UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

GENERACIÓN DE REPORTES CON TABLAS DE DATOS PARA INTELIGENCIA DE NEGOCIOS

REPORTE DE TRABAJO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

MARICELA TELLEZ FLORES

DIRECTOR DE TESIS:
DR. JORGE LUIS ORTEGA ARJONA
2018

Hoja de datos del jurado

1. Datos del alumno	1. Datos del alumno
Apellido paterno	Tellez
Apellido materno	Flores
Nombre (s)	Maricela
Teléfono	5533711874
Universidad Nacional Autónoma de México	Universidad Nacional Autónoma de México
Facultad de ciencias	Facultad de ciencias
Carrera:	Ciencias de la Computación
Número de cuenta:	401007341
2. Datos del tutor	2. Datos del tutor
Grado	Dr
Nombre	Jorge Luis
Apellido paterno	Ortega
Apellido materno	Arjona
3. Datos del sinodal 1	3. Datos del sinodal 1
Grado	M en C
Nombre	María Guadalupe Elena
Apellido paterno	Ibargüengoitia
Apellido materno	González
4. Datos del sinodal 2	4. Datos del sinodal 2
Grado	M en C
Nombre	Gustavo Arturo
Apellido paterno	Márquez
Apellido materno	Flores
5. Datos del sinodal 3	5. Datos del sinodal 3
Grado	M en I
Nombre	Gerardo
Apellido paterno	Avilés
Apellido materno	Rosas
6. Datos del sinodal 4	6. Datos del sinodal 4
Grado	Dr.
Nombre	Francisco
Apellido paterno	Valdés
Apellido materno	Souto
7. Datos de trabajo escrito	7. Datos de trabajo escrito
Título	Generación de reportes con tablas de datos para inteligencia de negocios
Subtítulo	
Número de páginas	98p
Año	2018

Resumen

Las empresas que cuentan con grandes volúmenes de datos en diversas fuentes de almacenamiento coinciden en la necesidad de disponer de ellos para su tratamiento, análisis o visualización. Lo anterior con el objetivo de generar información con esos datos que conlleve a la obtención de conocimiento y mejorar su competitividad.

Los reportes o informes empresariales computarizados permiten el acceso agíl a la información, y su importancia radica en que proporcionan un apoyo importante en la toma de decisiones. Estos reportes forman parte de lo que es conocido como Inteligencia de Negocios o BI, por sus siglas en inglés.

En este trabajo se describe una aproximación para generar reportes empresariales con una tabla de datos para resolver las necesidades específicas de una empresa. La generación o desarrollo se basa en el controlador —o plugin— *jqGrid*, que cuenta con una interfaz web, búsquedas personalizadas, filtros, paginación, entre otras funciones. También se muestra la aplicación de los reportes generados, que han conseguido cubrir con las demandas expuestas por la empresa y su aplicación se ha extendido a diversas áreas de negocio.

Índice general

Índice de figuras

Capítulo 1
Introducción

1.1. El Contexto

Una demanda constante en las empresas que cuentan con parte de su información almacenada en diversos medios, como una base de datos, archivos de texto, hojas de excel u otras fuentes de almacenamiento, es que ésta se mantenga al alcance del usuario en cualquier momento. Existen diversos medios, metodologías, sistemas, herramientas y recursos de información para acceder a los datos y permitir su consulta o manipulación.

Todo lo anterior forma parte de la base fundamental para establecer lo que es conocido como Inteligencia de Negocios. Por otro lado, los *reportes informáticos* exhiben, de manera organizada, la información contenida en una base de datos, a través de un diseño personalizado y de fácil acceso al usuario, con el objetivo de resolver las necesidades diarias de acceso a la información. Los reportes empresariales son un ejemplo de reporte informático y representan una herramienta de la Inteligencia de Negocios.

1.2. El Problema

La generación de los reportes empresariales (para inteligencia de negocios) que se encontraba implementada en una Empresa[1] no cubría las necesidades de tiempo de respuesta, efectividad en los datos mostrados y en el despliegue visual de la información. Además, no contaba con opciones de búsqueda y facilidad de acceso para el usuario.

1.3. El Objetivo

El objetivo principal es la generación de reportes empresariales de fácil acceso, con información efectiva y con un diseño que proporcione un uso más sencillo al usuario. También la inclusión de filtros de búsqueda personalizados, entre otras características.

[1]Se omite el nombre por aspectos de privacidad.

1.4. La Aproximación

La generación de reportes que se describe en este trabajo se realiza por medio de una tabla de datos (también conocida como *grid* de datos, *grid view o datagrid*). Una tabla de datos permite mostrar información contenida en diversas fuentes de información (por ejemplo, en servidores de almacenamiento, en una base de datos, en dispositivos físicos, en archivos de tipo Excel o archivos de texto plano) y posee funciones que actúan sobre los renglones o registros de la tabla. El controlador utilizado aquí es jqGrid (basado en JavaScript) que puede integrarse con cualquier tecnología de servidor y que ofrece soluciones para representar y manipular datos. El resultado final es una aplicación web que incluye los reportes generados.

1.5. Contribuciones

Los reportes cubren las necesidades solicitadas por la Empresa. La implementación permite agregar nuevos reportes que pueden personalizarse de manera individual para adecuarse a nuevos requerimientos.

Mejoras aportadas:

- Visualización por medio de la paginación de un mayor volumen de datos, de acuerdo a los estándares o métricas establecidos por la Empresa,
- Opción de búsquedas por medio de filtros para agilizar las consultas.
- Un nuevo diseño visual que mejore la experiencia del usuario al utilizar o consultar los reportes.

1.6. Estructura

- En el Capítulo 2 se mencionan antecedentes que incluyen los principales conceptos relacionados al proceso de la Inteligencia de Negocios (dato, información, conocimiento, etc.) y definiciones relacionadas a la generación de reportes con una tabla de datos (jqGrid, Ajax, etc.).

- En el Capítulo 3 se muestran algunas de las herramientas que son utilizadas en las empresas para crear reportes empresariales para Inteligencia de Negocios.
- En el Capítulo 4 se describe el proceso para la generación de los reportes.
- En el Capítulo 5 se muestran las características que solucionan los requerimientos y se muestran algunos ejemplos de los reportes generados.
- En el Capítulo 6 se concluye resumiendo los resultados a los requerimientos con los reportes generados.

Capítulo 2
Antecedentes

En este capítulo se describen los antecedentes, conceptos, definiciones y tecnologías a partir de las cuales se desarrolla la generación de reportes para BI expuesta en este trabajo. Se inicia con los antecedentes que forman parte de la Inteligencia de Negocios (*Bussines Intelligence* o BI) y se finaliza con la descripción del Proceso Unificado Ágil (en el que se basa la metodología para el desarrollo de este proyecto).

Cuando se haga referencia a reportes o *reporting*, significa que se refiere a reportes empresariales computarizados.

2.1. Inteligencia de negocios

A finales de la década de 1960, los primeros indicios de sistemas de información basados en la toma de decisiones comienzan con los Sistemas de Soporte a las Decisiones (DSS) [1].

Las anécdotas y la investigación demuestran que los DSS computarizados pueden proporcionar a los gerentes capacidades analíticas e información que mejore la toma de decisiones. De esta forma, los DSS se han creado para ayudar a los equipos de decisión y a los responsables de la toma de decisiones individuales. Algunos sistemas proporcionan información estructurada directamente a los gerentes. Otros sistemas ayudan a los gerentes y a los especialistas a analizar situaciones usando varios tipos de modelos. Algunos DSS almacenan conocimiento y lo ponen a disposición de los gerentes. Algunos sistemas apoyan la toma de decisiones para grupos grandes y pequeños. Es posible categorizar tres tipos de DSS: los basados en datos, los impulsados por modelos y los basados en el conocimiento [2].

Los DSS basados en datos incluyen organización de archivos y sistemas de informes de gestión, almacenamiento de datos (data warehousing) y sistemas de análisis, Sistemas de Información Ejecutiva (EIS) y Sistemas de Información Geográfica (GIS). Los sistemas de *Inteligencia de Negocios* son ejemplos de DSS basados en datos [2].

La BI se define como el conjunto de procesos, tecnologías y herramientas

necesarias para convertir datos en información, información en conocimiento y el conocimiento en planes que impulsen la acción de la empresa. La BI abarca el almacenamiento de datos, herramientas de análisis de negocio y la gestión de contenidos o conocimientos [4, 5].

Algunas de las principales limitantes a las que se enfrentan las empresas cuando se trata de analizar la información con la que cuentan son [6]:

- Datos que carecen de información - El almacenamiento de datos es relevante dentro de las actividades de una empresa. Entre los datos que usualmente son de interés, se encuentran los del tipo administrativo, por ejemplo: contabilidad, finanzas, ventas, compras, clientes, recursos humanos, informática o cualquier otro que proporcione información de la actividad de la empresa. Sin embargo, almacenar datos no es suficiente si se desea ser competitivo dentro del ramo de las actividades en que ésta se desarrolla. Es también necesario analizar la información para identificar patrones, con la finalidad de maximizar los recursos humanos, los tecnológicos y las estrategias que se apliquen en la toma de decisiones [6].
- Fragmentación - Es el uso de diferentes aplicaciones en cada departamento de una empresa, sin que estas proporcionen información global de las actividades que realiza. Por las características de la BI, la integración de datos no siempre puede realizarse de manera heterogénea, por ello, el manejo de la información fragmentada de una empresa es una limitante en la toma de decisiones de la misma. Así, la fragmentación de datos proporciona una versión parcial de la información real de la empresa, y su aplicación requiere realizar reportes por departamento mucho más detallados. Por lo anterior, en caso de modificaciones dichos reportes requerirán más tiempo en su realización, seguimiento y entrega, lo que se reflejará en un impacto negativo para la empresa [6].
- Manipulación manual - Cuando la empresa no tiene certidumbre de la información que se genera a través de los BI, será necesaria un análisis de negocios y reportes generados manualmente. Este tipo de prácticas

requiere la manipulación de datos para exportarlos a diferentes extensiones, para después manipularlos y generar la información que requiera la empresa. Este proceso típicamente es lento, costoso y poco confiable por la duplicación de trabajo que implica [6].

Todas las limitantes mencionadas impiden el logro completo del éxito empresarial, por lo que uno de los objetivos de la BI es crear alternativas que aproximen a una solución cuando dichos problemas puedan presentarse. La BI ha ido evolucionando y ampliando sus alcances. Las empresas suelen adecuar y utilizar la herramientas de BI existentes o incluso han ido creando nuevas alternativas de acuerdo a sus objetivos cuando las herramientas existentes no cubren las necesidades de consulta [6].

2.1.1. Datos, información y conocimiento

Es primordial para una empresa obtener conocimiento de los datos que, en conjunto, proporcionen información que resulte útil para la toma de decisiones. A continuación se definen datos, conocimiento e información [4].

- Datos: es una colección de elementos o hechos con valor en bruto, utilizados para el cálculo, el razonamiento o la medición. Los datos pueden ser recolectados, almacenados o procesados, pero por sí solos no proporcionan un contexto en el que pueda inferirse algún significado.
- Información: es el resultado de la recopilación y organización de datos de manera que se establezcan relaciones, por lo que proporciona el contexto y significado.
- Conocimiento: es la comprensión de la información basada en patrones reconocidos de una manera que proporciona una idea o aprendizaje de la información.

2.1.2. Estructura de una solución de Inteligencia de Negocios

La estructura para una solución de BI consiste de tres componentes y dos procesos. Los componentes son: el Origen de los datos, un Repositorio consolidado (por ejemplo, un almacén de datos) y la técnica o herramienta para el análisis, explotación o visualización de los datos. Los procesos corresponden a: el proceso de Extracción, Transformación y Carga de los datos (ETL) y el Acceso a los datos [6].

El diagrama de la Figura 2.1 muestra la estructura de una solución de BI.

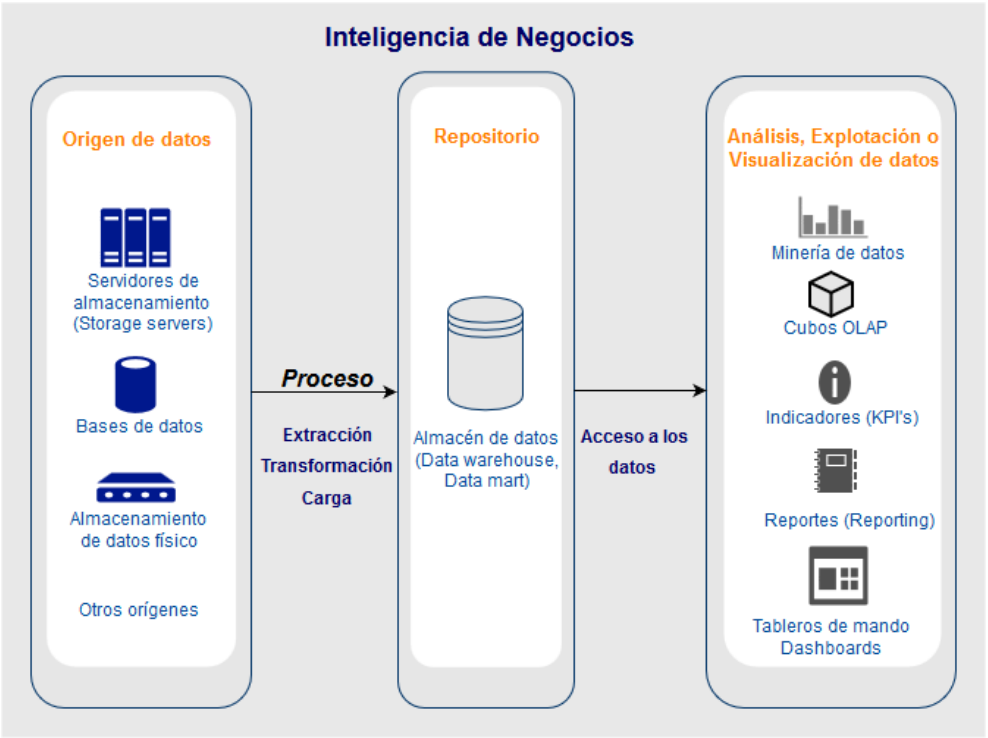


Figura 2.1: Estructura de la Inteligencia de Negocios [6]

Componentes de una estructura de BI

- Origen de datos - Se refiere al lugar donde se encuentran almacenados los datos de producción: Servidores de almacenamiento, Bases de datos,

Dispositivos de almacenamiento físicos y otros tipos de almacenamiento, por ejemplo, archivos (de texto, hojas de Excel, etc.) o sistemas de software (ERP, CRM) [6].

- Repositorio o almacén de datos - También conocido como Data Warehouse, en este tipo de repositorios podemos encontrar datos a través de modelos dimensionales, multidimensionales o tablas, seleccionados por un conjunto de técnicas. Su función es la de consultar, respaldar y organizar la información de la empresa de acuerdo con sus características [7, 8].

- Herramientas de análisis, de explotación y de visualización de la información - Este tipo de herramientas se caracterizan por incluir una interfaz de usuario, que permite interaccionar con la información. Los datos revisados pueden representarse a través de gráficas, lo que permite obtener indicadores de gestión que se generan a partir de reportes (dashboards), cubos, procesamiento analítico en línea (OLAP), indicadores de desempeño para la realización de actividades (KPI) y minería de datos[6].

Procesos de una solución de BI

- Extracción, transformación y carga de datos (ETL) - Es un conjunto de procesos que permite el almacenamiento de datos, la forma en que estarán definidos los campos y las fuentes que se requieran cualquier tipo de modificación [6].

- Acceso a los datos - Este proceso se realiza mediante técnicas o herramientas analíticas y de presentación. Como cuadros de mando con gráficos interactivos, tablas y mapas de datos. Las técnicas están guiadas por un conjunto definido de procedimientos y reglas [13]. Las herramientas permiten habilitar componentes, administrar consultas, monitorear procesos, realizar cálculos y establecer métricas [6]

2.1.3. Técnicas de Inteligencia de Negocios

La técnica de BI a elegir consiste en aquella que cubra las necesidades de análisis, visualización o explotación de los datos para los objetivos establecidos por la empresa [6].

