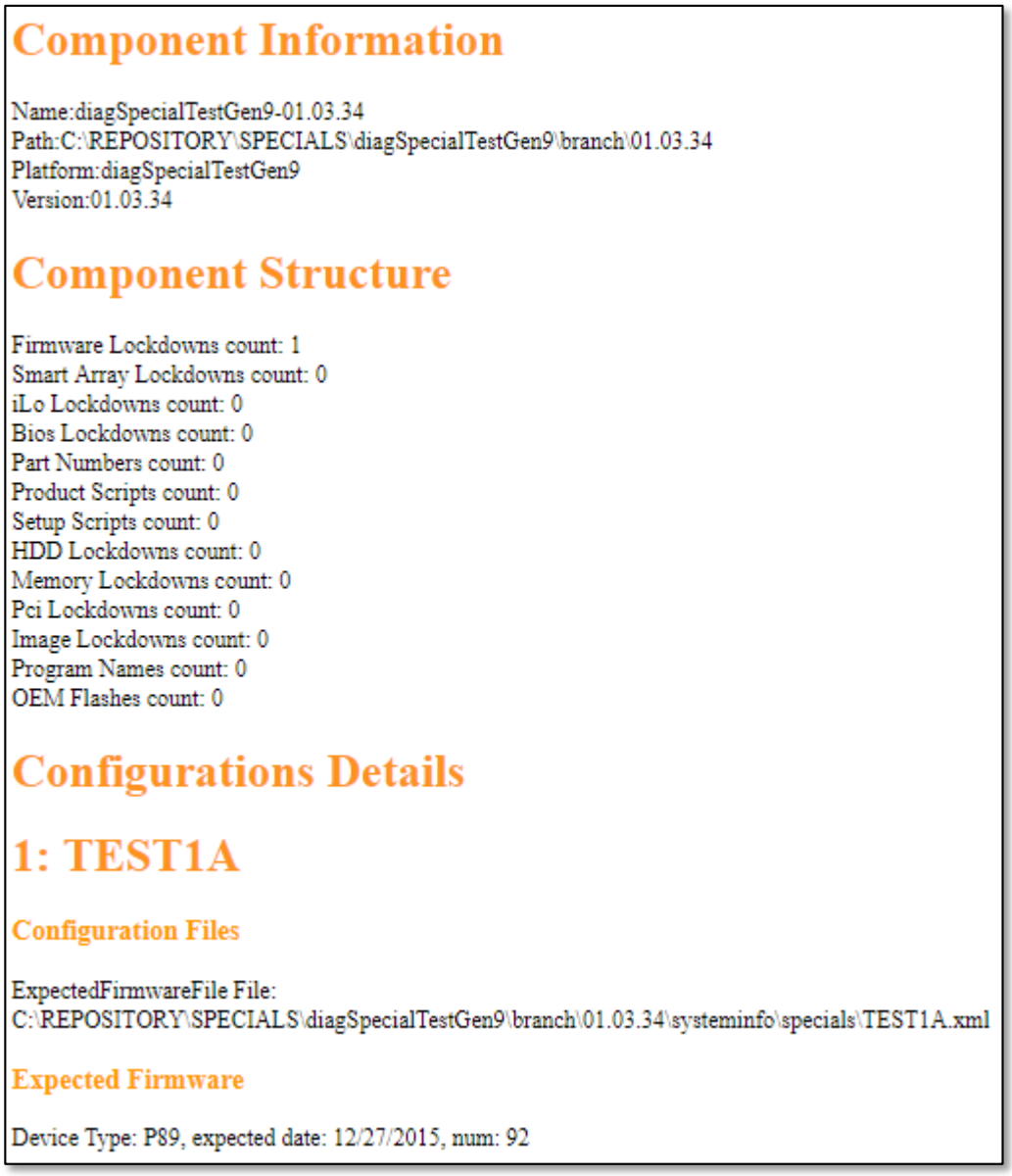


Fuente: Propia

Reporte de componente.

A continuación, se muestra un ejemplo de reporte de componente. Este reporte ha sido generado utilizando el formulario de exploración de componente y el archivo XML correspondiente al archivo XML mostrado anteriormente.

Figura 52: Reporte de componente.



Component Information

Name:diagSpecialTestGen9-01.03.34
Path:C:\REPOSITORY\SPECIALS\diagSpecialTestGen9\branch\01.03.34
Platform:diagSpecialTestGen9
Version:01.03.34

Component Structure

Firmware Lockdowns count: 1
Smart Array Lockdowns count: 0
iLo Lockdowns count: 0
Bios Lockdowns count: 0
Part Numbers count: 0
Product Scripts count: 0
Setup Scripts count: 0
HDD Lockdowns count: 0
Memory Lockdowns count: 0
Pci Lockdowns count: 0
Image Lockdowns count: 0
Program Names count: 0
OEM Flashes count: 0

Configurations Details

1: TEST1A

Configuration Files

ExpectedFirmwareFile File:
C:\REPOSITORY\SPECIALS\diagSpecialTestGen9\branch\01.03.34\systeminfo\specials\TEST1A.xml

