DISEÑO Y PROTOTIPO DE UN GENERADOR DE REPORTES

CANDELARIA FIGUEROA GARCÍA

JOSÉ URBINA NÁJERA

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS D.T. Y C.

2006

DISEÑO Y PROTOTIPO DE UN GENERADOR DE REPORTES

CANDELARIA FIGUEROA GARCÍA

JOSÉ URBINA NÁJERA

Monografía presentada como requisito para grado en Minor en Ingeniería de Software.

Director:

GIOVANNY RAFAEL VÁSQUEZ MENDOZA

MsC. Ciencias Computacionales.

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS D.T. Y C.

2006

AGRADECIMIENTOS

Queremos expresar la mayor gratitud a Dios y a nuestros Padres por brindarnos la oportunidad de estudiar en una buena Universidad y al apoyo incondicional que nos han dado en el transcurso de la carrera.

Igual y especialmente a nuestro profesor Moisés Quintana, y a nuestro director, el Ingeniero Giovanny Vásquez por su grandiosa orientación y colaboración para la realización de este trabajo.

De la misma manera queremos agradecer a todas aquellas personas que directa o indirectamente colaboraron con la elaboración de este trabajo y a las que nos apoyaron en la consecución de nuestros objetivos académicos y personales.

Tabla de contenido

Índice de tablas

Índice de ilustraciones

1 RESUMEN

Esta monografía es el resultado de un trabajo de diseño e implementación de un prototipo de un generador de reportes, aplicando las técnicas de ingeniería del software y herramientas disponibles en la actualidad, con el cual se pretende presentar una propuesta más accesible y fácil de utilizar para los programadores y desarrolladores de software con poca experiencia. Para este fin, se analizaron generadores de reportes ya existentes en el mercado, y se observaron sus características principales, con el fin de entender la forma en que estos funcionan. Finalmente se desarrolla un prototipo de generador de reportes, partiendo desde las etapas de análisis y diseño de éste, siguiendo las técnicas propuestas por la ingeniería de software, hasta la implementación final.

El trabajo inicia mostrando las generalidades de los generadores de reportes y de sus características principales, para después mostrar los beneficios que se pueden obtener al usar un generador de reportes.

A continuación se hace una introducción a los reportadores analizados para la realización de esta monografía. En ella se muestran sus características específicas, así como su accesibilidad para el público y su facilidad de uso.

En el siguiente capítulo se muestran las tecnologías utilizadas para la creación del prototipo que acompaña a este estudio. Las diferentes tecnologías

utilizadas incluyen librerías para el lenguaje Java, lenguajes de marcado y de hojas de estilo.

Después se muestran las generalidades del reportador que se realizará a lo largo de este estudio monográfico. Aquí, además, se hacen el análisis y el diseño del prototipo del reportador que va a desarrollarse. En estos se muestran las características y los requerimientos del generador de reportes, los cuales serán detallados mediante casos de uso y diagramas UML.

Para finalizar, se hacen unas conclusiones sobre el trabajo realizado y se dan recomendaciones para los lectores que se interesen por el proyecto y quieran hacer aportes a éste.

2 INTRODUCCIÓN

Los generadores de reportes son herramientas que han adquirido una gran importancia a nivel empresarial, debido a su gran versatilidad a la hora de crear documentos con formatos exclusivos de cada empresa; y a su facilidad para crear documentos de gran extensión, ya que se les puede indicar un origen de datos y ellos generan automáticamente el documento con la información contenida en éste. Su gran éxito ha conllevado a que se hayan creado varias de estas herramientas para distintos sistemas operativos, ya sea en forma de aplicaciones o como librerías para ser usadas por diferentes lenguajes de programación.

Es por esto que los generadores de reportes se perfilan como una de las aplicaciones con más demanda en el futuro, por su gran versatilidad y por sus mejoras constantes, lo que los hace más confiables a nivel empresarial y les permite acoplarse mejor a los sistemas de información de la empresa.