Las técnicas de BI más utilizadas por las empresas

- Minería de datos - La minería de datos es el proceso de descubrir conocimiento en grandes conjuntos de datos [27]. Consiste en realizar un análisis de datos para encontrar patrones en las relaciones entre ellos y crear un modelo automatizado para predecir el comportamiento del negocio [9].
- Cubos OLAP (acrónimo de OnLine Analytical Processing) - Un cubo OLAP –procesamiento analítico en línea– funciona particularmente con grandes conjuntos de datos al reducir el tiempo necesario para acceder y visualizar los datos. El rendimiento se mejora porque los datos se resumen en medidas y están predefinidos en jerarquías. Los usuarios pueden revisar los datos desde múltiples ángulos para que puedan responder rápidamente preguntas sin esperar resúmenes de datos o ejecutar varios informes en la misma fuente. Los datos también se pueden proteger a niveles dentro del cubo, por lo que una fuente de datos puede dar servicio a muchos usuarios diferentes en la organización [11].
- Indicadores clave de rendimiento (key performance indicators o KPI) - Generalmente se utilizan para indicar un objetivo que consta de varios valores: real, objetivo, varianza respecto al objetivo y tendencia. Con mucha frecuencia, los KPI se expresan y se muestran gráficamente, por ejemplo, como semáforos de diferentes colores (rojo, amarillo o verde). Los KPI suelen incluir capacidades de desglose que permiten a los encargados de la toma de decisiones revisar los datos detrás del KPI. Los KPI se pueden implementar como parte de un sistema OLAP, y a menudo forman parte de los sistemas de reportes, que se encuentran principalmente en el informe de tableros o cuadros de mando. Es

bastante común que los requisitos del negocio incluyan lo último como parte de una estrategia centralizada de gestión del rendimiento. Es posible crear KPI desde casi cualquier tipo de fuente de datos, como cubos OLAP o libros de Excel [10].

- Reportes de inteligencia de negocios - Los reportes de inteligencia de negocios son generados por herramientas que preparan informes cuantitativos que incluyen números, tablas o gráficas empresariales [5].
- Tableros de mando (dashboards) - Son una descripción visual de los datos corporativos esenciales. Pueden mostrar muchas instantáneas de datos posibles de múltiples aspectos de una empresa, o mostrar un conjunto muy específico de métricas relacionadas con una área comercial o departamento. Un dashboard puede estar compuesto por múltiples elementos, que algunos también denominan "widgets" o visualizaciones. Puede incluir: KPIs, tablas, gráficas, mapas o imágenes. Estos elementos centrales a menudo se pueden combinar para producir múltiples variaciones en un tema [12].

2.2. Reportes de inteligencia de negocios

En la época actual, la información es la clave para obtener una ventaja competitiva en el mundo de los negocios. Para que una empresa se mantenga competitiva, los gerentes y encargados de tomar las decisiones requieren de un acceso rápido y fácil a la información que es útil y valiosa para la empresa. Una forma de solucionar este problema es a través del uso de los sistemas de reporting como un sistema de información [14].

Cuando se habla de reportes en BI, se habla en dos sentidos. Primero, a la definición estricta de reporting. Y, segundo, lo que es "reporting"tomado en un sentido más general [26]. En el primer sentido, la presentación de reportes es el arte de recopilar datos de diversas fuentes y presentarlos a los usuarios finales de una manera que sea comprensible y esté lista para su análisis. En el segundo sentido, reporting significa presentar datos e información, por lo

que también incluye un análisis; en otras palabras, permite a los usuarios finales tanto ver como comprender los datos y actuar en consecuencia [26].

Los reportes se pueden clasificar de diferentes maneras. Una de ellas es diferenciarlos por el rol de quien genera el reporte: los reportes administrados son preparados por personal técnico como desarrolladores; los reportes ad hoc son dominio del usuario final no técnico. Otra forma de clasificar los reportes es mediante la identificación de suscaracterísticas más importantes como tablas de datos, informes de tablas cruzadas, características de visualización, etcétera [26].
En la siguiente subsección se mencionan de manera general algunas características de los reportes de inteligencia de negocios.

2.2.1. Características de los reportes

Un sistema de reporting o de gestión de reportes facilita la distribución de la información a los distintos niveles de la estructura organizativa y facilita a cada tipo de usuarios la información que requieren según las necesidades de cada momento. Para diseñar e implantar dicho sistema se requiere realizar previamente un estudio de las necesidades de información y de las bases de datos existentes [14].

Por lo anterior, podemos decir que un sistema de reporting permite la integración de datos de forma práctica y flexible a cualquier base de datos o el software que maneja la empresa, de tal manera, que pueda relacionarse con otra base de datos para facilitar su consulta y mejorar las opciones en la toma de decisiones [14].

Otra ventaja de los sistemas de reporting es la comparación de información a lo largo del tiempo, ya que permite realizar análisis de tendencias (recopilación histórica) y llevar un mejor control de los indicadores de gestión por periodo [14]. Este tipo de sistemas es considerado dinámico, porque puede compartirse entre usuarios con la finalidad de consulta, lo que permite tener la información disponible en todo momento. Entre sus ventajas se distingue la disminución de errores en la integración de datos por la manera

en que se genera la información ya que sólo el administrador puede modificar las bases de datos [14].

Si una empresa busca que su información sea comprensible y los datos estén disponibles para su análisis, el sistema de reporting será un elemento indispensable para la recopilación, presentación de datos y respaldo de información relevante para la empresa que permita justificar la toma de decisiones [26].

2.3. Conceptos, definiciones y tecnologías para el desarrollo de los reportes

2.3.1. Tabla de datos

Una tabla de datos muestra los valores desde una fuente de almacenamiento, donde una columna representa un campo y una fila representa un registro. El control de la tabla de datos permite realizar funciones tales como modificación, selección y ordenamientos de los elementos [15]. Ademaás, es un componente que muestra información en forma de tabla desde una fuente de almacenamiento, generalmente desde una base de datos. El controlador -o *plugin*- utilizado para la implementación es el controlador jqGrid, cuyas bibliotecas incluyen una intezfaz web, búsquedas personalizadas, filtros y paginación, entre otras funciones.

2.3.2. jqGrid

jqGrid es un controlador de jQuery creado por Tony Tomov[1], para la visualización y edición de datos en tablas web. Es de código abierto y se distribuye bajo la licencia del MIT. Su objetivo ha sido ahorrar código para insertar, modificar y eliminar datos desde la base de datos. jqGrid es un ejemplo de tabla de datos y es un controlador *JavaScript* habilitado para

[1]En Trirand Inc.

Ajax, que se puede integrar con cualquier tecnología de servidor. Posee una amplia cantidad de opciones a configurar, eventos y métodos para enriquecer las tablas [15].

2.3.3. Ajax

Ajax es el acronimo de "Asynchronous JavaScript and XML". Es una técnica que combina diversas tecnologías Web existentes [22]:

- XHTML, HTML y CSS, se utilizan para crear una presentación visual basada en estándares para crear una página web.
- DOM (Document Objet Model), es la representación de los objetos dentro del documento.
- XML o JSON, es el formato de los datos que son transferidos entre el servidor y el cliente.
- XMLHttpRequest, es el objeto que sirve para recuperar datos desde el servidor.

JavaScript es el encargado de unir todas las tecnologías mencionadas. En las siguientes subsecciones se describe brevemente cada componete.

2.3.4. XHTML, HTML y CSS

Algunas de las tecnologías de mayor uso en la construcción de páginas web son: el lenguaje de marcado de hipertexto (HTML) y las hojas de estilo en cascada (CSS), ambos son considerados la base para crear gráficos y secuencias de comandos de páginas y aplicaciones web. A grandes rasgos, el primero ayuda a generar la estructura de la futura página y con la segunda se realiza el diseño (aural y visual). Ambos pueden aplicarse a diferentes tipos de dispositivo electrónico con conexión a internet [28].

El lenguaje HTML describe las estructuras de las páginas web y los elementos que lo integran son [28]:

- La realización de textos con encabezados, que pueden incluir tablas, listas o imágenes.

- El diseño de formularios para la realización de transacciones con servicios del tipo remoto, que involucren la compra de productos o servicios, buscadores de información o reservas

- La recuperación de información. Si la configuración lo permite es posible guardar la información a través de enlaces de hipertexto o por selección manual.

- La inclusión de vídeos, clips de audio, hojas de cálculo, simulaciones u otras aplicaciones que pueden ser utilizadas dentro de la página web.

El marcado de sus elementos es otra de las características asociadas al lenguaje HTML. Por ejemplo, la estructura del idioma de la página puede etiquetarse a partir de fragmentos de su contenido como "párrafo.º "listado", "tabla", entre otros [28].

Hay una variante del lenguaje HTML que se llama XHTML. Este lenguaje utiliza en su sintaxis un lenguaje de marcado extensible (XML) y aunque tiene los mismos elementos que HTML es ligeramente diferente, porque XML le permite utilizar herramientas como: transformaciones de lenguaje de marcado extensible (XSLT)) [28].

El lenguaje que se utiliza para describir la presentación de páginas web se conoce como hoja de estilo de cascada (CSS). Incluye atributos que le dan forma a la página web, por ejemplo, el color de la letra, el tipo de fuente o los marcos. Además, esta herramienta facilita adaptar el diseño de la página web a los diferentes tipos de dispositivos electrónicos, variando el tamaño de la pantalla del dispositivo o el tipo de impresora para la impresión de documentos [28].

La separación de estructuras para la presentación de páginas web es una de las ventajas del lenguaje CSS, ya que no depende del lenguaje HTML e interactúa con lenguajes de marcado del tipo XML. Esto, facilita la perso-

nalización, edición y mantenimiento de la página, así como el intercambio de estilos entre páginas web [28].

2.3.5. Modelo de objeto de documento

En una página web, los objetos que componen la página (o documento) están representados como una estructura de árbol. Algunos navegadores pueden mostrar representaciones de este tipo de estructura. JavaScript considera que cada uno de los elementos en el árbol del documento son objetos, y puede utilizar a Javascript para manipular esos objetos. La representación de los objetos dentro del documento se llama Modelo de objeto de documento (DOM). Cada uno de los objetos en el árbol también se le llama un nodo de árbol. Se puede usar Javascript para modificar cualquier aspecto del árbol, como agregar, acceder, cambiar, eliminar nodos en el árbol. Cada objeto en el árbol es un nodo. Si el nodo contiene una etiqueta HTML, se le denomina nodo de elemento. De lo contrario, se le denomina nodo de texto [21].

2.3.6. JSON

Dentro de los formatos livianos de intercambio de datos se encuentra JavaScript Object Notation (JSON). Es uno de los más conocidos por su facilidad de lectura y escritura para el programador, y fácil de generar para las computadoras. Sus componentes están basados en el lenguaje JavaScript (Java), lo que permite al programar usarlo de manera independiente del lenguaje en que este programada la página web. JSON utiliza algunas convenciones familiares a los programadores que se asocian a lenguajes como: C, C++, C-Sharp, Java, JavaScript, Perl y Python entre otros, lo que facilita el intercambio de datos [29].

Las estructuras en las que está basado JSON son [29]:

- Una colección de pares del tipo nombre/valor, en algunos lenguajes de programación dicha colección puede definirse dependiendo el lenguaje

elegido. Entre los tipos más comunes se encuentra: objeto, diccionario, lista de claves o matriz asociativa.

- Una lista ordenada de valores, en este caso lo usual es una matriz, vector o secuencia ya que son los elementos más usados en los lenguajes de programación para listas.

Por otro lado, las formas en las que está basado JSON son [29]:

- Un *objeto* es un conjunto desordenado de pares de nombre / valor. Un objeto comienza con "{ "(llave izquierda) y termina con "}"(llave derecha). A cada nombre le siguen ":"(dos puntos) y los pares nombre / valor están separados por ","(coma) [29].
- Un *arreglo* es una colección ordenada de valores. Un arreglo comienza con "["(corchete izquierdo) y termina con "]"(corchete derecho). Los valores están separados por ","(coma) [29].
- Un *valor* puede ser una cadena entre comillas dobles, o un número, o verdadero o falso o nulo, o un objeto o una matriz. Estas estructuras se pueden anidar [29].
- Una *cadena* es una secuencia de cero o más caracteres Unicode, encerrados entre comillas dobles, usando escapes de barra invertida. Un caracter se representa como una cadena de caracteres individuales. Una cadena es muy similar a una cadena en C o en Java [29].
- Un *número* es muy parecido a un número en C o en Java. A excepción de los formatos octales y hexadecimales que no se usan [29].

2.3.7. Netbeans

NetBeans es un entorno de desarrollo integrado (IDE) que permite desarrollar rápida y fácilmente aplicaciones de escritorio, móviles y Java web, así como aplicaciones HTML5 con HTML, JavaScript y CSS. El IDE también

proporciona un gran conjunto de herramientas para desarrolladores PHP y C / C ++. Es de código abierto y gratuito y tiene una gran comunidad de usuarios y desarrolladores en todo el mundo [30].

2.3.8. Apache Tomcat

Apache Tomcat es un servidor/contenedor de Servlet/JSP de código abierto de la fundación Apache. Consiste, esencialmente, en lo siguiente [24]:

- Un servidor web.
- Una aplicación que ejecuta servlets de Java.
- Una aplicación que convierte páginas y documentos JSP en servlets de Java

El software Apache Tomcat es una implementación de de las tecnologías Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket [38].

2.3.9. Lenguaje SQL

SQL (Structured Query Language) es un lenguaje no procedural que es utilizado para manejar datos en sistemas manejadores de bases de datos relacionales. Es un conjunto de comandos usados por los programadores y desarrolladores de aplicaciones para acceder a datos almacenados en cualquier base de datos[16]. Por ejemplo, Oracle es un sistema de gestión, o manejador, de base de datos relacionales [17].

2.3.10. Elementos de Java

Java Enterprise Edition

El objetivo de la plataforma Java EE es proporcionar a los desarrolladores un gran conjunto de APIs, acortando el tiempo de desarrollo, reduciendo la

complejidad de las aplicaciones y mejorando el rendimiento de las aplicaciones [34].

La plataforma Java EE se desarrolla a través del Java Community Process (JCP), que es responsable de todas las tecnologías Java. Grupos de expertos, compuestos por integrantes interesados en el tema, han creado solicitudes de especificación de Java (JSR) para definir las distintas tecnologías Java EE. El trabajo de la Comunidad Java bajo el programa JCP ayuda a garantizar los estándares de estabilidad de la tecnología Java y la compatibilidad multiplataforma [34].

Java EE es el estándar en el software empresarial. Cada versión integra nuevas características que se alinean con las necesidades de la industria, mejora la portabilidad de la aplicación y aumenta la productividad del desarrollador [35].

Java SE Development Kit

El Java SE Development Kit (JDK) es un entorno de desarrollo para crear aplicaciones, applets y componentes, utilizando el lenguaje de programación Java. Este incluye herramientas útiles para desarrollar y probar programas escritos en dicho lenguaje de programación y que se ejecutan en la misma plataforma [33].

Java Servlet

Los servlets son la tecnología de la plataforma Java para ampliar y mejorar los servidores web. Los servlets proporcionan un método basado en componentes e independiente de la plataforma para crear aplicaciones basadas en Web, sin las limitaciones de rendimiento de los programas CGI. A diferencia de los mecanismos de extensión de servidor patentados (como la API Netscape Server o los módulos Apache), los servlets son independientes del servidor y de la plataforma. Ésto permite seleccionar una estrategia de "lo mejor de tu clase"para los servidores, plataformas y herramientas [31].

Los servlets tienen acceso a toda la familia de la API de Java. También

pueden acceder a una biblioteca de llamadas específicas de HTTP y recibir todos los beneficios del lenguaje Java, incluida la portabilidad, el rendimiento, la reutilización y la protección contra fallas [31].

Los servlets actualmente son una opción popular para crear aplicaciones web interactivas. Los contenedores de servlets de terceros están disponibles para el servidor web Apache, Microsoft IIS y otros. Los contenedores de servlets suelen ser un componente de los servidores web y de aplicaciones, como BEA WebLogic Application Server, IBM WebSphere, Sun Java System Web Server, Sun Java System Application Server y otros [31].

La tecnología JSP es una extensión de la tecnología servlet creada para admitir la creación de páginas HTML y XML. Hace que sea más fácil combinar datos de plantilla fijos o estáticos con contenido dinámico. Incluso si se está cómodo escribiendo servlets, hay varias razones convincentes para investigar la tecnología JSP como complemento al trabajo existente [31].