Expected Firmware

Device Type: P89, expected date: 12/27/2015, num: 92

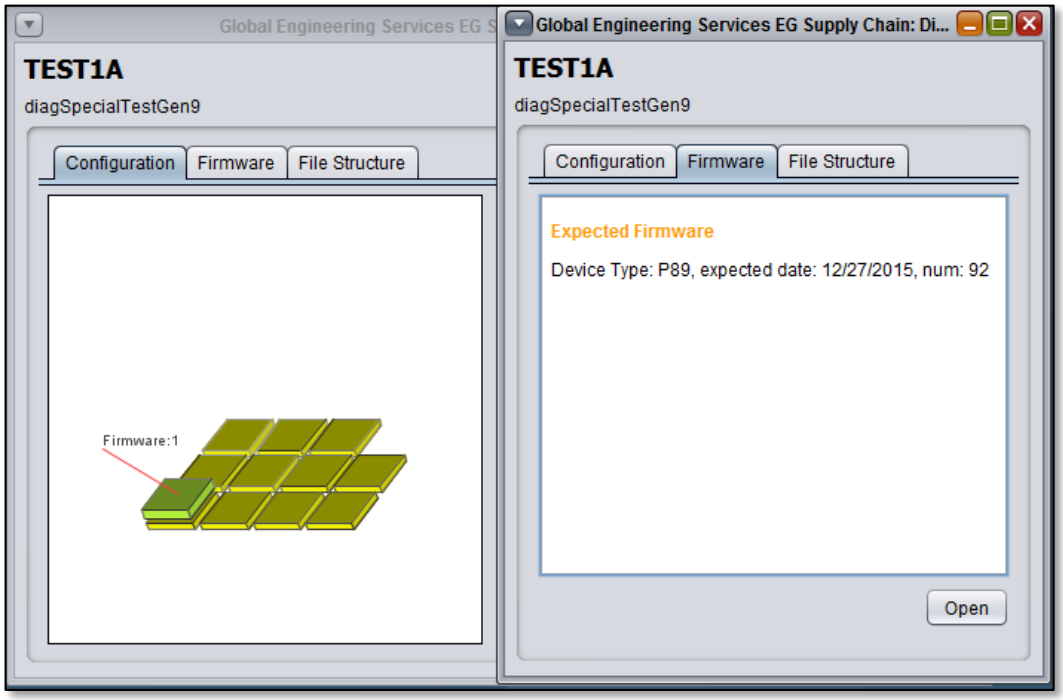
Fuente: Propia

Exploración de configuración.

A continuación, se muestra el detalle de la configuración “TEST1A”. En la pestaña “Configuración se puede observar el gráfico que muestra una columna correspondiente a la lista de firmware personalizado, con un valor de uno. La pestaña “Firmware” contiene la

información relacionada con ese firmware. El dispositivo P89, correspondiente al BIOS, debe tener la versión correspondiente a la fecha 12/27/2015.

Figura 53: Detalle de una configuración.