Este trabajo muestra el análisis, el diseño y partes del código de un generador de reportes que se desarrolló con el fin de mostrar una nueva propuesta para los programadores y desarrolladores de software. También se muestran las diferentes tecnologías que se utilizaron para la creación del prototipo diseñado. Cabe anotar que los términos generador de reportes y reportador se utilizarán indistintamente, y que cada uno hace referencia al otro.

3 GENERADORES DE REPORTES

3.1 GENERALIDADES

La información siempre ha sido un recurso invaluable a lo largo de la historia. En la actualidad, gracias al avance de los computadores y al auge de Internet, los volúmenes de información que se manejan son cada vez mayores; y cuando se hace necesario procesar esta información, se necesita obtener sólo la información necesaria, darle formato y guardarla en un archivo o imprimirla. Es entonces cuando aparecen los generadores de reportes, también llamados reportadores.

Los reportes son el medio principal para obtener la información que se necesita. En una empresa, el usuario los utiliza para dar a conocer lo que está ocurriendo en la empresa durante un periodo de tiempo determinado. La principal fuente de datos para los reportes son las bases de datos y brindan al usuario final a posibilidad de consultar y publicar lo que las bases de datos poseen, pero hay un problema: generar un reporte a veces requiere requiere ciertos conocimientos de bases de datos o de algunas herramientas de software.

Los generadores de reportes pretenden eliminar esta complejidad al situarse en un punto intermedio entre los detalles técnicos y el usuario final, de forma que un usuario pueda generar sus reportes de manera rápida y sencilla sin

necesidad de depender de los usuarios técnicos y sin necesidad de conocer aspectos técnicos como el lenguaje SQL, unión de tablas y conceptos similares.

Los reportadores han sido una gran ayuda para las empresas, por su facilidad de acoplamiento con los sistemas de información disponibles a nivel empresarial, la que los convierte en poderosas herramientas para recuperar información de los almacenes de datos que posea la empresa de una forma rápida y sencilla.

Los reportadores son herramientas que obtienen información de un origen de datos previamente establecido, el cual puede ser un archivo o una base de datos; le dan formato a gusto del usuario y finalmente permiten enviar la información ya formateada a la impresora o exportarla a otro tipo de archivo a elección del usuario.

Muchos reporteadores se encuentran en forma de aplicaciones listas para ser usadas, pero hay reporteadores que se distribuyen como librerías para lenguajes de programación. La mayoría de estas librerías se encuentran para lenguajes como C/C++ o Java, que son los lenguajes de programación más populares en la actualidad, aunque también se encuentran librerías para otros lenguajes, como Delphi o Visual Basic.

La ventaja de que los reporteadores se encuentren como aplicaciones es que

se pueden usar apenas se obtienen; mientras que si se obtienen como librerías, se puede crear una interfaz gráfica de acuerdo a las necesidades específicas de la empresa, e incluso se podría adecuar el funcionamiento del reportador, si la licencia de éste lo permite.

3.2 BENEFICIOS QUE SE OBTIENEN DE LOS REPORTEADORES

Los generadores de reportes son herramientas que ayudan a obtener información de un origen de datos de una forma rápida y sencilla. A continuación se citan los requisitos de un generador de reportes que están llegando a ser cada vez más importantes en las empresas.

3.2.1 RAPIDEZ PARA CREAR DOCUMENTOS

Los generadores de reportes permiten crear documentos de gran tamaño en pocos pasos, debido a su capacidad para obtener los datos necesarios de un origen especificado por el usuario. Estos orígenes de datos son en su mayor parte bases de datos, pero también se suelen usar archivos de texto plano, archivos XML e incluso algunos de los reporteadores más poderosos permiten usar archivos binarios como entrada (por ejemplo, archivos de Microsoft Excel).