JavaServer Pages

La tecnología JavaServer Pages (JSP) permite a los desarrolladores y diseñadores web desarrollar rápidamente y mantener fácilmente páginas web dinámicas y ricas en información que aprovechan los sistemas comerciales existentes. Como parte de la familia de tecnología Java, la tecnología JSP permite el desarrollo rápido de aplicaciones basadas en web que son independientes de la plataforma. La tecnología JSP separa la interfaz de usuario de la generación de contenido, lo que permite a los diseñadores cambiar el diseño general de la página sin alterar el contenido dinámico subyacente [32].

JDBC

La API Java Database Connectivity (JDBC) es el estándar de la industria para la conectividad independiente de la base de datos entre el lenguaje de programación Java y una amplia gama de bases de datos SQL y otras fuentes de datos tabulares, como hojas de cálculo o archivos planos. JDBC proporciona una API de nivel de llamada para el acceso a bases de datos

basadas en SQL [36].

La tecnología JDBC permite utilizar el lenguaje de programación Java para explotar las capacidades "Write Once, Run Anywhere que se traduce como "Escriba una vez, ejecute donde sea para aplicaciones que requieren acceso a datos empresariales. Con un controlador habilitado para tecnología JDBC, puede conectar todos los datos corporativos incluso en un entorno heterogéneo [36].

JavaScript

JavaScript (a menudo abreviado como JS) es un lenguaje liviano, interpretado y Orientado a Objetos con funciones de primera clase. Es conocido como el lenguaje de scripting para páginas web. También se usa en muchos entornos que no son de navegador. Es un lenguaje basado en prototipos y multi-paradigma, es dinámico y admite estilos de programación orientados a objetos, imperativos y funcionales [37].

JavaScript se ejecuta en el lado del cliente web, que se puede utilizar para diseñar y programar la forma como se deben comportar las páginas web al invocar un evento. JavaScript es un lenguaje de fácil aprendizaje, ampliamente utilizado para controlar el comportamiento de la página web [37].

Contrario a la idea popular, JavaScript no es "interpretado por Java". En pocas palabras, JavaScript es un lenguaje dinámico, que admite la construcción de objetos basados en prototipos. La sintaxis básica es similar tanto a Java como a C ++ para reducir el número de conceptos nuevos requeridos para aprenderlo. Las construcciones de lenguaje, como los enunciados if, for y while loops, y switch y try ... catch funcionan casi de la misma manera que en estos lenguajes [37].

JavaScript puede funcionar como un lenguaje Orientado a Objetos y a procedimientos. Los objetos se crean mediante la vinculación de métodos y propiedades. A diferencia de las definiciones de clases sintácticas comunes en lenguajes compilados como C ++ y Java. Una vez que se ha construido

un objeto, se puede usar como un plano (o prototipo) para crear objetos similares[37].

Las capacidades dinámicas de JavaScript incluyen construcción de objetos en tiempo de ejecución, listas de parámetros variables, variables de función, creación de scripts dinámicos (a través de eval), introspección de objetos (vía para ... en) y recuperación de código fuente (los programas de JavaScript pueden descompilar los cuerpos de funciones en su texto original) [37].

2.3.11. Aplicación Web

En la plataforma Java EE, los componentes web proporcionan las capacidades de extensión dinámica para un servidor web. Los componentes web pueden ser servlets de Java, páginas web implementadas con tecnología JavaServerFaces, puntos finales de servicios web o páginas JSP. La Figura 2.2 ilustra un ejemplo de la interacción entre un cliente web y una aplicación web que utiliza un servlet (los clientes y los servlets se comunican por medio de los objetos HttpServletRequest y HttpServletResponse). El cliente envía una solicitud HTTP al servidor web. Un servidor web que implementa la tecnología Java Servlet y JavaServer Pages convierte la solicitud en un objeto HTTPServletRequest. Este objeto se entrega a un componente web, que puede interactuar con los componentes de JavaBeans o una base de datos para generar contenido dinámico. El componente web puede generar un HTTPServletResponse, o puede pasar la solicitud a otro componente web. Un componente web finalmente genera un objeto HTTPServletResponse. El servidor web convierte este objeto a una respuesta HTTP y lo devuelve al cliente [39].

Una aplicación web consiste de componentes web: archivos de recursos estáticos, como imágenes y CSS, y clases y bibliotecas de ayuda. El contenedor web proporciona muchos servicios de soporte que mejoran las capacidades de los componentes web y los hacen más fáciles de desarrollar. Sin embargo, dado que una aplicación web debe tener en cuenta estos servicios, el proceso para crear y ejecutar una aplicación web es diferente al procesp de crear las

clases Java tradicionales e independientes. El proceso para crear, implementar y ejecutar una aplicación web se puede resumir de la siguiente manera [39]:

- Desarrollar el código del componente web.
- Desarrollar el descriptor de despliegue de la aplicación web, si es necesario.
- Compilar los componentes de las aplicaciones web y las clases auxiliares a las que hacen referencia los componentes.
- Opcionalmente, empaquetar la aplicación en una unidad desplegable.
- Implementar la aplicación en un contenedor web.
- Acceder a una URL que haga referencia a la aplicación web.

2.4. Proceso Unificado Ágil

Una metodología de desarrollo de software es una forma de administrar un proyecto de desarrollo. Contempla la solución a problemas como: la selección de características para su inclusión en la versión correspondiente, cuándo se lanza el software, quién trabaja en qué y qué pruebas se realizan. Ninguna metodología es mejor para todas las situaciones u organizaciones. En la práctica, cada organización implementa su gestión de proyectos de desarrollo de software de una manera diferente. A menudo es ligeramente diferente de un proyecto a otro [20].

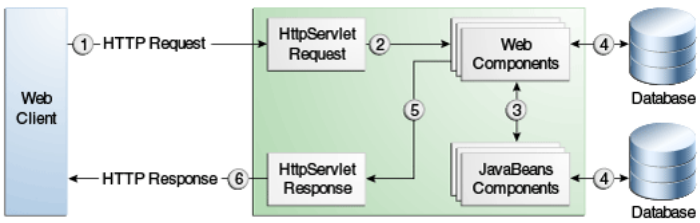


Figura 2.2: Diagrama de manejo de solicitudes de una aplicación web [39].

El Proceso Unificado Ágil (AUP) es una metodología de desarrollo de software basada en el Proceso Unificado Rational (RUP) de IBM, cuyo ciclo de vida es secuencial en sus fases e iterativo en sus disciplinas (o flujos de trabajo). Liberando entregables incrementales en el tiempo (Figura 2.3) [18].

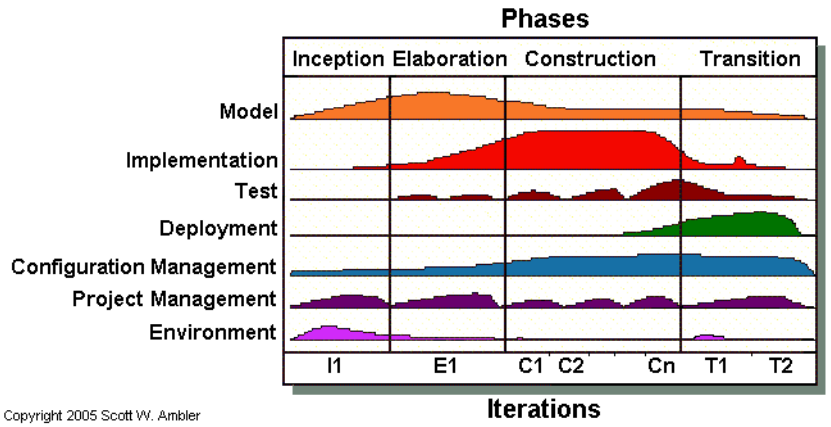


Figura 2.3: Fases y Actividades del Proceso Unificado Ágil [18]

2.4.1. Flujos de trabajo del Proceso Unificado Ágil

Los flujos de trabajo (o disciplinas) de AUP son ejecutados definiendo las actividades que el equipo de desarrollo lleva a cabo para construir, validar y liberar software funcional, que cumpla con las necesidades de los involucrados [19].

AUP combina los flujos de trabajo de modelos de negocios, requisitos y análisis y diseño de RUP en un único flujo de trabajo llamado Modelo [19, 18]:

- Modelado. Este flujo ayuda a comprender la organizacin de la empresa, tambin se utiliza para identificar los puntos vulnerables en proyectos y la identificacin de posibles soluciones para dichos puntos [18].
- Implementación. Este flujo yiene como objetivo la transformacin de modelos a cdigo ejecutable, con la finalidad de realizar pruebas bsicas del tipo unitario [18].

- Pruebas. Es un flujo que ayuda a la evaluacin objetiva para garantizar la confiabilidad y calidad del sistema. Es decir, a travs de las pruebas es posible identificar errores, verificar cualidades y comprobar funcionabilidad del sistema [18].
- Despliegue. Este tipo de flujo tiene como objetivo la planificacin y estrategias necesaria para la entrega del sistema, con la finalidad de tenerlo disponible para el usuario final [18].
- Administración de la configuración. Se encarga del seguimiento, administracin, edicin y compilacin de las distintas versiones del sistema desde el inicio y hasta el final del proyecto. Con el objetivo de administrar el acceso a los productos generados por el proyecto [18].
- Administración del proyecto. Es un flujo que se encarga de dirigir todas las actividades relacionadas con el proyecto, entre estas se encuentran: la gestin de riesgos, direccin de personal, la asignacin de tareas, seguimiento del progreso del proyecto, la coordinacin entre las autoridades relacionadas con los recursos presupuestales (internas o externas a la empresa), y la entrega final del proyecto [18].
- Ambiente. Este flujo garantiza que el equipo involucrado en el desarrollo del sistema tenga las herramientas (software y hardware) disponibles, y las guas de procesos sean los adecuados [18].

2.4.2. Fases del Proceso Unificado Ágil

Las fases y sus principales actividades a lo largo de un proyecto son [18]:

- Inicio. Identificar el alcance inicial del proyecto, una arquitectura potencial para su sistema, obtener fondos iniciales del proyecto y la aceptación de las partes interesadas.
- Elaboración. Crear un prototipo arquitectónico del sistema y probar su viabilidad. Comenzar a configurar el entorno para la fase de construcción mediante la compra de hardware, software y herramientas.

- Construcción. Desarrollar software en funcionamiento de forma regular e incremental que cumpla con las necesidades de mayor prioridad de las partes interesadas del proyecto.
- Transición. Validar e implementar el sistema en su entorno de producción.

2.5. Resumen

En este capítulo se han visto las definiciones y conceptos más relevantes para el desarrollo de los reportes. Se define lo que son datos, información, conocimiento, Inteligencia de Negocios; la estructura de una solución de BI y sus principales herramientas. Se definen los reportes de BI y se mencionan sus caracteristcas generales. Se define lo que signifca una tabla de datos. Se presenta un resumen de las tecnologías jqGrid, Ajax, XHTML, CSS, DOM, JSON, Netbeans, Apache Tomcat (servidor de aplicaciones), Lenguaje SQL, los principales elemetos de Java (Java EE, JDK, Java Servlet, JSP, JDBC, JavaScript), Aplicación Web y el Proceso Unificado Ágil AUP.

Capítulo 3

Trabajo relacionado: Software para generar reportes de inteligencia de negocios

En este capítulo se presentan de manera general las principales características de algunas aplicaciones de software disponibles actualmente para el mercado empresarial y que proporcionan una opción para la presentación de reportes dentro de la Inteligencia de Negocios (BI).

Además, se revisa la aplicación HP Business Service Manager (BSM), que es la utilizada por la Empresa como medio de control de acceso a los sistemas de reportes (pero no como una fuente de desarrollo de los mismos).

3.1. Business Service Manager

La aplicación de software HP Business Service Manager (BSM), desarrollada por Hewlett Packard (HP), es una herramienta de gestión que ayuda a las empresas a optimizar el rendimiento y la disponibilidad de las aplicaciones en producción y resuelve proactivamente los problemas que surgen, ayudando así a las aplicaciones críticas de producción para que funcionen según se requiera y entreguen resultados de índole empresarial [25].

BSM consiste en un conjunto integrado de aplicaciones para el monitoreo en tiempo real del rendimiento y la disponibilidad desde una perspectiva empresarial. Esto incluye la administración del nivel de servicio, la administración del usuario final, el manejo de eventos, la administración de la disponibilidad del sistema, y los reportes y alertas personalizados. BSM se basa en una base común de flujos de trabajo compartidos, servicios de administración e informes, activos compartidos y experiencia [25].

La plataforma BSM consiste en servidores y componentes propietarios, fuentes de datos, herramientas de scripting y servidores de terceros, como bases de datos y servidores de correo, que se configuran en el entorno de red empresarial. BSM está impulsado por un conjunto de servidores que son responsables de ejecutar las aplicaciones, facilitando la administración del sistema, el manejo de datos, informes y alertas [25].

Es necesario instalar los servidores de BSM en una o más computadoras en el entorno de red empresarial [25]:

- Servidor BSM Gateway. Responsable de ejecutar las aplicaciones BSM, producir informes, operar las áreas de administración, recibir muestras de datos de los recopiladores de datos y distribuir estos datos a los componentes relevantes de BSM, y dar soporte al bus. Para trabajar con BSM, la máquina del servidor Gateway debe ejecutar servidores web.

- Servidor de procesamiento de datos de BSM. Responsable de agregar datos, ejecutar el motor de lógica de negocio (Business Logic Engine, BLE) y controlar el modelo de servicio en tiempo real (Real Time Service Model,RTSM).

BSM ayuda a los clientes a reducir el tiempo promedio de detección (MTTD) y el tiempo de inactividad del usuario al detectar proactivamente los problemas de rendimiento y disponibilidad de las aplicaciones, ayudando a escalar los problemas al departamento correcto con la prioridad correcta, así como la resolución de problemas de rendimiento antes de que se incumplan los objetivos de nivel de servicio. Esto ayuda a las organizaciones a alcanzar el objetivo de maximizar el valor de las operaciones de TI y reducir el Costo total de propiedad (TCO) de la infraestructura de TI [25].

3.2. Pentaho

Pentaho es un conjunto de aplicaciones que permite a los usuarios empresariales acceder, descubrir y mezclar todos los tipos y tamaños de datos. Con un espectro de análisis cada vez más avanzados, desde los informes básicos hasta el modelado predictivo. Los usuarios pueden analizar y visualizar datos a través de múltiples dimensiones, minimizando la dependencia de TI [40].

Características [40]:

- Análisis visual interactivo. Los usuarios son autosuficientes y pueden acceder de inmediato, analizar y visualizar cualquier dato. Proporciona análisis visual interactivo con exploración, filtrado de lasas, zoom

y resaltado de atributos para una mayor comprensión. Biblioteca de visualizaciones interactivas - incluyendo mapas geo-cartográficos, cuadriculas térmicas y gráficos de dispersión / burbuja.

- Tarjetas de instrumentos gráficos y responsables. Son tableros interactivos que ofrecen a los usuarios empresariales indicadores clave de rendimiento en una interfaz gráfica para mejorar el rendimiento de la organización. Incluye capacidad de diseño de tablero de arrastrar y soltar basado en web, que incluye navegación, perforación y una biblioteca de controles de filtro.
- Soluciones integrales. Sus capacidades de generación de informes abarcan desde informes interactivos de autoservicio hasta informes de empresas de alto volumen y alto formato. Informes interactivos intuitivos basados en la web para usuarios empresariales. Diseñador gráfico de informes empresariales para usuarios avanzados. Salida en formatos populares: HTML, Excel, CSV, PDF y RTF. Almacenamiento en caché en memoria para resultados rápidos. Publica informes directos para NoSQL [25].
- Gestión y administración optimizadas. Incluye seguridad analítica, permisos de contenido, control de versiones, bloqueo y caducidad. Fiabilidad, respaldo y recuperación de información. Integración de datos con rearranque y restauración de tareas. Monitoreo y auditoría de rendimiento con generación de informes.

3.3. jasperReports

jasperReports es un motor de informes desarrollado por Jaspersoft y distribuido bajo una licencia de código abierto. Incluye a iReport como el diseñador de informes históricos de Jaspersoft. Estas herramientas han existido desde 2001 y se utilizan ampliamente en los segmentos de informes de muchas aplicaciones empresariales [41].