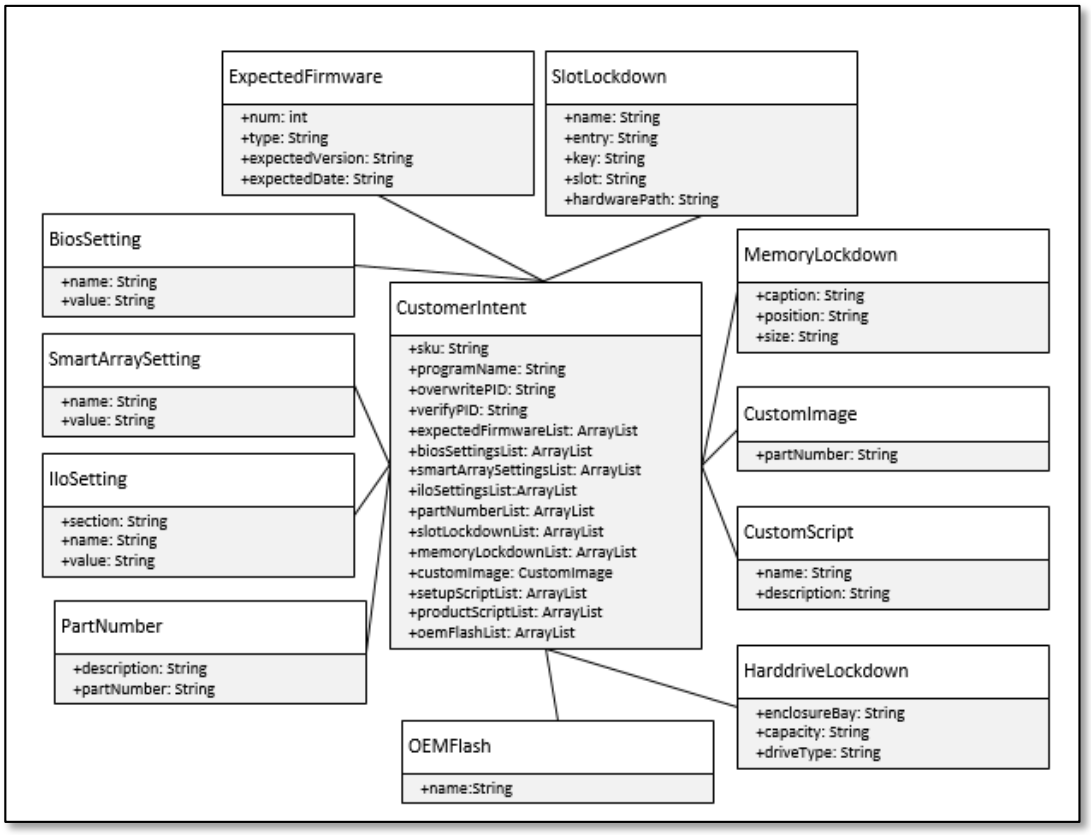
Fuente: Propia

Diagramas UML.

Diagrama UML de la clase "Intención de cliente".

Un componente, conceptualmente hablando, consiste en una colección de configuraciones. Estas configuraciones, como se ha mencionado anteriormente, son descritas en un documento llamado "Documento de intención de cliente", en el cual, se describe con precisión las características que cada orden debe tener.

Figura 54: Diagrama UML de la clase "Intención de cliente".



Fuente: Propia

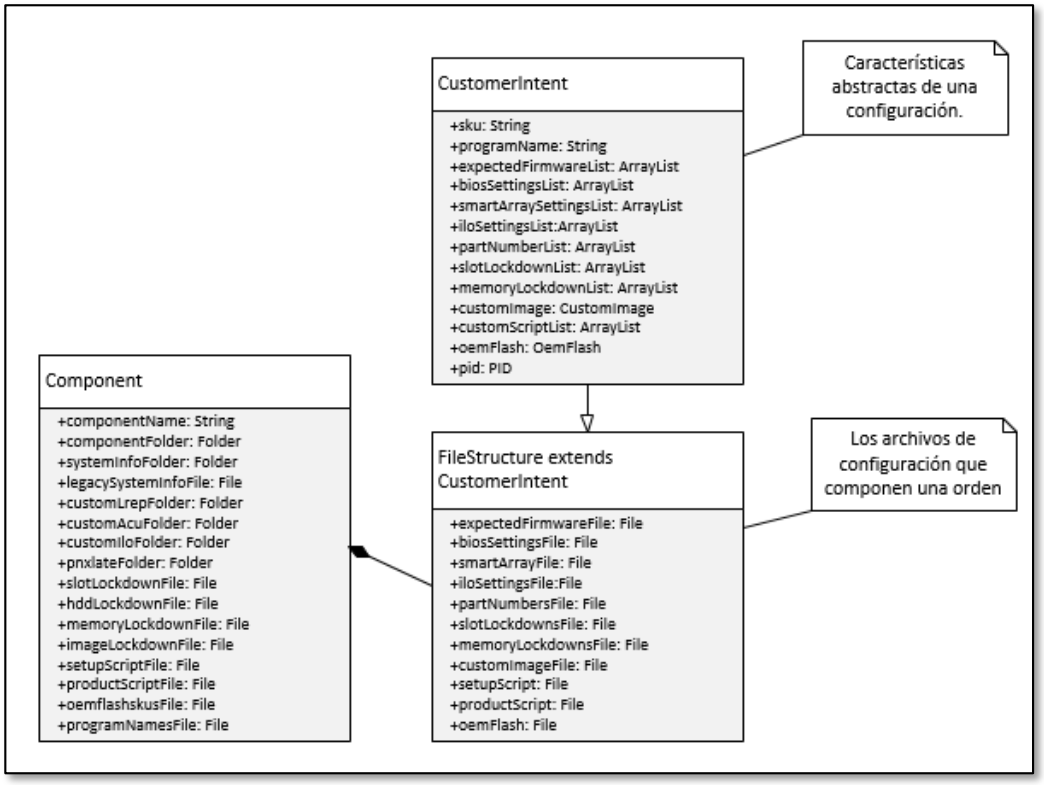
Diagrama UML de las clases "Estructura de archivos" y "Componente".

La clase "intención de cliente" es una representación de las características de una configuración u orden. Por lo tanto, esta clase no contiene ninguna información de los componentes o los archivos que los componen.

Por esta razón, se ha derivado una clase a partir de "intención de cliente", llamada "estructura de archivo", la cual contiene, no solo los mismos campos de la clase original, sino además, información de cuáles archivos del componente contienen la información de dicha configuración. Algunos archivos del componente pueden contener información de una sola configuración mientras otros contienen información de varias configuraciones.

Finalmente, se tiene la clase componente. Esta contiene información de los directorios y archivos principales que la constituyen, así como una lista de clases tipo “estructura de archivos” que contiene la información de cada configuración incluida en el componente.

Figura 55: Diagrama UML de las clases “Estructura de datos” y “Componente”.



Fuente: Propia

Programación

Entradas y salidas.