El motor de JasperReports puede generar informes en formatos PDF,

HTML, XML, CSV, RTF, XLS, ODT, ODS, Flash, DOCX, XLSX y TXT. Utiliza JFreeChart para generar gráficos y puede integrarse en cualquier aplicación desarrollada en Java [41].

Además de las bases de datos clásicas, también soporta servidores analíticos multidimensionales, lo que permite aprovechar al máximo las posibilidades de un servidor Mondrian, directamente en un informe de JasperReports [41].

Actualmente, el diseñador de informes iReport ha sido reemplazado por Jaspersoft Studio, un plugin de Eclipse, que ahora es el diseñador oficial de informes y constructor de JasperReports. JasperSoft continua apoyando iReport, arreglando errores importantes, pero no integra nuevas características. La generación de informes con iReport es compatible con JasperSoft Studio.

JasperReports tiene una versión para la comunidad (AGPL) y una edición comercial (bajo una licencia propietaria). La edición comercial proporciona esencialmente una biblioteca de gráficos más avanzados [41].

3.4. Discusión

En la Empresa se utilizaba la herramienta BSM para gestionar el rendimiento de las aplicaciones, sistemas, redes y recursos de almacenamiento y como un medio para mostrar recursos externos de diversas aplicaciones de sotware. No la utilizaban para la generación de reportes debido a la curva de aprendizaje necesaria, a la inversión para capacitar a un desarrollador y a las limitaciones para crear desarrollos detallados.

Pentaho es una herramienta que sobrepasa las necesidades de generación de informes de la Empresa. Una de las mayores ventajas de esta aplicación es que permite realizar tableros interactivos, pero dichos tableros ya son generados con la herramienta BSM. Además, se cuentan con contratos y compromisos con Hewlett Packard para la herramienta BSM y no es posible destinar recursos a otras herramientas de BI.

jasperReports es una aplicación que requiere poseer conocimientos es-

pecíficos de la herramienta, además de los conocimientos en desarrollo en Java. Es decir, para realizar los reportes solicitados era necesario destinar tiempo en la curva de aprendizaje para desarrollar los complementos e integrarlos.

Por lo anterior, las herramientas disponibles poseen varias ventajas pero tienen limitaciones al incluir requerimientos muy específicos o detallados. Por lo que no proporcionan una respuesta a las necesidades de la empresa. La solución que permite cubrir las necesidades para la generación de los reportes solicitados es utilizar tecnologías existentes. Donde la curva de aprendizaje no implique demasiado tiempo invertido y no haya costos extras de uso de licencias de software u otros recursos adicionales para la Empresa.

3.5. Resumen

En este capítulo se han revisado las principales características de las herramientras más comunes para reporting en la Inteligencia de Negocios, que son BSM de HP, Pentaho y jasperReports.

Se discute que el principal inconveniente de este tipo de herramientas es que no siempre cubren las necesidades de las empresas, como la creación de reportes detallados y el tiempo solicitado de desarrollo. Además, suelen requerir inversión para su instalación, capacitación y soporte.

Capítulo 4
Generación de reportes con tablas de datos para Inteligencia de Negocios

En este capítulo, se expone como objetivo principal el desarrollo para la generación de reportes con una tabla de datos para la Inteligencia de Negocios (BI). Comenzando con una breve explicación de la forma como se trabaja en la Empresa (multinacional, dedicada a la fabricación, comercialización y distribución de bebidas y aperitivos) que cuenta con distintos equipos de trabajo que siguen diversos flujos de actividades y bajo sus propias metodologías.

La metodología de desarrollo a seguir para la generación de los reportes se basa en el Proceso Unificado Ágil (AUP) (veáse la Sección 2.4). Los flujos de trabajo se adaptan de acuerdo a las necesidades de este proyecto y se explican en el siguiente orden:

1. Modelado
2. Implementación
3. Pruebas
4. Despliegue

El proceso de desarrollo se describe separando cada uno de los flujos de trabajo de AUP (Figura 2.3) y se mencionan en cada flujo las actividades realizadas en alguna de las fases siguientes:

a. Inicio
b. Elaboración
c. Construcción
d. Transición

Las actividades que se describen en cada flujo de trabajo y fase del proyecto son las realizadas por el *Desarrollador* (que corresponde al creador del presente trabajo). El flujo de trabajo de la *Administración del proyecto* se omite debido a que no son actividades del desarrollador sino de otros roles del equipo. La mayor interacción del desarrollador se realiza con el administrador de base de datos, el equipo de pruebas IT y el gerente de negocio.

Los flujos de trabajo de *Administración de la configuración* y *Ambiente* se describen en una misma sección como Configuración y Ambiente. En AUP, dichos flujos se describen después del flujo de *Despliegue* pero, en este caso, se realizan antes iniciar el desarrollo y pueden consultarse en el Apéndice A

de este trabajo.

4.1. Modelado

Las actividades de modelado de negocio y análisis de requerimientos del flujo de *Modelado* tienen mayor relevancia en la fase de Inicio. Así mismo, las actividades de diseño en la fase de Elaboración. Y, en menor grado, en la fase de Construcción, cuando existan casos en que deban modificarse las necesidades de negocio por algún factor ajeno a los acuerdos iniciales.

4.1.1. Modelado de negocio

En la fase de Inicio se analizan con el gerente de negocio los reportes utilizados hasta el momento en la Empresa. Se concluye que los reportes originales (ver Figura 4.1) cuentan con una lista estática (no modificable) para los filtros de búsqueda, lo cual dificulta las consultas respecto a las necesidades de disposición de la información. El lenguaje de programación utilizado para dichos reportes (lenguaje ASP 3.0) también dificulta la creación de mejoras, al no mostrar la información de la consulta completa cuando se tienen resultados que sobrepasan los 4MB permitidos por el servidor web IIS 6.0 (Internet Information Services). Además, el servidor mencionado no puede ser configurado por aspectos de pérdida de las garantías de soporte por parte de sus proveedores.

En la fase de Elaboración se establecen las actividades del equipo de trabajo y el plan de ejecución, que se refiere a la interacción entre los integrantes del equipo.

Actividades del equipo por rol

Desarrollador:

- Construir la aplicación web de los reportes de BI.
- Generar el reporte solicitado con las características proporcionadas por el *Gerente de negocio*.
- Realizar pruebas internas a cada reporte.

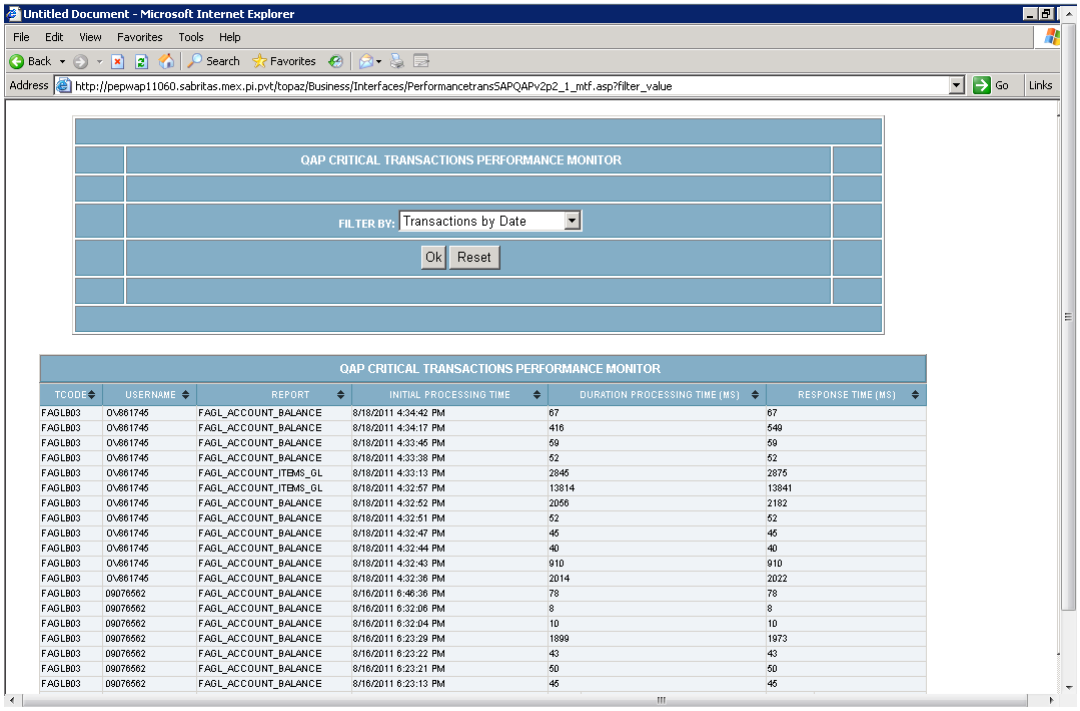


Figura 4.1: Ejemplo de un reporte original de la empresa

- Entregar el reporte finalizado al *Equipo de pruebas IT*.
- Asegurar la disposición de la aplicación en el ambiente de pre-producción, en cada nueva versión de la aplicación de reportes,
- Crear la documentación para la solicitud de reportes, para el control de los cambios en los reportes y el manual de despliegue de la aplicación.

Administrador de la base de datos:

- Crear las réplicas o instancias de las bases de datos necesarias para la generación de los reportes y su puesta en producción.
- Asegurar la integridad de la información al realizar las réplicas de las bases de datos para desarrollo, pruebas y producción.
- Crear las tablas con la información solicitada por el *Gerente de negocio* para la creación del reporte.
- Entregar al *Desarrollador* los datos de acceso a la bases de datos donde se encuentran las tablas del reporte a generar.

- Adquirir y brindar mantenimiento al equipo para el ambiente de pre-producción y producción.

Equipo de pruebas de TI:

- Proporcionar la infraestructura al *Desarrollador* para iniciar el desarrollo.
- Crear el ambiente de trabajo en pre-producción para el equipo donde se realiza el desarrollo.
- Recibir el reporte generado por el *Desarrollador* y realizar pruebas con las rṕelicas de las bases de datos proporcionadas por el administrador de la base de datos.
- Realizar las pruebas respectivas y entregar los resultados al *Gerente de negocio* para la validación del reporte.
- Agregar el reporte en BSM una vez validado.

Gerente de negocio:

- Entregar y analizar los requerimientos con el *Desarrollador*.
- Ejecutar las pruebas de validación y documentar los resultados.
- Revisar cambios o correcciones con el *Desarrollador*.
- Validar el reporte con datos de negocio.

El plan de ejecución para realizar un reporte se muestra en la Figura 4.2. Se presentan por medio de una flecha las actividades de interacción entre los roles durante el desarrollo de un reporte, de acuerdo a las listas de actividades que se acaban de mencionar.

4.1.2. Análisis de requerimientos

En la fase de Inicio se acuerda que el objetivo es generar una serie de reportes que cubran las necesidades demandadas por la Empresa y que sean publicados en la aplicación de Inteligencia de Negocios BSM mencionada en la Sección 3.1.

Requerimientos

Con el objetivo de establecer los requerimientos, se realiza una revisión

Figura 4.2: Plan de ejecución

con el gerente de negocio de lo solicitado por el cliente final. Se tiene la siguiente lista de requerimientos en orden de prioridad:

R1. Facilidad de acceso a los reportes: Permitir que la visualización de los reportes sea de forma inmediata, por medio de una URL, que permita ser publicado en una herramienta BSM de BI.

R2. Crear filtros para los datos de la consulta: Incluir tanto filtros dinámicos como estáticos que permitan buscar o seleccionar cualquier cadena de información requerida.

R3. Efectividad en los datos: Mostrar la información completa de acuerdo a los filtros seleccionados, garantizando que lo que se muestra es información fidedigna.

R4. Despliegue de información dinámica del reporte: Mostrar la información actualizándola de acuerdo a intervalos de tiempo establecidos.

R5. Permitir el ordenamiento de los datos de la consulta: Incluir en cada columna de la tabla de datos opciones de ordenamiento ascendente y descendente.

R6. Cambiar el diseño visual del reporte: Establecer un nuevo diseño que muestre mejoras visuales respecto a los reportes originales de la Em-

presa.

Otros requerimientos

Los siguientes requerimientos forman parte de las actividades posteriores al desarrollo y entrega de la aplicación web:

O1. Cambios en las necesidades de diseño: Poder incluir en un mismo reporte, además de filtros, indicadores, gráficas, concentrados y otros tipos de características.

O2. Atención a cambios en los requerimientos de negocio: Realizar modificaciones a los reportes generados de acuerdo a los cambios en el origen de los datos.

O3. Mantenimiento y soporte a la aplicación de los reportes: Mantener la disposición de los reportes.

Los requerimientos solicitados corresponden a solicitudes generales y no establecen especificaciones muy detalladas. Esto permite tener la libertad de elegir las tecnologías para desarrollar las funcionalidades y realizar propuestas que solucionen las reglas establecidas. Los aspectos técnicos también se dejan a criterio del desarrollador, siempre y cuando la aplicación web pueda instalarse en el servidor disponible por la parte usuaria (los requisitos se mencionan en el Apéndice A, en la Sección A.2).

4.1.3. Diseño

El diseño de la aplicación se inicia una vez que se ha revisado el modelo de negocio y se han establecido los requerimientos. La actividad de diseño tiene mayor relevancia en la fase de *Elaboración*.

En esta fase se establece:

- La arquitectura de la aplicación web.
- Los algoritmos de las funcionalidades principales.
- Los escenarios base y secundarios.

- El propotipo de un reporte en su forma más simple.

En las siguientes subsecciones se describen cada uno de los puntos mencionados.

Arquitectura de la aplicación web

Los componentes que conforman la arquitectura de la aplicación web (ver Figura 4.3) son el dispositivo de usuario (navegador web), el servidor Web (Apache tomcat) y el servidor para gestionar la base de datos (Oracle). También se indica la relación con el almacén de datos (componente de BI mencionado en la Sección 2.1.2) .

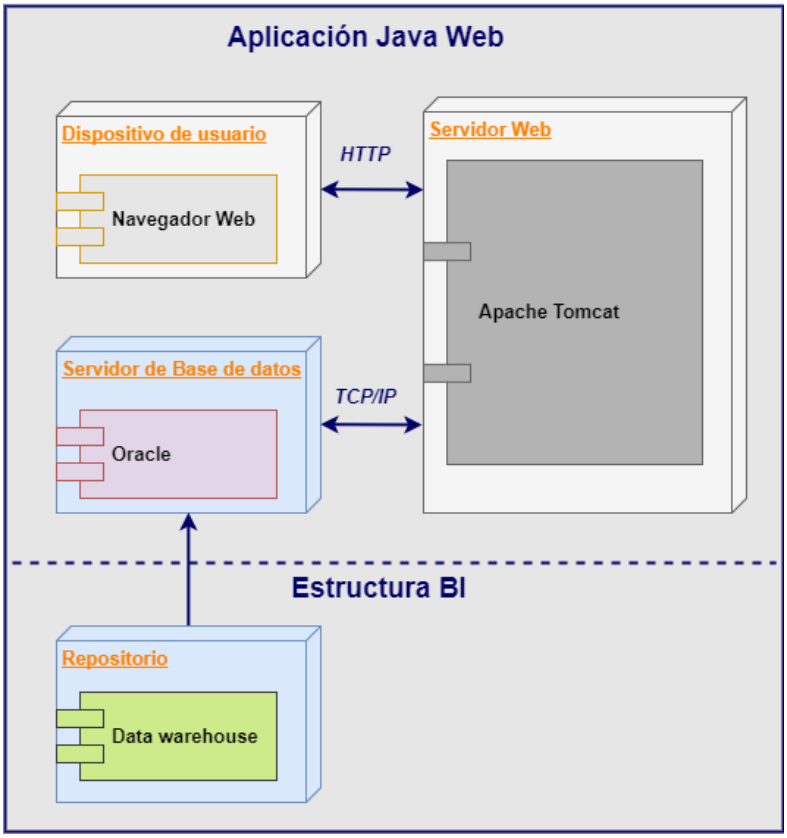


Figura 4.3: Diagrama de componentes de la aplicación web

Funcionalidades principales

Las funcionalidades principales para la generación de un reporte están desarrolladas del lado del servidor o del lado del navegador. En la Tabla 4.1 se indican las funcionalidades gestionadas en cada uno de ellos.