Análisis de componentes.

Este código es utilizado por el módulo de análisis para obtener todas las configuraciones que poseen arreglos de discos personalizados. El nombre de la

configuración y el nombre de su archivo correspondiente son colocados en una lista, para eventualmente extraer la información relevante del archivo de configuración.

```
while (line != null) {
  SmartArraySetting smartArraySetting = new
SmartArraySetting();
  String values[] = line.split("=");
  smartArraySetting.setName(values[0].trim());
  smartArraySetting.setValue(values[1].trim());
  newFileStructure.getSmartArrayList().add(smartArraySetting);
}
```

Generación de reportes.

Generación de código HTML para la elaboración de reportes. Esta función toma como parámetro un objeto del tipo "estructura de datos", la cual contiene información de una configuración específica, y devuelve el código HTML correspondiente a la configuración de arreglos de disco. Este código es utilizado para generar reportes.

```
output += h3 + "SmartArray Setting" + ch3;
for (SmartArraySetting smartArraySetting :
SmartArraySettingList()) {
  output += smartArraySetting.getName() + ": ";
  output += smartArraySetting.getValue() + br;
}
return output;
```

Validaciones.

Validación de la estructura de archivos de un componente.

Esta porción de código se encarga de verificar que la ruta seleccionada para el análisis corresponda realmente a un componente. Esto se logra verificando que existan los directorios y archivos que normalmente constituyen un componente de diagnóstico. Si no se encuentra ni siquiera un archivo con información de componentes la validación falla y el análisis se cancela.

```
final String systemInfoPath = "\\systeminfo\\specials";
component.setSystemInfoFolder(systemInfoPath);
```

```
SystemInfoFolder = new File(component.getSystemInfoFolder());
isValidComponent = false;
if (SystemInfoFolder.exists()) {
  isValidComponent = true;
}
return isValidComponent;
```

Clases y formularios de cada módulo

Cada módulo está compuesto por lo general de un formulario, parte de la interface gráfica, y una clase específica, la cual provee las funciones y procedimientos necesarios para proveer la funcionalidad requerida por el módulo. Estos formularios y clases han sido nombrados utilizando la convención de nombres del departamento. A continuación, se muestran los formularios y clases que conforman los módulos descritos en el alcance.

Cuadro 43: Clases y formularios desarrollados

Módulo de validación	Clase ComponentValidator Formulario ComponentValidationForm
Módulo de análisis	Clase ComponentAnalyzer Clase ComponentXMLGenerator Formulario ComponentAnalysisForm
Módulo de exploración	Clase ComponentXMLReader Formulario ComponentReaderForm Formulario ConfigurationInspectorForm
Módulo de comparación	Clase ComponentComparison Formulario ComponentComparisonForm Formulario ComparisonInspectorForm
Módulo de consultas	Formulario ComponentLibraryForm

Módulo de seguridad	Clase SecurityManagement Formulario LoginForm
Módulo de reportes	Clase ComponentReportGenerator Formulario ComponentReportsForm
Módulo de mantenimiento	Clase MaintenanceManagement Formulario SystemPreferencesForm Formulario UsersAdministrationForm

Pruebas

A continuación, se presenta el detalle de las pruebas aplicadas al prototipo con el fin de comprobar que se cumpla con los objetivos planteados en el alcance. Se considera que cada prueba tiene un resultado satisfactorio solamente si el resultado obtenido coincide con el resultado esperado.

Datos utilizados durante las pruebas.

Para realizar estas pruebas, se diseñó un conjunto de catorce versiones de componentes. Cada versión incluye archivos con datos de prueba correspondiente a una categoría diferente, con el fin de verificar el funcionamiento de cada uno de los aspectos individualmente.

Pruebas del módulo de seguridad.

Cuadro 44: Pruebas del módulo de seguridad

Prueba realizada	Resultado esperado	Resultado obtenido
Se inició sesión en la aplicación usando el	El módulo de consultas debería aparecer a la	Los resultados obtenidos corresponden a los

formulario de inicio de sesión y proporcionando un usuario y clave de inicio válidos.	izquierda, con el nombre del usuario utilizado para iniciar la sesión. El menú principal y la barra de herramientas deberían habilitarse después de iniciar sesión exitosamente.	resultados esperados.
Se intentó iniciar sesión en la aplicación usando un usuario inexistente. Se intentó además iniciar una sesión con un usuario existente pero con una clave diferente a la clave almacenada en la base de datos.	Se debería desplegar un mensaje de error indicando que el usuario o la clave no son correctos. El menú principal y la barra de herramientas deberían permanecer deshabilitados.	Los resultados obtenidos corresponden a los resultados esperados. Se empleó un usuario inexistente y se utilizó un usuario existente pero con una clave errónea. En ambos casos se recibió un mensaje de error.

Fuente: Propia

Pruebas del módulo de mantenimiento.