Tabla 4.1: Gestión de las funcionalidades

	Servidor	**Navegador**
Obtención de datos del objeto JSON	*	*
Construcción de la consulta	*	
Conteo de registros para la paginación	*	
Ejecución de las consultas	*	
Resultado de la consulta	*	*
Paginación	*	*
Avance automático del reporte		*

Cada reporte puede incluir alguna otra característica gestionada del lado del servidor o del navegador. A continuación, se mencionan brevemente los pasos necesarios para desarrollar cada una de las funcionalidades de la tabla 4.1

Obtención de datos del arreglo JSON:

- Recibir en el servlet los parámetros enviados desde el navegador, como el tipo de ordenamiento, columna de ordenamiento, filtros de búsqueda (checks o introducidos desde la cabecera), total de registros mostrados por página, etc..
- Crear un arreglo y un objeto tipo JSON.
- Asociar en otro objeto tipo JSON los parámetros con su respectivo valor. Por ejemplo, ordenamiento : ascendente.

Construcción, conteo de registros, ejecución y resultado de la consulta:

- Construir la consulta y agregar los parámetros recibidos del objeto JSON.
- Obtener el total de registros de la consulta con un SELECT.

- Calcular el total de páginas del reporte (total de páginas = total de registros/total de registros por página).
- Ejecutar la consulta realizando la evaluación de la expresión obtenida.

Paginación:

- Realizar el conteo de registros de la consulta.
- Obtener el dato correspondiente a la cantidad de renglones que se utilizan por página.
- El resultado de la paginación se imprime en el reporte.

Avance automático:

- Se recibe el parámetro del tiempo de espera y el momento en el que se actualiza la tabla de datos.
- Se consulta la base de datos para actualizaciones en los datos.
- Se actualiza el reporte.

Escenario base

El escenario base para la generación de un reporte considera las llamadas entre los componentes principales durante la *carga incial*, correspondiente al momento de consultar el reporte en el navegador por primera vez.

El Diagrama de interacción de la carga inicial entre el navegador y el servidor es la que se muestra en la Figura 4.4:

1. El usuario ingresa la URL del reporte en un navegador.
2. El navegador realiza una petición al servidor de acuerdo al protocolo HTTP.
3. El servidor realiza la ejecución del JSP, cuyo resultado es enviado de vuelta al navegador que realiza la petición.
4. El navegador recibe la página HTML generada, junto con los archivos asociados (por ejemplo, scripts de Javascript, hojas de estilo CSS, imágenes, etc.).

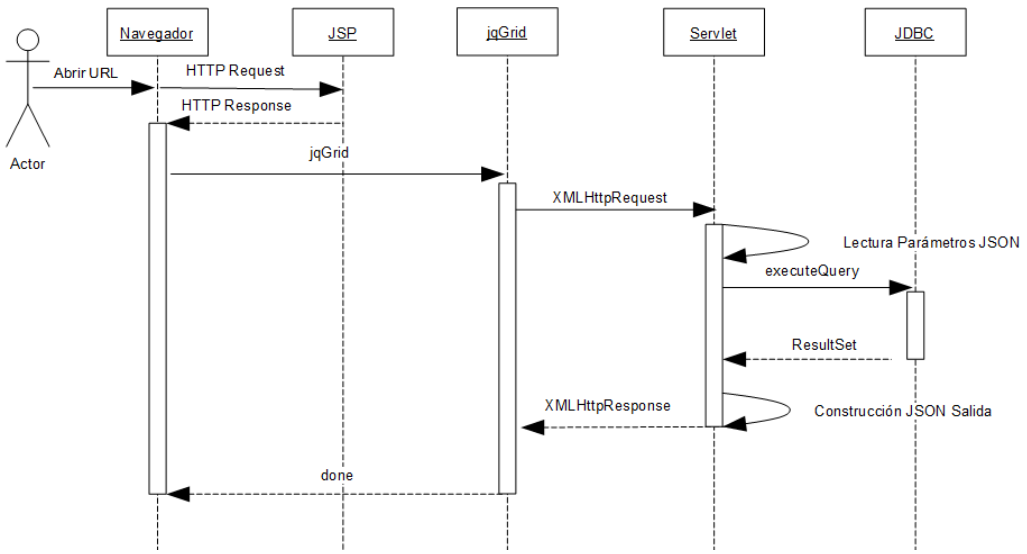


Figura 4.4: Diagrama de Interacción de la Carga inicial

5. El navegador interpreta la página y, al finalizar, se ejecuta la función Javascript de jQuery ready.
6. El código de jQuery de jqGrid realiza una llamada al servidor, por medio del objeto XMLHttpRequest, para obtener los datos de acuerdo a los parámetros especificados.
7. El servidor recibe una petición de acuerdo al protocolo HTTP y realiza la ejecución de un servlet. En el servlet se ejecutan tres pasos:

   7.1 Lectura de los parámetros de la consulta, interpretando el formato JSON, y construyendo las cadenas de consulta a la base de datos de acuerdo a los parámetros.

   7.2 Ejecución de la consulta a la base de datos y obtención de los resultados.

   7.3 Construcción del objeto JSON de salida de acuerdo a los resultados.

8. El código de jQuery, de jqGrid, recibe el objeto de salida en formato JSON y, de acuerdo a dichos resultados, se manipula el árbol DOM de

la página para mostrar la tabla con los datos obtenidos.

Escenarios de actualización

En esta sección se describen los escenarios correspondientes a las actualizaciones manual y automática de los elementos del reporte.

Actualización manual de los elementos del reporte:

La tabla de datos cuenta con controles que permiten cambiar los parámetros de búsqueda establecidos en la carga inicial. Cuando el usuario realiza cambios en esos parámetros, se realiza una nueva llamada al servidor y una actualización de los datos mostrados.

El Diagrama de Interacción de la actualización manual del reporte se muestra en la Figura 4.5:

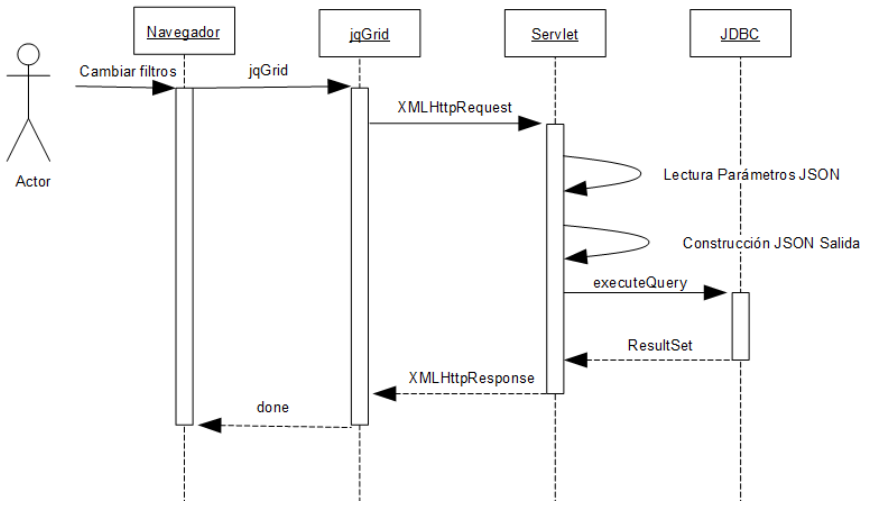


Figura 4.5: Diagrama de Interacción de la Actualización manual

1. El usuario realiza cambios utilizando los controles disponibles en la tabla de datos.
2. Las peticiones se capturan como eventos JavaScript que actualizan los parámetros y envían una petición al servidor de acuerdo a los parámetros especificados.

3. El servidor recibe una petición de acuerdo al protocolo HTTP y realiza la ejecución de un servlet. En el servlet se ejecutan lo siguiente:

   3.1 Lectura de parámetros de la consulta, interpretando el formato JSON y la construcción de las cadenas de consulta a la BD de acuerdo a dichos parámetros.

   3.2 Ejecución de la consulta a la base de datos y obtención de los resultados.

   3.3 Construcción del objeto JSON de salida de acuerdo a los resultados.

4. El código jQuery de jqGrid recibe el objeto de salida en formato JSON. De acuerdo a dichos resultados y los parámetros dados, se actualiza el DOM de la página para mostrar la tabla de datos con los datos obtenidos.

Actualización automática de los elementos del reporte:

Los reportes cuentan con un procedimiento de ejecución automática para la actualización de la información del reporte. Para dicho procedimiento, se usa la función *setTimeout* de JavaScript, que permite ejecutar una función cada cierta unidad de tiempo. De tal forma que se realiza periódicamente una llamada al servidor con los últimos parámetros establecidos, tal que si hay cambios en la base de datos, éstos se ven reflejados inmediatamente. La ejecución de la actualización se activa con el botón del avance automático, y los cambios se realizan con una nueva llamada al servidor y una actualización de los datos mostrados.

El Diagrama de Interacción de la actualización automática del reporte se muestra en la Figura 4.6:

Escenario para excepciones

Los escenarios para manejo de excepciones del servlet utilizan las clases propias para el control de excepciones que se mencionan en el flujo de la

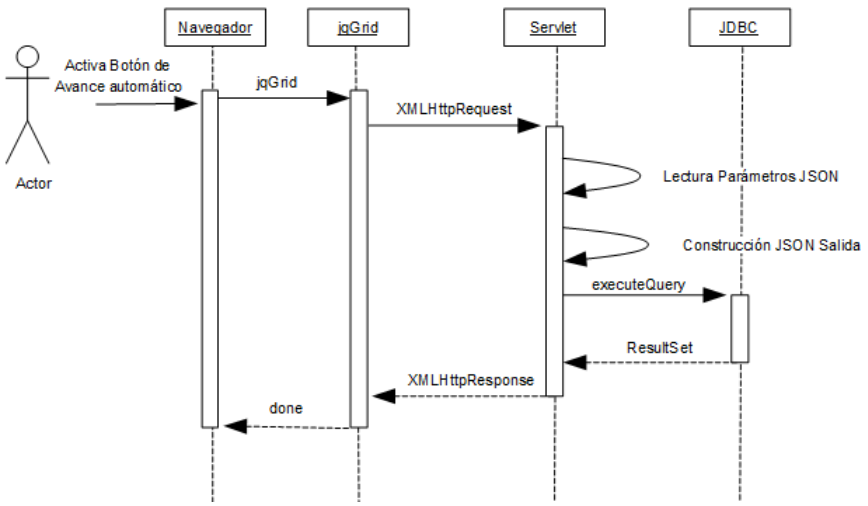


Figura 4.6: Diagrama de Interacción de la Actualización automática

Sección 4.2.3. Para la disponibilidad del reporte se redactan manuales del alta y baja del servicio que se mencionan en el flujo de la sección 4.4.

Prototipo de un reporte

En la Figura 4.7 se tiene un prototipo de un reporte en su forma más simple, con información local o estática.

El prototipo del reporte está formado por cuatro secciones básicas [15]:

- La leyenda - Es el título del reporte.
- La cabecera - Incluye los campos o nombres de las columnas del reporte.
- El cuerpo - Se compone de todos los registros de datos de cada uno de los elementos de la cabecera.
- La paginación - En esta parte se establecen las características de la paginación; cantidad de elementos por página, botón Anterior, botón Siguiente, botón Primero y botón Último.

Estas son las secciones estándar pero pueden agregarse o eliminarse de acuerdo a las necesidades del reporte.

Figura 4.7: Prototipo de un reporte

4.2. Implementación

Las actividades de este flujo tienen relevancia en la fase de Construcción y menor grado de aparición en la fase de Transición generalmente para corrección de errores o cambios de bajo impacto en los requerimientos o el diseño original.

La generación del reporte se describe por medio del desarrollo de un reporte que incluye las características o funcionalidades que solucionan cada uno de los requerimientos listados de la Sección 4.1.2. Éstos se asocian a distintos tipos de reportes, que se revisan en el siguiente capítulo: dirección de acceso del reporte (**R1**); filtros de búsqueda estáticos para mostrar segmentos de información y filtros de búsqueda dinámicos para buscar informacin en las columnas de la tabla de datos (**R2**); paginación (**R3**); actualización

automática de la tabla de datos por tiempo establecido, en la paginación o en el avance automático (**R4**); ordenamiento de columnas (**R5**); un nuevo diseño visual (**R6**) y otros elementos visuales como campos totales, calendario, gráficas, etc. (**O1**).

La solución al requerimiento (**O2**) se gestiona por medio de documentación diseñada para el control de cambios mencionada en la Sección 4.4. El requerimiento (**O3**) se soluciona por medio del acceso al servidor donde se encuentra instalado el sistema, y la creación de scripts de bash mencionados en la Sección 4.4.

4.2.1. Estructura de un reporte

Un reporte se conforma de un JSP, un servlet y una liga de acceso o URL. Es posible utilizar un mismo servlet para mostrar distintos reportes, pero cada reporte está asociado a un único JSP.

Los elementos de un reporte son los siguientes:

- URL. Es la dirección o liga de acceso al reporte, que se ingresa por medio de un navegador.
- JSP (JavaServer Page). Es un archivo con extensión `jsp` que incluye código JavaScript para configurar y desplegar la tabla de datos. También incluye HTML y CSS para aspectos visuales del reporte.
- Servlet. Es un archivo `Java` que corresponde al código de las funcionalidades gestionadas por el servidor mencionadas en la Sección 4.1.3.

En las secciones posteriores, se describen el JSP y Servlet de un reporte básico, cuyo resultado se muestra en la Figura 4.8.

4.2.2. JSP

Un archivo JSP es un código ejecutable del lado del cliente o navegador. Incluye la declaración y configuración de la tabla de datos, el código Javascript (jQuery), las llamadas de jqGrid (Ajax) y el diseño visual del reporte.

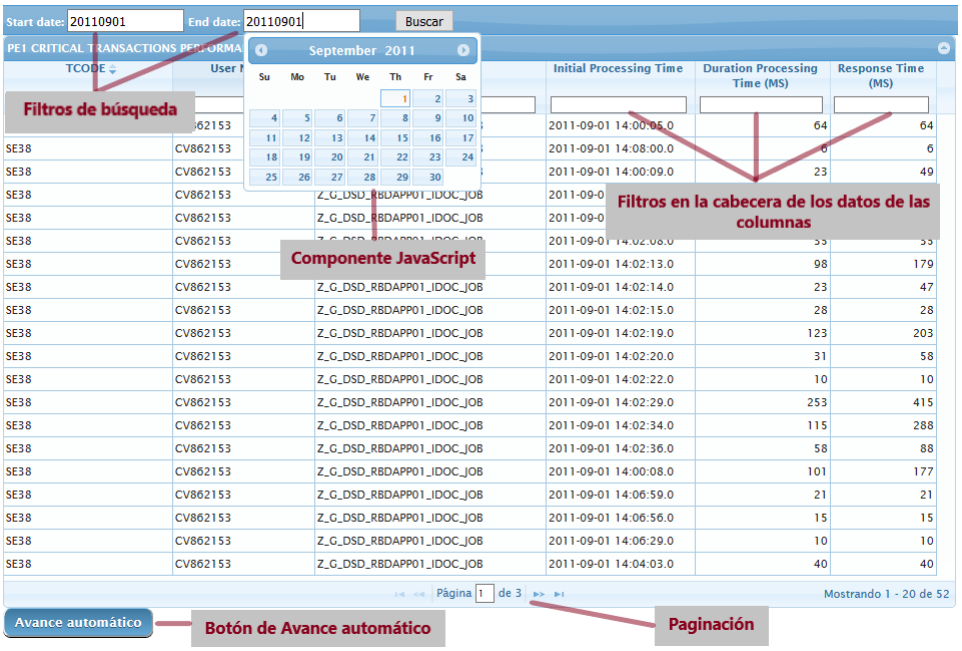


Figura 4.8: Reporte Básico

A continuación, se describen las partes más significativas que conforman el JSP.

El objeto document

El objeto ***document*** incluye la llamada de jQuery (JavaScript) a jqGrid para declarar la tabla de datos. Contiene las propiedades y los métodos para acceder a todos los objetos desde JavaScript, es decir, la página que se está visualizando, texto, imágenes, enlaces, etc. Este objeto permite agregar dinámicamente contenido a la página o realizar cambios. Una página no se puede manipular hasta que el documento esté listo. jQuery detecta dicho estado de preparación. El código dentro de `$(document).ready ()` (ver Código 4.1) sólo se ejecuta cuando la página Document Object Model (DOM) está lista para que se ejecute el código JavaScript.

```
\$(document).ready(function (){
      jQuery("#list").jqGrid({
      ...
      });
});
```

Código 4.1: Objeto **document**

Configuraciones principales

Las configuraciones asociadas a los requerimientos para obtener el reporte de la Figura 4.8 se muestran en el Código 4.2 y son las siguientes:

- El valor de la variable `URL = bsmgrid/GeneradorReporteServlet` hace referencia a la ubicación y al nombre del servlet. Significa que el servlet con nombre **GeneradorReporteServlet** se encuentra en la carpeta **bsmgrid**. Esta ubicación corresponde a la estructura de archivos típica de una aplicación web, y se indica en la Sección A.2 del Apéndice A.
- El valor de la variable `mtype = GET` establece el método utilizado para enviar la información al servlet. Se elige el tipo de solicitud **GET** debido a que se solicitan pocos parámetros y exclusivamente para su consulta. La información que se envía es la URL con los parámetros correspondientes (filtros estáticos y dinámicos, características del ordenamiento, entre otros necesarios).
- El valor de la variable `datatype = JSON` establece el tipo de formato en que se recibe el arreglo JSON desde el servlet. Se elige este formato debido a que existen clases en Java para manejarlo. La construcción del arreglo se describe en la Sección 4.2.3.
- El valor de la variable `gridview = true` establece que todos los renglones de la tabla se muestren al mismo tiempo. Lo anterior se elige para que los datos se muestren de manera inmediata (agregar la referencia de la documentación de jqGrid).
- El valor de la variable `pager = #pager` indica la referencia al nombre de la barra inferior de paginació, que se conforma de un componente HTML (que se describe más adelante en esta sección) y la información se construye en el servlet que se describe en la siguiente Sección 4.2.3.

- El valor de la variable `sortname = DIA` establece el nombre del campo respecto al que se realiza el ordenamiento de la tabla. En este caso, la columna **DIA** del modelo de datos (siguiente sección de este apartado).
- El valor `sortorder = ASC` establece que el ordenamiento se realiza en forma ascendente sobre la columna **DIA** del punto anterior.
- El valor de la variable `groupOp = OR` establece el operador lógico de la búsqueda entre los filtros para construir la consulta SELECT en el servidor.

Los valores listados forman parte de los parámetros que son enviados al servlet en el hashmap `Filters` (que se explica en la Sección 2.3.10).

```
jQuery("#list").jqGrid({
 url:'/bsmgrid/GeneradorReporteServlet',}
 mtype: "GET",
 datatype: "JSON",
 ...
 gridview: true,
 pager: "#pager",
 sortname: 'DIA',
 sortorder: "ASC",
 groupOp:"OR",
 ...
});
```

Código 4.2: **Configuraciones principales**

Configuración del modelo de la tabla de datos

En esta parte se describe la configuración del modelo de datos (ver Código 4.3) mostrando los parámetros básicos necesarios para el ejemplo de un reporte.

- El arreglo `colNames` corresponde a las columnas, separadas por comas, que componen la tabla de datos.
- El arreglo `colModel` corresponde a las características de cada una de las columnas declaradas. Cada columna se representa por un renglón que incluye:

- La asignación de la variable `name:nombre` que indica el nombre de la columna.
- La asignación de la variable `index:indice` que indica el índice de la columna. Se utiliza para enviar el nombre del campo al servidor y simplificar la construcción del filtro ORDER BY en la consulta SELECT.
- La asignación de la variable `width:ancho` indica el ancho de la columna. En este caso el valor de `ancho` corresponde a un entero.
- La asignación de la variable `hidden:true` indica que la columna se oculta. Por defecto, todas las columnas se muestran en la tabla de datos.

```
jQuery("#list").jqGrid({
   ...
   colNames:['TCODE','TASK', 'User Name','REPORT','Initial
      Processing Time','Duration Processing Time (MS)','
      Response Time (MS)','CPU Time (MS)','MEMORY Time (MS)
      '],
   colModel :[
   {name:'TCODE', index:'TCODE', width:240},
   {name:'TASK', index:'TASK', width:130, hidden: true},
   {name:'ACCOUNT', index:'ACCOUNT', width:200},
   {name:'REPORT', index:'REPORT', width:330},
   {name:'Initial_Processing_Time', index:'
      Initial_Processing_Time', width:210},
   {name:'TIMEINWP', index:'TIMEINWP', width:190, align:'
      right'},
   {name:'RESPTI', index:'RESPTI', width:150, align:'right'},
   {name:'CPUTI', index:'CPUTI', width:150, align:'right',
      hidden: true},
   {name:'MAXMEM', index:'MAXMEM', width:150, align:'right',
      hidden: true}],
  ...
});
```

Código 4.3: **Configuraciones del modelo**

Otras configuraciones

En la siguiente lista se tienen otras configuraciones que son indispensables para la completar la configuración del reporte (ver Código 4.4).

- El valor de la variable `loadonce = false` indica que se deshabilita dicha opción para que no se carguen los datos y que el ordenamiento y filtrado se realicen del lado del servidor. Es decir, que se carguen de acuerdo a las peticiones o solicitudes realizadas por el usuario.
- El valor de la variable `viewrecords = true` muestra el número de registros inicial y final en la tabla (distinto al número total de registros en la consulta). Esta información se muestra en la barra de la paginación (abajo y a la derecha por defecto) en este formato: "Ver X a Y desde Z". Si este valor es verdadero, hay otros parámetros que se pueden ajustar.
- El valor de la variable `rownumbers = false` indica que no es necesario agregar una columna adicional, al inicio de las columnas, con una lista secuencial de los registros.
- El valor de la variable `rowNums = 20` indica la cantidad de registros mostrados por página. Se establece a 20 registros que equivalen a una cantidad equivalente al tamaño de una página en el navegador.
- El valor de la variable `height` indica la altura de la tabla de datos y `autowidth` indica el ancho. Se establece una altura fija acorde al número de renglones en una página.
- La variable `gridComplete` se utiliza para indicar que una vez que la tabla de datos esté completa se ejecute la función `reloadIcons()` (o las funciones incluidas). El evento se dispara independientemente del parámetro **datatype** y después del ordenamiento, la paginación, etc.

```
\begin{lstlisting}[style=basic,language=My,numbers=none]
 jQuery("#list").jqGrid({
   ...
   loadonce:false,
   viewrecords: true,
   rownumbers: false,
   rowNum:20,
   height: 360,
   autowidth: true,
   ...
   gridComplete: function() {
                 reloadIcons();
```

```
        }
});
```

Código 4.4: **Otras onfiguraciones**

Funcionalidades JavaScript, jQuery y jqGrid

En esta sección se muestra el código de las funcionalidades del reporte que son parte del JSP y desarrolladas por medio de JavaScript, jQuery y jqGrid.

Actualización manual

La actualización manual del reporte se lleva a cabo cuando el usuario selecciona los filtros estáticos y/o los ingresa en la barra de búsqueda (filtros) dinámicos.

Filtros de búsqueda estáticos

La función `gridReload()` es necesaria para enviar al servidor los filtros de búsqueda estáticos para mostrar segmentos de información (ver Código 4.5). Ésta se ejecuta cuando se carga la tabla de datos al consultar el reporte en el navegador.

Por medio del valor **setGridParam** se indica que se agregan parámetros al arreglo que se envía al servlet **GeneradorReporteServlet.java** que corresponden a los campos dateArrival y dateArrival2.

```
function gridReload(){
  var dateArrival = jQuery("#dateArrival").val();
  var dateArrival2 = jQuery("#dateArrival2").val();
  jQuery("#list").jqGrid('setGridParam',{url:"/bsmgrid/
     GeneradorReporte?dateArrival="+dateArrival+"&
     dateArrival2="+dateArrival2,page:1}).trigger("reloadGrid
     ");
}
```

Código 4.5: **Envío de filtros estáticos al servlet**

Mediante código HTML, que es la parte correspondiente al despliegue visual, se especifica el contenido estático y mediante fragmentos de JavaScript

se genera el contenido dinámico del reporte, como puede verse en el Código 4.6.

```
<body>
<div id=inter>
Start date:
<input name="txtfechainicio" type="text" id="dateArrival"
onkeydown="doSearch(arguments[0]||event)" size="15" maxlength
   ="9" />
End date:
<input name="txtfechafin" type="text" id="dateArrival2"
onkeydown="doSearch(arguments[0]||event)" size="15" maxlength
   ="9" />
<button onclick="gridReload()" id="submitButton" style="
   margin-left:30px;">Buscar</button>
</div>
<table id="list" ><tr><td/></tr></table>
<div id="pager" ></div>
</body>
```

Código 4.6: **Despliegue visual**

Filtros de búsqueda dinámicos

Los filtros de búsqueda que el usuario introduce en los encabezados de las columnas (ver las variables en el Código 4.7) corresponden a los filtros dinámicos. Éstos son enviados al servidor con la variable filters en una llamada en segundo plano a jqGrid por medio de un arreglo JSON.

```
_search true
dateArrival 20110901
dateArrival2  20110901
filters {"groupOp":"AND","rules":[{"field":"TIMEINWP","op":"
   bw","data":"15"}]}
page  1
rows  20
sidx  TCODE
sord  desc
```

Código 4.7: **Envío de filtros**

Actualización automática

La función recursiva **reloadForTime(segundos de espera,estado)** realiza el avance automático, utilizando la función *setTimeout* por un tiempo

establecido. Y la actualización de la tabla de datos con *jQuery("#list"). trigger("reloadGrid")* (ver Código 4.8).

```
function reloadForTime(segsEspera,changeAtCall){
 if(actualizaAutom){
   if(changeAtCall){
         if(!jQuery("#next").hasClass("ui-state-disabled")){
             jQuery("#next").click();
         } else if(!jQuery("#first").hasClass("ui-state-
            disabled")){
             jQuery("#first").click();
             jQuery("#list").trigger("reloadGrid");
         }else {
             jQuery("#list").trigger("reloadGrid");
         }
   }
     setTimeout("reloadForTime("+segsEspera+",true)",
        segsEspera*1000);
 }
}
```

Código 4.8: **Avance automático**

Esta función está asociada a un botón que activa la actualización de la tabla de datos y el avance automático (ver Código 4.9).

```
var actualizaAutom = false;
jQuery("#actualizaAutom").click(function(e){
  actualizaAutom = (!actualizaAutom);
  if(actualizaAutom){
    reloadForTime(30,false);
    jQuery(this).val("Detener avance automÃ¡tico");
  } else {
    jQuery(this).val("Avance automÃ¡tico");
  }
});
```

Código 4.9: **Avance automático**

Botón de Avance autómatico

En el Código 4.10, se muestra la asociación del botón de avance con la llamada a jQuery del id actualizaAuto.

```
<input class="button" type="button" id="actualizaAuto" value=
   "Avance autom&aacute;tico" />
```

Código 4.10: **Avance automático**

Hasta este punto se concluye con la descripción de las funcionalidades del archivo JSP asociado al desarrollo de un reporte.

4.2.3. Servlet

Los servlets permiten generar información en forma dinámica para el reporte, a partir de los parámetros de la petición que envía el navegador web desde el JSP. Cada servlet se asocia a un reporte por medio de un archivo JSP, y es posible asociar un mismo servlet para mostrar distintos reportes.

En general, el servlet asociado al reporte recibe la información del JSP, la procesa por medio de las clases y métodos correspondientes y, posteriormente, despliega los resultados en el JSP. A continuación, se describe el código de las funcionalidades más importantes de un servlet asociado a un reporte. La descripción incluye los métodos básicos de un servlet: las clases de Java importadas, la recepción de los parámetros desde el navegador, la construcción de la consulta a partir de los paraámetros recibidos, la construcción del arreglo JSON para enviarlo al navegador (como respuesta a su solicitud) y el código de la paginación. Todo lo que se menciona es con el fin de que este trabajo funcione como una guía para recrear los reportes.

Métodos básicos de un Servlet

Los métodos básicos para gestionar las llamadas al servlet por parte del navegador y procesar las solicitudes HTTP son (ver Figura 4.11):

- `doGet()` se utiliza para recuperar datos del servidor web. Se elige doGet() porque sólo se realizan consultas al servidor. El contenedor del servlet crea una sola instancia de cada servlet y la solicitud de cada usuario se maneja con un hilo separado. Cada hilo llama al método doGet().
- `processRequest()` se utiliza como servicio para gestionar las solicitudes HTTP del tipo GET.

```
// Gestiona una llamada tipo GET.
// @param request servlet request
// @param response servlet response
```

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response)

// Procesa las solicitudes Http: GET
// @param request servlet request
// @param response servlet response
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
```

Código 4.11: **Métodos básicos utilizados en el servlet**

Clases importadas

En esta parte se listan las clases de Java que se importan para utilizar sus métodos.

- Clases básicas. Se utilizan para la declaración: java.io.PrintWriter y java.util.HashMap.
- Clases para la gestión de la base de datos. Son necesarias para establecer la conexión a la base de datos y realizar consultas a las tablas:
  - java.sql.Connection
  - java.sql.ResultSet
  - java.sql.Statement
  - javax.sql.DataSource.
- Clases para la gestión de un Servlet. Son necesarias para iniciar el contexto del servlet y poder invocar solicitudes HTTP:
  - javax.naming.Context
  - javax.naming.InitialContext
  - javax.servlet.http.HttpServlet
  - javax.servlet.http.HttpServletRequest
  - javax.servlet.http.HttpServletResponse.
- Clases para el manejo de objetos JSON. Son necesarias para para manejar los archivos JSON:
  - org.json.simple.JSONArray
  - org.json.simple.JSONObject
  - org.json.simple.JSONValue.
- Clases para el control de excepciones.

- java.io.IOException
- java.sql.SQLException
- javax.naming.NamingException
- javax.servlet.ServletException

Tipo de salida

La primera línea del Código 4.12 invoca al método setContentType sobre el objeto response. En otras palabras, se está indicando que la respuesta de este Servlet es en formato HTML. Posteriormente se indica que el objeto llamado out se forma del método getWriter del objeto response. En otros términos, todo valor asignado a **out** se envía a la impresión del Servlet (llevada acabo por getWriter).

```
    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
throws ServletException, IOException {
  response.setContentType("text/html;charset=UTF-8");
  PrintWriter out = response.getWriter();
```

Código 4.12: **Métodos básicos utilizados en el Servlet**

Mapeo de parámetros

En esta parte se declara un `HashMap` con nombre `mappings` para el manejo de los parámetros recibidos en la llamada de jqGrid desde el cliente. Se utiliza este objeto por el tipo de llamada asíncrona (Ajax) y porque no es necesario que los elementos cuenten con un orden, pues basta identificarlos en una relación `parámetro:valor`.

```
//Declaración
private HashMap<String, String> mappings = new HashMap<String
    , String>();
...
//Obtención de campos de filtrado para las fechas
mappings.put("Initial_Processing_Time", "startdate");
...
//Asignación del parámetro
String field = jsobj.get("field").toString();

if (mappings.containsKey(jsobj.get("field").toString())) {
    field = mappings.get(jsobj.get("field").toString());
```

```
}
```

Código 4.13: **Mapeo de parámetros**

Declaración del objeto y del arreglo JSON

En esta parte se declara un objeto JSON, donde se almacenan los filtros y reglas enviados por el JSP.

```
JSONObject obj = (JSONObject) JSONValue.parse(request.
   getParameter("filters"));
JSONArray array = (JSONArray) obj.get("rules");
...
//Declaración del campo Filtros enviado por jqGrid (Ajax)
boolean firstFilter = true;
//Construcción de la consulta
for (Object o : array) {
       JSONObject jsobj = (JSONObject) o;
       ...
       if (firstFilter) {
          where1String += "" + field + " LIKE '" + jsobj.get(
             "data") + "%'";
          firstFilter = false;
          } else {
          where1String += " AND " + field + " LIKE '" + jsobj
             .get("data") + "%'";
       }
 }
```

Código 4.14: **Declaración del objeto y del arreglo JSON**

Generación de filtros

En esta parte se comienza a crear la cadena que se va a incluir en la consulta que se despliega en la tabla de datos del reporte.