Cuadro 45: Pruebas del módulo de mantenimiento

Prueba realizada	Resultado esperado	Resultado obtenido
Se hizo uso del formulario de manejo de	Los cambios deberían aparecer en el archivo	Los resultados obtenidos corresponden a los

variables globales para ajustar las rutas de los directorios utilizados por la aplicación y cambiar el usuario y clave predeterminados.	global.xml, ubicado en el directorio predeterminado de la aplicación.	resultados esperados. Se inspeccionó, además el archivo global.xml y se verificó que su contenido correspondiera a las variables seleccionadas.
Se utilizó el formulario de administración de usuarios para crear un nuevo usuario.	El nuevo registro de usuario debería ser insertado en la tabla de usuarios de la base de datos. Al cerrar y abrir la aplicación el usuario debe aparecer en la lista de usuarios del formulario de administración de usuarios.	Los resultados obtenidos corresponden a los resultados esperados. Se confirmó que un nuevo registro haya sido insertado en la tabla usuario.
Se utilizó el formulario de administración de usuarios para editar un usuario existente. Se verificó que no se permitiera dejar campos en blanco.	El registro correspondiente al usuario debería reflejar los cambios. Al cerrar y abrir la aplicación y acceder otra vez al formulario de administración de usuarios los cambios efectuados deben permanecer presentes.	Los resultados obtenidos corresponden a los resultados esperados. Se confirmó que el registro haya sido actualizado en la tabla usuarios de la base de datos.

Fuente: Propia

Pruebas del módulo de validación.

Cuadro 46: Pruebas del módulo de validación

Prueba realizada	Resultado esperado	Resultado obtenido
Se utilizó el formulario de validación de componentes para validar cada uno de los componentes de prueba.	El sistema debería presentar una lista de todos los archivos validados. Debería, además ser capaz de reportar cualquier error sintáctico encontrado durante la validación.	Los resultados obtenidos corresponden a los resultados esperados. Los componentes de prueba fueron validados y se confirmó que están libres de errores.

Fuente: Propia

Pruebas del módulo de análisis.

Cuadro 47: Pruebas del módulo de análisis

Prueba realizada	Resultado esperado	Resultado obtenido
Como se mencionó anteriormente, se diseñó un conjunto de catorce versiones de componentes. Cada versión incluye una única configuración con	El sistema debería extraer la información contenida en cada componente. La ruta, nombre, plataforma y versión del componente deben ser adecuados, así como las estadísticas de la	Los resultados obtenidos corresponden a los resultados esperados. Se procedió a ejecutar el análisis en cada uno de los catorces componentes de prueba. Se verificaron que

información de una categoría diferente. Se utilizó el módulo de análisis para procesar cada uno de los componentes.	composición del componente. La bitácora del análisis debe describir el proceso de análisis incluyendo las advertencias que apliquen. El archivo XML de componente generado debe tener la estructura e información esperada en cada caso.	los datos obtenidos y el documento XML generado correspondieran con la información contenida en cada versión. Además se confirmó que la información pertinente fuese insertada en la base de datos.

Fuente: Propia

Pruebas del módulo de consultas.

Cuadro 48: Pruebas del módulo de consultas

Prueba realizada	Resultado esperado	Resultado obtenido
Cada vez que un componente es analizado, el formulario de consultas correspondiente a la librería de componentes debe actualizarse, mostrando el nuevo componente en las listas	El usuario debe estar en capacidad de seleccionar un componente del módulo de consultas y seleccionarlo para su exploración. Esto debe desplegar el formulario de exploración de componentes.	Los resultados obtenidos corresponden a los resultados esperados. Se seleccionaron los componentes disponibles, utilizando todos los controles, y el formulario de exploración se desplego

correspondientes y habilitando opciones para explorar el componente obtenido.		con la información correcta.

Fuente: Propia

Pruebas del módulo de exploración.

Cuadro 49: Pruebas del módulo de exploración

Prueba realizada	Resultado esperado	Resultado obtenido
El formulario de exploración ha sido desplegado desde el formulario de consultas, de forma que el contenido de cada componente de prueba se muestre de forma automática. Además, se utilizó el formulario de exploración para abrir cada archivo XML generado desde el sistema de archivos.	La información del componente y de su contenido debería coincidir con los resultados obtenidos en el análisis. Cada configuración contenida debe mostrarse en la lista de configuraciones. El reporte general, y el reporte de cada configuración deberían contener la información correspondiente. Además, la pestaña de estructura de archivos debe enlistar todos	Los resultados obtenidos corresponden a los resultados esperados.

	los archivos que constituyen cada configuración.	

Fuente: Propia

Pruebas del módulo de exploración de configuración.

Cuadro 50: Pruebas del módulo de exploración de configuración

Prueba realizada	Resultado esperado	Resultado obtenido
El formulario de exploración de configuración se ha utilizado para inspeccionar cada una de las configuraciones contenida en los componentes de prueba. Este formulario puede desplegarse desde el formulario de exploración de componente.	El formulario de exploración debería desplegar el gráfico correspondiente a cada configuración, debe habilitar las pestañas adecuadas y presentar la información de forma precisa.	Los resultados obtenidos corresponden a los resultados esperados.

Fuente: Propia

Pruebas del módulo de comparación de componentes.

Cuadro 51: Pruebas del módulo de comparación de componentes

Prueba realizada	Resultado esperado	Resultado obtenido

El formulario de comparación de componentes se ha utilizado para comparar varias combinaciones de componentes de prueba.	El formulario de comparación debe mostrar puntualmente las configuraciones incluidas en cada versión del mismo componente y señalar cuáles configuraciones han cambiado. Además, debe mostrar un gráfico en el que se resalte el aspecto – o los aspectos - de la configuración que han cambiado.	Los resultados obtenidos corresponden a los resultados esperados.

Fuente: Propia

Pruebas del módulo de detalle de comparación.

Cuadro 52: Pruebas del módulo de comparación de componentes

Prueba realizada	Resultado esperado	Resultado obtenido
El formulario de exploración de comparación ha sido utilizado para inspeccionar las dos	El formulario de exploración de comparación debe mostrar puntualmente las configuraciones correspondientes. Mostrar el	Los resultados obtenidos corresponden a los resultados esperados.

versiones de configuración correspondientes. Este formulario debe desplegar la información pertinente a cada versión de forma que sea sencillo identificar sus similitudes y diferencias.	gráfico adecuado y la información que corresponde a cada versión.	

Fuente: Propia

Conclusiones

A continuación, se presentan las conclusiones obtenidas a lo largo de la elaboración de este trabajo.

Desarrollo del prototipo

El objetivo principal de este trabajo, tal y como se mencionó durante la introducción, ha consistido en el desarrollo de un prototipo funcional para la validación de componentes de software. Tanto el prototipo, como los módulos de validación, de análisis, de exploración, de comparación, de consultas, de seguridad, de reportes y de mantenimiento, descritos en el apartado correspondiente al alcance, fueron desarrollados de conformidad con la arquitectura y las especificaciones establecidas.

Requerimientos de sistema.

Utilizando como base la información brindada por el equipo de ingenieros y los resultados obtenidos a partir del cuestionario de evaluación aplicado, se lograron generar los casos de uso y requerimientos de sistema que fueron utilizados durante el diseño y la elaboración del software, y que fueron esenciales para alcanzar el objetivo específico correspondiente. Estos requerimientos fueron obtenidos, analizados y validados con el fin de proveer al sistema con la funcionalidad necesaria para resolver los problemas presentados durante el planteamiento del objeto de estudio.

Diseño de la arquitectura del sistema.

Se logró exitosamente diseñar la arquitectura global del sistema, así como el diseño de cada módulo, interfaz y estructuras de datos requeridas para proporcionar la funcionalidad necesaria para cumplir con los requerimientos de sistema especificados. Durante la etapa de diseño se utilizaron recursos, tales como los casos de uso, tablas de requerimientos y

diagramas UML, con los cuales se elaboraron diagramas de arquitectura de sistema, descripciones completas de las interfaces requeridas, la base de datos del sistema, incluyendo la estructura XML requerida por los módulos de análisis y exploración de componentes, entre otros.

Validación del prototipo.

Se utilizaron datos de prueba y datos de producción para validar todos los módulos desarrollados y determinar si estos cumplen con las especificaciones establecidas, tal y como se propuso en el apartado correspondiente a los objetivos específicos del proyecto. En todos los casos, los resultados obtenidos durante la fase de pruebas corresponden al comportamiento esperado, lo que permitió confirmar que el sistema proporciona la funcionalidad requerida para solventar las necesidades descritas durante el planteamiento del problema de estudio.

Información oportuna y precisa de componentes.

Durante el planteamiento de la problemática de estudio se mencionó la carencia de información oportuna y precisa de las distintas versiones de componentes existentes, la cual es necesaria a lo largo de todo el proceso de elaboración y validación de nuevas versiones de componente. Gracias a las capacidades de análisis, exploración y comparación de componentes, incluidos en el prototipo elaborado, los miembros del equipo tendrán información pertinente y precisa del contenido de estos, resolviendo la problemática mencionada.

Detección de errores sintácticos.

Durante el planteamiento de la problemática de estudio se mencionó, también, la necesidad de diseñar herramientas que permitan detectar errores de tipo sintáctico en

cualquier estructura XML o script PERL contenida en los componentes. Como se explicó en ese apartado, este tipo de errores no es identificado frecuentemente utilizando los procesos y herramientas actuales. Durante la elaboración del prototipo se logró la creación de un módulo de validación, el cual está en capacidad de analizar múltiples archivos a la vez y detectar este tipo de errores, el cual puede ser utilizado por el equipo para resolver este problema.

Conocimiento adquirido

El desarrollo de este trabajo ha permitido, al estudiante, comprobar de cerca la experiencia de elaborar un sistema en su totalidad, desde su mera concepción, pasando por cada una de las actividades fundamentales del proceso de software hasta experimentar la satisfacción de ver la tarea concluida. Además, la experiencia de trabajar con tecnologías como Java y NetBeans, MySQL Server, siguiendo un proceso de desarrollo de software formal, entre otras cosas, ha representado una oportunidad de aprendizaje muy enriquecedora.