```
    else {
          if (firstFilter) {
             where1String += " to_char(" + jsobj.get("field")
                 + ",'yyyymmdd') LIKE '" +
                  + jsobj.get("data").toString() + "%' ";
             } else {
                  where1String += "AND to_char(" + jsobj.get(
                     "field") +
                       + ",'yyyymmdd') LIKE '" +    jsobj.get
                          ("data").toString() + "%' ";
               }
```

```
    }
```

Código 4.15: **Generación de filtros**

Contexto del servlet

Cuando se trabaja con JSPs y servlets se utilizan rutas relativas para establecer la referencia a los archivos. El *servlet context* (contexto del servlet) maneja la información a nivel de toda la aplicación web. La clase ServletContext contiene métodos que sirven para que un servlet se comunique con su contenedor. Todos los servlets de una aplicación tienen el mismo ServletContext. En el código 4.16 se muestra la declaración del contexto.

```
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup("java:comp/env");
DataSource ds = (DataSource) envCtx.lookup("jdbc/bsmgrid");
```

Código 4.16: **Contexto del servlet**

Conexión y consulta a la base de datos

En esta parte, se realiza la conexión y la consulta del total de registros (count) que es un dato necesario para realizar los cálculos de la paginación.

```
//Conexión a la base de datos
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup("java:comp/env");
DataSource ds = (DataSource) envCtx.lookup("jdbc/bsmgrid");
connection = ds.getConnection();
Statement select = connection.createStatement();
...
//Consulta
ResultSet resultCount =
select.executeQuery(" SELECT count(*) from BSMGRID.
   ZMXRBAST0002
" + where1String + " " + where2String);
resultCount.next();
count = resultCount.getLong(1);
resultCount.close();
```

Código 4.17: **Conexión y consulta a la base de datos**

Impresión de la paginación

Una vez que la consulta se tiene por completo, ésta se ejecuta y se comienza a imprimir en el reporte la paginación.

```
...
int ncols = result.getMetaData().getColumnCount();
  out.println("{");
  out.println("\"total\":" + total_pages + ",");
  out.println("\"page\":\"" + page + "\",");
  out.println("\"records\":\"" + count + "\",");
  out.println("\"rows\":[");
boolean first = true;
int index = 1;

while (result.next()) {
  if (!first) {
    out.print(",{");
  } else {
    out.print("{");
    first = false;
  }
  out.print("\"id\": \"" + index + "\",");
  out.print("\"cell\": [");

  boolean firstrow = true;
  for (int i = 1; i <= ncols; i++) {
    if (!firstrow) {
      out.print(",");
    } else {
      firstrow = false;
    }
  out.print("\"" + result.getString(i) + "\"");
  }
  out.print("]}");
  index++;
}
out.println("]");
out.println("}");
...
```

Código 4.18: **Impresión de la paginación**

De esta forma, se mencionan las partes fundamentales que integran el código del servlet asociado a un reporte.

Hasta aquí, se tiene la descripción que involucra la generación de reportes con una tabla de datos para la Inteligencia de Negocios.

4.3. Pruebas

Las pruebas forman parte de las fases Construcción y Transición. Para la aceptación de los reportes, se realizan a diversos niveles de prueba respecto a su objetivo: durante la implementación, por medio de validaciones realizadas por el área de pruebas TI y, al final, por las validaciones realizadas por el gerente de negocio.

Pruebas durante la implementación

Las pruebas de verificación se realizan con datos predefinidos y conociendo los resultados esperados. Para verificar los resultados de las búsquedas, se ejecutan las consultas directamente a la BD por medio de la opción correspondiente en Netbeans, y se comparan los resultados obtenidos con los esperados.

1. Se construye la consulta de un conjunto de registros.