Recomendaciones

A continuación, se presentan algunas recomendaciones relacionadas con la implementación del prototipo y su mantenimiento posterior.

Instrucciones de instalación.

Se recomienda, al encargado de la implementación en producción del prototipo, el diseño de un documento con instrucciones precisas de cada uno de los pasos que deben tomarse con el fin de instalar la aplicación en la computadora de cada uno de los ingenieros involucrados. Se recomienda que el documento no sea muy extenso pero que contenga, por lo menos, las siguientes secciones: requerimientos mínimos y recomendados de software y hardware. Una sección con instrucciones claras de los pasos por seguir para instalar cualquier dependencia necesaria para el buen funcionamiento de la herramienta y para instalar los archivos que conforman el sistema.

Capacitación

Al encargado de la implementación del sistema se le recomienda impartir una capacitación breve, de una sola sesión, con una duración máxima de una o dos horas, pero que contemple todas las funciones del sistema. Esto con el fin de que el personal se familiarice con la funcionalidad incluida en la aplicación. Se recomienda utilizar primero los componentes de prueba, desarrollados durante la etapa de validación del prototipo, ya que estos tienen una composición más simple y es más fácil observar el proceso de análisis y extracción de información. Posteriormente, puede mostrarse el análisis de componentes reales, los cuales contienen cientos de órdenes diferentes, con el fin de que el personal pueda observar las salidas reales del sistema.

Procedimientos

Se recomienda a los directores de proyecto y gerentes involucrados, la implementación de procedimientos que ayuden a obtener el mayor provecho posible de la herramienta.

Estos pueden incluir un nuevo procedimiento en el cual los desarrolladores o validadores deban incluir, cada vez que se haga un lanzamiento de una nueva versión de componente, un archivo de texto, que contenga la información generada por el módulo de reportes de la herramienta, que pueda ser utilizado por los clientes finales como una referencia del contenido del componente.

Estos procedimientos podrían establecerse en una o dos reuniones de corta duración (una hora máximo) entre algunos representantes de cada área interesada - por ejemplo las áreas de desarrollo, validación y manejo de proyectos -, y podría seguir el formato de una lluvia de ideas, de forma que las ideas más relevantes prevalezcan. Los temas a considerar serían: que cambios deberían ocurrir en procedimientos existentes, o que nuevos procedimientos son requeridos para obtener el mayor provecho del nuevo sistema.

Mantenimiento del software

Finalmente, se recomienda crear un ciclo de mantenimiento de software, durante el cual puedan implementarse mejoras o cambios en la herramienta, según sea conveniente. Esto con la finalidad de contar siempre con una herramienta que mejor se ajuste a las necesidades del equipo.

A lo largo del ciclo de mantenimiento, que podría ser mensual o bimestral, el responsable del sistema se encargaría de recolectar y priorizar peticiones de cambio o peticiones de mejora. Se seguiría un modelo de desarrollo similar al utilizado durante el

desarrollo de la aplicación, con sus respectivas fases de especificación de requerimientos, desarrollo y pruebas.

Además, se recomienda que el encargado del sistema tenga conocimiento, tanto en NetBeans como en el lenguaje Java, tecnologías de bases de datos MySQL y el lenguaje XML. En caso de el encargado no tenga conocimiento de dichos temas, se recomienda revisar literatura disponible relacionada principalmente al lenguaje Java y el manejo de entrada y salida de datos.

REFERENCIAS

Allen, M., 2000. *Estructuras de datos en JAVA.* España: Addison.

Balena, F., 2006. *Programando Microsoft Visual Basic 2005: El lenguaje.* Primera Edición ed. Estados Unidos: Microsoft Press.

Beneke, T. & Wieldt, T., 2013. *JavaOne 2013 Review: Java Takes on the Internet of Things.* [En línea]

Disponible en: http://www.oracle.com/technetwork/articles/java/afterglow2013-2030343.html

[Último acceso: 1 Junio 2017].

Bray, T. & Paoli, J., 2008. *Extensible Markup Language (XML) 1.0.* [En línea]

Disponible en: https://www.w3.org/TR/REC-xml/

[Último acceso: Quinta Edición Junio 2017].

Brooks, C., Freeman, W., Ho, J. & Smith, D., 2006. *CompTIA server+*. Estados Unidos: Marcraft International.

Bruegge, B. & Allen, A., 2002. *Ingeniería de software orientada a objetos.* México: Prentice Hall.

Collins-Sussman, B., W. Fitzpatrick, B. & Michael Pilato, C., 2011. *Version Control with Subversion: For Subversion 1.7.* [En línea]

Disponible en: http://svnbook.red-bean.com/en/1.7/svn-book.pdf

[Último acceso: 11 07 2017].

Connolly, T. & Begg, C., 205. *Database Systems.* Cuarta Edición ed. United States of America: Addison-Wesley.

Deitel, H. & Deitel, P., 2008. *Cómo programar en Java.* Séptima edicion ed. México: Pearson Education.