```
SELECT rowkey, procedimiento, fecha_ini, fecha_fin,
    estatus, desc_error, tipo_rep
FROM bsmgrid.BSM_REPLOG  WHERE (tipo_rep LIKE '%azul%')
ORDER  BY rowkey desc
```

2. Se ejecuta la consulta en Netbeans obteniendo los resultados de la Figura 4.9.

#	ROWKEY	PROCEDIMIENTO	FECHA_INI	FECHA_FIN	ESTATUS	DESC_ERROR	TIPO_REP
1	100	Recursos humanos	2017/29/04	2018/05/12	OK	Error conocido	azul
2	94	Gestión de activos	2018/05/11	2018/05/12	OK	Error desconocido	azul
3	93	Atención al cliente	2018/17/02	2019/24/01	null	Error conocido	azul
4	86	Nóminas	2017/30/08	2017/23/06	null	Error desconocido	azul
5	64	Publicidad	2018/10/06	2018/13/05	OK	Error desconocido	azul
6	61	Finanzas	2018/09/09	2017/29/12	null	Error desconocido	azul
7	59	Publicidad	2018/13/03	2018/28/03	null	Error conocido	azul
8	55	Publicidad	2017/03/10	2018/02/03	OK	Error desconocido	azul
9	39	Gestión de activos	2017/21/04	2017/28/07	Wrong	Error conocido	azul
10	26	Ventas y Mercadotécnia	2018/17/05	2018/17/10	null	Error conocido	azul
11	23	Ventas y Mercadotécnia	2017/24/07	2017/19/05	OK	Error conocido	azul
12	21	Gestión de la calidad	2017/21/05	2017/06/09	OK	Error conocido	azul
13	20	Nóminas	2019/06/02	2017/09/03	OK	Error desconocido	azul
14	4	Gestión de la calidad	2018/17/10	2018/04/06	null	Error desconocido	azul
15	2	Gestión de la calidad	2018/19/12	2018/21/12	null	Error conocido	azul

Figura 4.9: Resultado esperado

3. Se realiza la misma consulta en el reporte escribiendo en el campo de búsqueda el valor de **azul** en la columna `tipo_rep` (ver Figura 4.10).

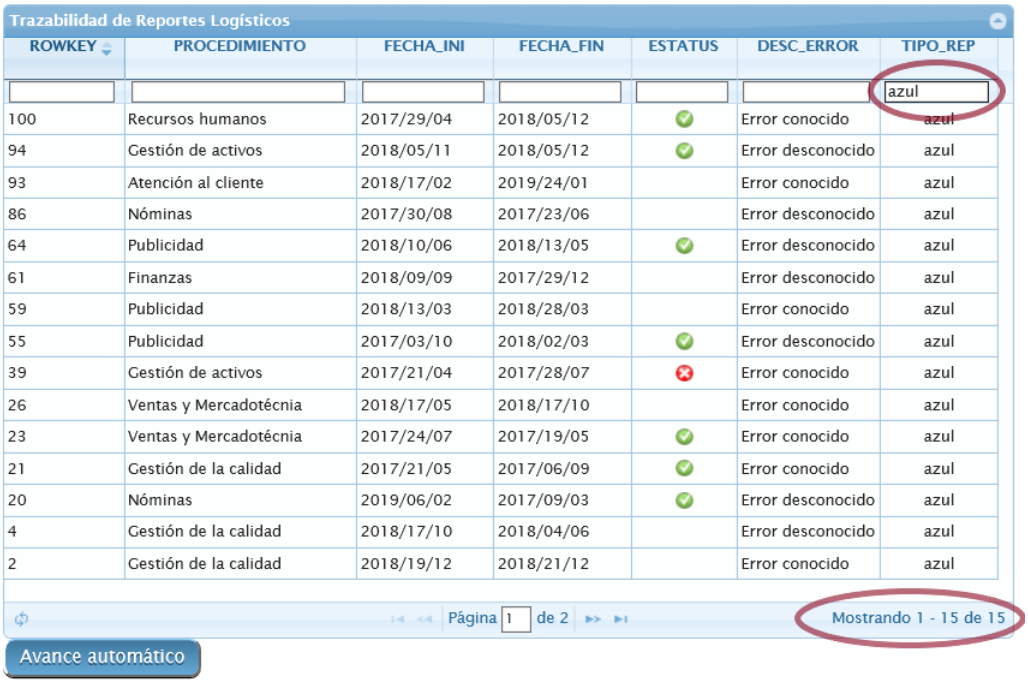


Figura 4.10: Resultado en el reporte al filtrar

4. Se comparan los resultados obtenidos en el paso 2 y 3.
5. Se observa que se obtienen los mismos resultados por lo que la prueba ha finalizado

4.4. Despliegue

En la fase de Transición, el despliegue se realiza por medio de la instalación de un archivo WAR que se genera al compilar el proyecto.

- Ingresar al servidor donde se encuentra alojada la aplicación de reportes BSMGRID, con un usuario con privilegios de escritura, borrado y ejecución.

- Situarse en la carpeta:
  D:\apache-tomcat-6.0.33-windows-x86\apache-tomcat-6.0.33\webapps
- Iniciar el servicio:
  D:\apache-tomcat-6.0.33-windows-x86\apache-tomcat-6.0.33\bin - startup.bat
- Pegar en la misma ubicación el nuevo archivo bsmgrid.war.
- Verificar que aparezca la carpeta "bsmgrid".
  En caso de que no se genere la carpeta, detener e iniciar el servicio, reiteradamente, hasta que la carpeta aparezca.
  Detener el servicio:
  D:\apache-tomcat-6.0.33-windows-x86\apache-tomcat-6.0.33\bin - shutdown.bat
- Verificar que la aplicación ya se encuentre disponible.
  Esto puede realizarse consultando alguno de los reportes, por ejemplo:
  http:\\localhost:8080\bsmgrid\index.jsp

4.5. Resumen

En este capítulo se ha descrito el desarrollo de la generación de reportes con tablas de datos para BI por medio de una metodología basada en AUP. Se explican las actividades realizadas en cada flujo de trabajo (Modelado, Configuración y Ambiente, Implementación, Pruebas y Despliegue) a lo largo de las fases (Inicio, Elaboración, Construcción y Transición). La implementación se lleva a cabo por medio de una aplicación web utilizando el lenguaje de programación Java y las tecnologías de JavaScript, JavaServer Pages, JDBC y jqGrid.

Capítulo 5
Atención a los requerimientos

En este capítulo se describen las respuestas a los requerimientos de la sección 4.1.2, indicando la característica que lo soluciona y mostrando ejemplos de reportes con dichas características.

5.1. Características de los reportes

A continuación se listan las características y se describen las soluciones a cada uno de los requerimientos de la Sección 4.1.2:

C1 Dirección de acceso del reporte. La aplicación web está compuesta de archivos JSP, donde cada uno de ellos corresponde a un reporte. Cada reporte se encuentra en el servidor web en una dirección de este tipo. De esta forma se soluciona el requerimiento **R1**.

C2 Filtros de búsqueda estáticos para mostrar segmentos de información y filtros de búsqueda dinámicos para buscar información en las columnas de la tabla de datos. Estas búsquedas con Filtros estáticos y dinámicos resuelve el requerimiento **R2**.

C3 Paginación. La Paginación resuelve el requerimiento **R3**, con la presentación de todos los datos de la consulta, permitiendo que sean mostrados dependiendo de la cantidad deseada de registros por página.

C4 Actualización automática de la tabla de datos por tiempo establecido, por medio de la Paginación o por Avance automático. Estas opciones de actualización automática resuelven el requerimiento **R4**, y se ejecuta del lado del cliente con JavaScript.

C5 El envío de las opciones de ordenamiento resuelven el requerimiento **R5**.

C6 Nuevo diseño visual. El nuevo diseño generado con jqGrid resuelve el requerimiento **R6**.

Para cumplir los requerimientos secundarios de la Sección 4.1.2, se realizan las siguientes acciones:

O1 Inclusión de elementos visuales. Se incluyen en los reportes: cantidades

totales, calendario, gráficas, indicadores y el avance automático.

O2 Documentación para el control de versiones.

O3 Creación de scripts para levantamiento del servicio de los reportes.

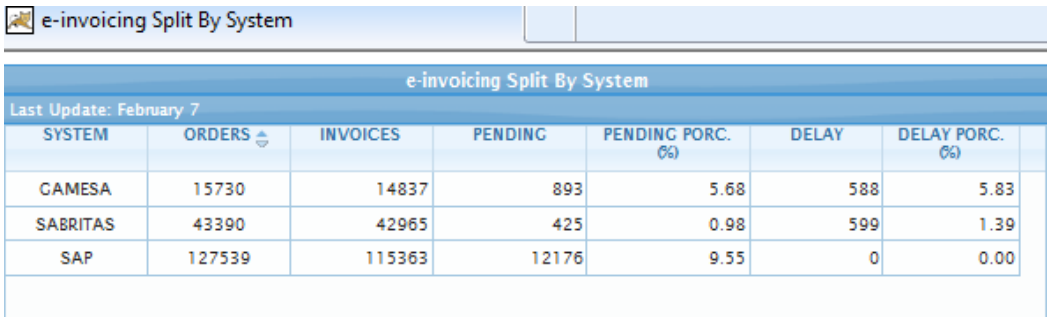
5.2. Reportes generados

Los reportes desarrollados se incluyen en la aplicación BSM por medio de un *portlet* (componente que muestra datos en varios formatos y permiten personalizar el contenido y la apariencia [42]). La seguridad de acceso queda a cargo del administrador de dicha herramienta. La liga de los reportes sólo puede ser consultada en la red interna (intranet).

A continuación, se muestran algunos de los tipos de reportes generados y se menciona la característica relacionada al reporte de la lista de la Sección 5.1.

Reporte simple (Figura 5.1)

Este reporte incluye las características **C1** y **C4**. Es utilizado por el Equipo de Facturación Electrónica y Cartera para revisar el estado de las facturas de distintos sistemas internos.


e-invoicing Split By System

e-invoicing Split By System

Last Update: February 7

SYSTEM	ORDERS	INVOICES	PENDING	PENDING PORC. (%)	DELAY	DELAY PORC. (%)
GAMESA	15730	14837	893	5.68	588	5.83
SABRITAS	43390	42965	425	0.98	599	1.39
SAP	127539	115363	12176	9.55	0	0.00

Figura 5.1: Reporte simple

Reporte con filtros (Figura 5.2)

Este reporte cumple con las características **C1**, **C2**, **C3**, **C4**, **C5** y **C6**. Es utilizado por el Equipo de Marketing para revisar la trazabilidad de los productos de acuerdo al almacén o cedis donde se encontraban originalmente.

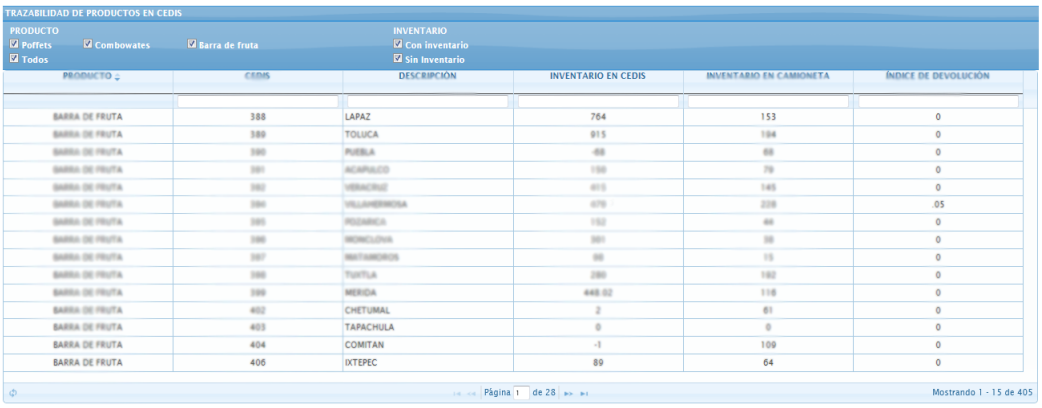


Figura 5.2: Reporte con filtros

Reporte con indicadores (Figura 5.3)

Este reporte cumple con las características **C1**, **C2**, **C3**, **C4**, **C5**, **C6** y **O1**. Es utilizado por el Equipo de Facturación Electrónica y Cartera para revisar la trazabilidad de las facturas, incluyendo indicadores del estatus de éstas.

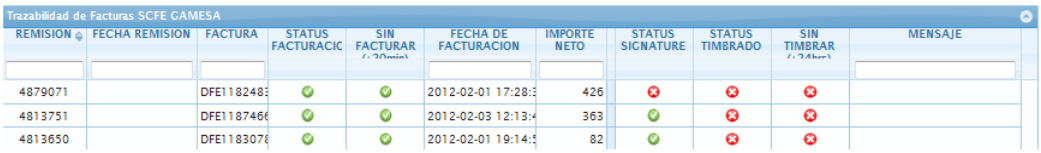


Figura 5.3: Reporte con indicadores

Reporte de concentrado (Figura 5.4)

Este reporte cumple con las características **C1**, **C2**, **C3**, **C4**, **C5**, **C6** y **O1**. Es utilizado por el Equipo de Recursos Humanos para conocer las incidencias presentadas en los legados.

Figura 5.4: Reporte de concentrado

Reporte compuesto (Figura 5.5)

Este reporte cumple con las características **C1**, **C2**, **C3**, **C4**, **C5**, **C6** y **O1**. Es utilizado por el Equipo Sustain DataStage para revisar el estatus de las ejecuciones actuales de las interfaces de sus procesos y el histórico de ejecuciones anteriores.

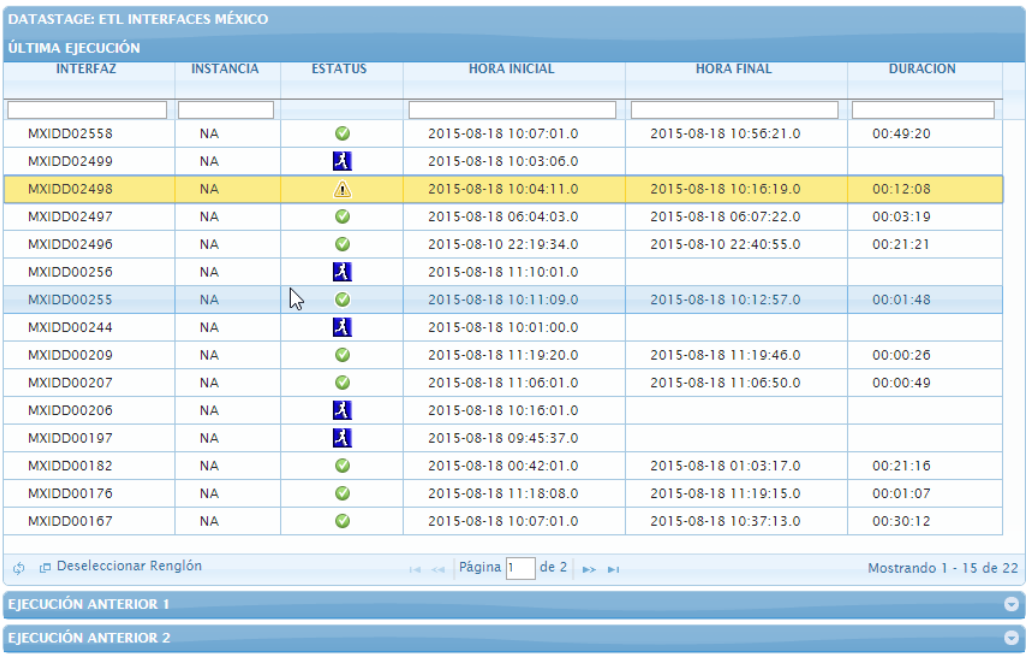


Figura 5.5: Reporte compuesto

Reporte con gráficas (Figura 5.6)

Este reporte cumple con las características **C1**, **C2**, **C3**, **C4**, **C5**, **C6** y **O1**. Es utilizado por el Equipo de Proyectos específicos para revisar la trazabilidad de los productos de acuerdo al almacén o cedis donde se encontraban originalmente, incluyendo las cantidades totales de productos en un transporte, el índice de devolución y el porcentaje por sección geográfica.

Reporte con sumas totales (Figura 5.7)

Este reporte cumple con las características **C1**, **C2**, **C3**, **C4**, **C5**, **C6** y **O1**. Es utilizado por el Equipo de Marketing para revisar la trazabilidad de los productos de acuerdo al almacén o cedis donde se encontraban originalmente, incluyendo las cantidades totales de productos en un transporte y el índice de devolución.

Figura 5.6: Reporte con gráficas

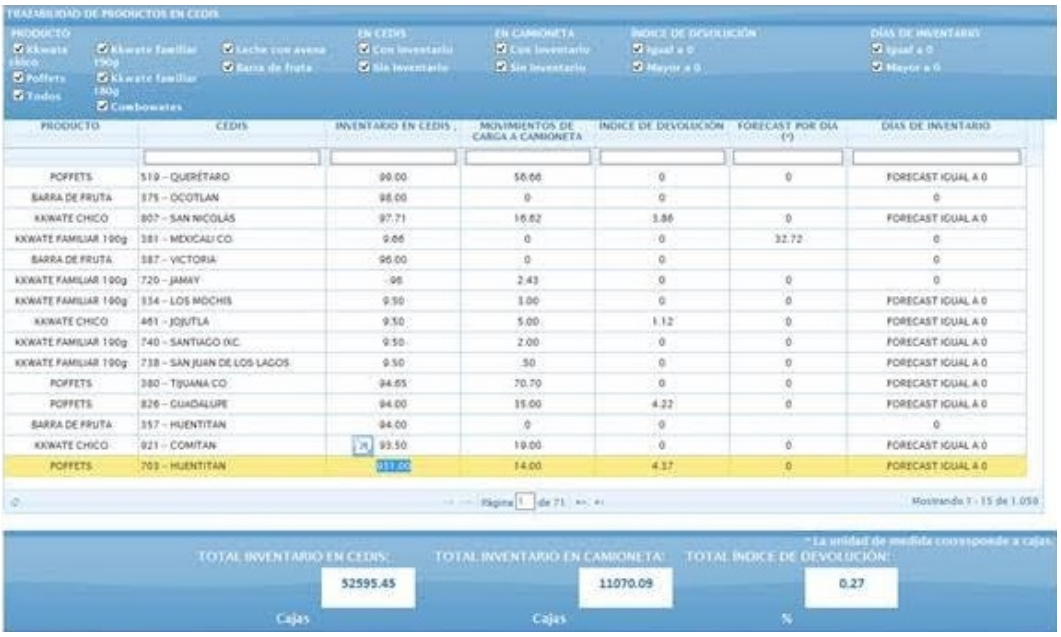


Figura 5.7: Reporte con sumas totales

5.3. Resumen

En este capítulo se ha descrito la atención a los requerimientos, asociando las características de los reportes con los requerimientos solicitados por la Empresa. Además, se muestran algunos ejemplos de los reportes generados y las características que posee cada uno de ellos.

Capítulo 6
Conclusiones

6.1. Resumen

En este capítulo se describen las conclusiones a este trabajo: se realiza un replantamiento del objetivo y los resultados obtenidos; se mencionan las contribuciones, el trabajo futuro y una breve semblanza sobre la estancia en la Facultad de ciencias de la U.N.A.M. .

6.2. Replantamiento del objetivo

Como se menciona en la Introducción, el objetivo principal es: "...la generación de reportes empresariales de fácil acceso, con información efectiva y con un diseño que proporcione un uso más sencillo al usuario. También la inclusión de filtros de búsqueda personalizados, entre otras características."

6.2.1. Discusión

Para alcanzar el objetivo se compararon algunas herramientas existentes de BI en el mercado que solucionaran los requerimientos de la empresa. Se analizó la viabilidad de utilizar dichas herramientas de acuerdo a los recursos (hardware, software y humanos) con los que disponía la empresa. De lo anterior, debido a las limitantes de las herramientas existentes para configurarse y a la baja disponibilidad de algunos recursos de la empresa, la idea más factible fue desarrollar una nueva aplicación para generar los reportes empresariales de BI.

Una vez decidido que se iba a desarrollar una nueva aplicacin para la generación de los reportes de BI,se investigaron las tecnologías existentes que permitieran reducir el tiempo de desarrollo. Java fue lenguaje seleccionado por su portabilidad y libertad para la generación de aplicaciones web. También se realizaron pruebas con los controladores existentes para manejar tablas de datos. Se eligió jqGrid por ser un controlador completamente configurable, por su forma asíncrona para actualizar los datos de las tablas y porque posee una interfaz sencilla y atractiva.

6.2.2. Resultados

Los reportes solucionaron las necesidades de la Empresa y son una herramienta útil y de fácil acceso. Su uso se extiende a diversas áreas de negocio como con:

- Business Foods - Equipo de Marketing.
- Facturación Electrónica e intercambio Electrónico - Equipo de Facturación Electrónica y Cartera.
- Integración SAP - Equipo Sustain DataStage.
- Trazabilidad de papa - Proyectos Específicos (proyecto BRAIN).
- HCM - Equipo de Recursos Humanos.

6.3. Contribuciones

- Los reportes cubren las necesidades solicitadas por la Empresa y, hasta ahora, se siguen utilizando.
- La implementación permite agregar nuevos reportes que pueden personalizarse de manera individual para adecuarse a nuevos requerimientos.
- Se emplean tecnologías de desarrollo existentes que hasta el momento no han sido utilizadas en la Empresa para el desarrollo de reportes de inteligencia de negocios.

6.4. Trabajo futuro

Actualmente, están en proceso las negociaciones con la Empresa para crear un Portal para la generación de reportes que incluya opciones para que un usuario operativo pueda editar algunas características de los reportes.

6.4.1. Semblanza

Por último, es importante mencionar que la formación recibida durante mi estancia en la Facultad de Ciencias me ha permitido, a lo largo de mi experiencia profesional, contar con los conocimientos y capacidades suficientes para resolver los problemas planteados por las distintas entidades en las que me he desempeñado.

Apéndice A
Configuración y Ambiente

Las actividades de este flujo tienen mayor relevancia en la fase de *Elaboración* y, en menor grado, en las fases de Construcción y Transición.

A.1. Configuración

Las tareas asociadas a la Configuración consisten en establecer la documentación apropiada para el control de cambios y mantenimiento de los reportes.

El control de cambios se lleva por medio del documento "Reportes - control de cambios". En dicho documento se lleva el historial con la fecha de creación y modificaciones al reporte. Las modificaciones se realizan por medio de la interfaz y conservando la estructura original del reporte. Los cambios que pueden realizarse serán: reglas para establecimiento de los filtros, formato de campos.

El mantenimiento corresponde a las siguientes acciones:

- Modificaciones a los reportes después de su entrega y validación.
- Revisiones de logs para incidencias no manifestadas previamente.
- Correcciones y mejoras a características funcionales o no funcionales.

A.2. Ambiente

El ambiente de desarrollo es el siguiente:

- Netbeans 6.9.
- Java Enterprise Edition 6.
- jqGrid 4.2.0.
- Java Servlet.
- Java Servlet Pages.
- CSS.
- jQuery UI - 1.9.2.
- Apache Tomcat 6.0.33.

Características del servidor de despliegue:

- Windows Server 2003 instalado.
- Procesador 2.41Ghz.

Características de la aplicación web:

- Árbol de clases Java.
- Árbol de archivos JSP.
- Archivo web.xml
- Archivo config.xml
- Archivo context.xml

Para facilitar el desarrollo iterativo y mantener el origen de la aplicación separado de los archivos compilados, se utiliza la estructura del directorio de la aplicación Java BluePrints [39].

La aplicación tiene la siguiente estructura:

- src / java: archivos fuente de Java para el módulo.
- web: páginas web (jsps), hojas de estilo, archivos de etiquetas e imágenes.
- web / WEB-INF: archivos de configuración.
- nbproject: archivos del proyecto NetBeans.

Bibliografía

[1] Peter G. W. Keen, Michael S. Scott Morton. *Decision Support Systems: An Organizational Perspective*, Addison-Wesley, 1978.

[2] Daniel J. Power. *Decision Support Systems: Concepts and Resources for Managers*, Greenwood Publishing Group, 2002

[3] Doug Daetz, Bill Banard, Rick Norman. *Customer Integration: The Quality Function Deployment (QFD) Leader's Guide for Decision-Making*, John Wiley and Sons, 1995.

[4] David Loshin. *Business Intelligence: The Savvy Manager's Guide*, Edition 2, Newnes, November 2012.

[5] Rajiv Sabherwal, Irma Becerra-Fernandez. *Business Intelligence: Practices, Technologies, and Management*, 2011.

[6] Oracle Corporation. *¿Qué es la Inteligencia de Negocios?*, Febrero 2011.

[7] B.A. Devlin, P.T. Murphy. *An architecture for a business and information system*, 1998.

[8] Bill Inmon. *Building the Data Warehouse*, Wiley, 1992.

[9] Carlos Coronel, Steven Morris, Peter Rob. *Bases de Datos: Diseño, Implementación y Administración*, Capítulo 13, 2013.

[10] Kevin S. Goff Lynn Langit Davide Mauri, Sahil Malik, and John C. Welch *Smart Business Intelligence Solutions with Microsoft SQL Server 2008*, Microsoft Press, 2009.

[11] Tricia Aanderud, Angela Hall, *Building Business Intelligence Using SAS*, SAS Institute, 2012.

[12] Adam Aspin *Business Intelligence with SQL Server Reporting Services*, Published by Apress, 2015.

[13] Scott A. Pugh, Randall S. Morin, and Barb A. Johnson *Implementing dashboards as a Business Intelligence tool in the forest inventory and analysis program.*, New Directions in Inventory Techniques and Applications Forest Inventory and Analysis (FIA) Symposium 2015, 317-320, 2015.

[14] Luis Muñiz. *El Reporting herramienta clave para mejorar la gestión empresarial*, Sisconges & Estrategia, 2012.

[15] Gabriel Mandricks. *Instant JqGrid*, July 2013.

[16] Kishore Bhamidipati. *SQL Programmer's Reference*, McGraw-Hill Osborne Media, 1998.

[17] Ramez A. Elmasri, Shamkant B. Navathe. *Fundamentos de Sistemas de Bases de Datos*, Pearson, Addison Wesley, 2013.

[18] Scott W. Ambler. *The Agile Unified Process*, `http://www.ambysoft.com/unifiedprocess/agileUP.html` 2006.

[19] Charles Edeki *Agile Unified Process*, International Journal of Computer Science and Mobile Applications, Vol.1 Issue.3, pg. 13-17, September 2013.

[20] Young David. *Software Development Methodologies*, 2013.

[21] Tom Negrino, Dori Smith. *JavaScript and Ajax for the Web: Visual QuickStart Guide*, 2009.

[22] Tom Negrino, Dori Smith. *JavaScript*, 8th Edition, 2012.

[23] Harvey Deitel, Peter Deitel. *Cómo programar en JAVA*, Pearson, Prentice Hall, 2004.

[24] Giulio Zambon *Beginning JSP, JSF and Tomcat: Java Web Development*, Apress, 2012

[25] Hewlett-Packard Development Company, L.P. *HP Business Service Management for the Windows operating system. Software Version: 9.01. Deployment Guide*, 2010.

[26] *Reporting in BI* `https://www.logianalytics.com/resources/bi-encyclopedia/reporting-bi/`, 2017.

[27] Jiawei Han, Micheline Kamber, Jian Pei *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2012.

[28] *XHTML, HTML y CSS* `https://www.w3.org/standards/webdesign/htmlcss`, Consultado en 2017

[29] *JSON* `https://www.json.org/`, Consultado en 2017

[30] *IDE Netbeans* `https://netbeans.org/features/index.html`, Consultado en 2017.

[31] *JavaServer Pages* `http://www.oracle.com/technetwork/java/jsp-138580.html`, Consultado en 2017.

[32] *Java Servlet* `http://www.oracle.com/technetwork/java/javaee/servlet/index.html`, Consultado en 2017.

[33] *Java SE Development Kit (JDK)* `http://www.oracle.com/technetwork/java/javase/jdk-8-readme-2095712.html`, Consultado en 2017.

[34] *Java EE at a Glance* `http://www.oracle.com/technetwork/java/javaee/overview/index.html`, Consultado en 2017.

[35] *Java Enterprise Edition (JEE) - Overview* https://docs.oracle.com/javaee/7/tutorial/overview.htm, Consultado en 2017.

[36] *Java Database Connectivity (JBDC)* http://www.oracle.com/technetwork/java/javase/jdbc/index.html, Consultado en 2017.

[37] *JavaScript* https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript?redirectlocale=en-US&redirectslug=JavaScript, Consultado en 2017.

[38] *Apache Tomcat* http://tomcat.apache.org/, Consultado en 2017.

[39] *Aplicación Web* https://docs.oracle.com/javaee/7/tutorial/webapp001.htm, Consultado en 2017.

[40] *Pentaho* https://help.pentaho.com/Documentation/8.1, Consultado en 2017.

[41] *JasperReports* https://netbeans.org/features/index.html, Consultado en 2017.

[42] *Portlet* https://community.dynatrace.com/community/display/BSMDOC/BSM+Portlets+Overview, Consultado en 2018.