Deitel, P. & Deitel, H., 2008. *Internet y World Wide Web. Como programar.* Cuarta Edición ed. New Jersey: Pearson Education.

Eck, D., 2014. *Introduction to programming using Java [Introducción a la programación usando Java].* [En línea]
Disponible en: http://math.hws.edu/javanotes

Hernández, R., Fernández, C. & Baptista, M., 2003. *El proceso de la investigación.* Tercera Edición ed. México: Mc Graw Hill.

Hernández, R., Fernández, C. & Baptista, M., 2010. *Metodología de la investigación.* Quinta Edición ed. México: Mc Graw Hill.

Hernández, R., Fernández, C. & Baptista, P., 2006. *Metodología de la investigación.* Cuarta Edición ed. México: The McGraw-Hill.

Hewlett Packard, 2015. *HP ROM-Based Setup Utility User Guide.* [En línea]
Disponible en: http://h20566.www2.hpe.com/hpsc/doc/public/display?docId=c00191707
[Último acceso: 8 Junio 2017].

Jones, C. & Drake, F., 2002. *Python y XML.* Primera Edición ed. Estados Unidos de America: O'Reilly.

Joyanes, L., 2006. *Programación en C++. Algoritmos, estructuras de datos y objetos.* Segunda Edición ed. España: McGraw-Hill.

Kendall, K. & Kendall, J., 2005. *Análisis y diseño de sistemas.* Sexta Edición ed. México: Pearson Education.

Microsoft, 2016. *Overview of windows programing in C++*. [En línea]
Disponible en: https://docs.microsoft.com/es-es/cpp/mfc/sdi-and-mdi
[Último acceso: 21 06 2017].

Oracle Corporation, 2012. *Las 10 razones principales para usar mysql como base de datos integrada.* [En línea]
Disponible en: https://www.mysql.com/why-mysql/white-papers/las-10-razones-principales-para-usar-mysql-como-base-de-datos-integrada/
[Último acceso: 06 06 2017].

Oracle Corporation, 2016. *Java™ Platform Standard Ed. 7: Class DocumentBuilder.* [En línea]
Disponible en: https://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/DocumentBuilder.html
[Último acceso: 13 07 2017].

Oracle Corporation, 2016. *NetBeans IDE 8.2 release notes.* [En línea]
Disponible en: https://netbeans.org/community/releases/82/relnotes.html
[Último acceso: 27 Mayo 2017].

Oracle Corporation, 2017. *What is Java.* [En línea]
Disponible en: https://www.java.com/es/download/faq/whatis_java.xml
[Último acceso: 14 07 2017].

Pérez, M., 2016. *WINDOWS 10 Práctico.* Primera Edición ed. México: Alfaomega.

Ramírez, J., 2012. *Procedimiento para la elaboración de un análisis FODA como una herramienta de planeación estratégica en las empresas.* [En línea]
Disponible en: https://www.uv.mx/iiesca/files/2012/12/herramienta2009-2.pdf

Schwartz, R., Foy, B. & Phoenix, T., 2011. *Learning Perl.* Sexta Edición ed. Estados Unidos: O'Reilly.

Sommerville, I., 2011. *Ingeniería de software 9.* México: Pearson.

ANEXOS

Anexo 1

Cuestionario utilizado como instrumento de recolección de datos.

1. ¿Considera que los procedimientos de validación actuales son suficientes para detectar todos los errores introducidos durante la producción de nuevas versiones de componentes?

○ Muy de acuerdo

○ De acuerdo

○ En desacuerdo

○ Muy en desacuerdo

2. ¿Existe información de las características de las órdenes incluidas en cada versión de componente de software?

○ Si

○ No

3. ¿Cuenta con un sistema informático que le asista durante el proceso de validación de componentes?

○ Si

○ No

4. ¿Considera importante contar con un módulo que le ayude a encontrar errores de sintaxis en los archivos que constituyen un componente?

○ Muy de acuerdo

○ De acuerdo

○ En desacuerdo

○ Muy en desacuerdo

5. ¿Considera importante contar con un módulo que le permita analizar automáticamente el contenido de un componente?

○ Muy de acuerdo

○ De acuerdo

○ En desacuerdo

○ Muy en desacuerdo

6. ¿Considera importante contar con un módulo que le permita explorar el contenido de los componentes?

○ Muy de acuerdo

○ De acuerdo

○ En desacuerdo

○ Muy en desacuerdo

6. ¿Considera importante contar con un sistema que le permita generar reportes del contenido de los componentes?

○ Muy de acuerdo

○ De acuerdo

○ En desacuerdo

○ Muy en desacuerdo

7. ¿Considera importante contar con un módulo que le ayude a explorar distintas versiones de un mismo componente y obtener detalles de sus diferencias?

○ Muy de acuerdo

○ De acuerdo

○ En desacuerdo

○ Muy en desacuerdo

8. ¿Considera importante contar con un módulo que le permita generar reportes del contenido de los componentes?

○ Muy de acuerdo

○ De acuerdo

○ En desacuerdo

○ Muy en desacuerdo