3.2.2 COMPATIBILIDAD CON VARIAS BASES DE DATOS

Los reporteadores que pueden utilizar bases de datos deben ser compatibles con la mayoría de sistemas gestores de bases de datos disponibles en el mercado, para poder amoldarse a las necesidades del entorno donde son desplegados.

3.2.3 PLANTILLAS PERSONALIZADAS

Los reporteadores actuales, para evitar que el usuario tenga que diseñar su reporte desde cero cada vez que desee crear un reporte, incluyen formatos prediseñados en forma de plantillas para que el usuario escoja la que necesite. Además, el usuario puede diseñar sus propias plantillas de acuerdo a sus gustos y/o necesidades.

3.2.4 VARIOS FORMATOS DE SALIDA

Los generadores de reportes actuales tienen varias opciones de salida, la más popular es la salida directa a la impresora; pero además los reporteadores pueden guardar los reportes en un formato propio de cada reportador, o pueden exportar los reportes a algunos de los formatos de documento más populares en la actualidad, como son HTML, XML, PDF, XLS (Microsoft Excel), archivos de texto plano, entre otros formatos.

3.3 CASOS DE ESTUDIO

En este capítulo se podrán conocer algunos de los reporteadores más populares en el mercado, así como sus principales características; y las ventajas y desventajas que conlleva su uso por parte de los programadores y desarrolladores principiantes.

3.3.1 JASPER REPORTS

JasperReports es un producto desarrollado por JasperSoft Corporation, empresa que pertenece a la comunidad del software libre, escrito 100% en Java, con licencia LGPL (GNU Lesser General Public License), lo cual lo hace accesible a cualquier persona que quiera utilizarlo. Se puede usar como aplicación o se pueden usar sus librerías en programas propios. Entre sus características se encuentran un diseño visual de reportes muy intuitivo, el cual permite diseñar de forma sencilla encabezados y pies de página, columnas y grupos; además permite hacer cálculos, formatear texto e introducir imágenes.

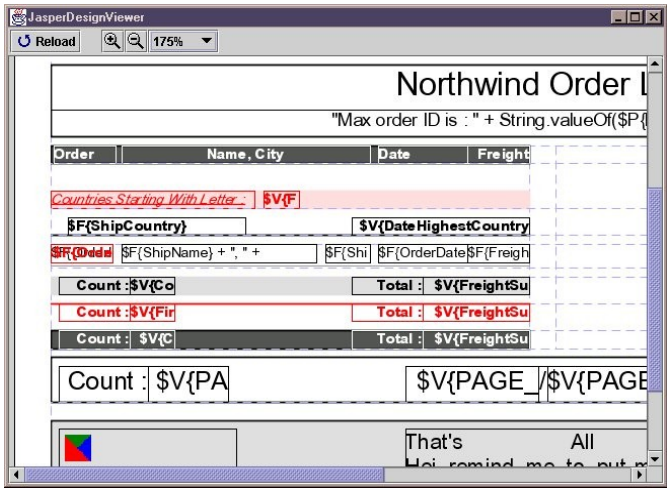

Imagen 3.1. *Vista de diseño de Jasper Reports*

Entre las ventajas de usar este reportador se encuentra el hecho que, gracias a que es un producto libre, los usuarios tienen disponibilidad de obtener el código fuente y poder adaptar el reportador a sus necesidades. Además, permite enviar los reportes a la impresora o exportarlos a formatos como PDF, XML, HTML, XLS (Microsoft Excel) o CSV (comma-separated values, valores separados por comas), el cual es un formato de texto plano.

Las desventajas de usar JasperReports, son que el reportador es un poco difícil de manejar para un programador principiante y que sólo se puede usar en aplicaciones Java, lo que excluye a programadores que usen otros lenguajes.

3.3.2 CRYSTAL REPORTS

Crystal Reports es un generador de reportes desarrollado por Business Objects. Este reportador ha sido la herramienta líder en el campo de los generadores de reportes desde hace muchos años. Al igual que Jasper Reports, puede usarse como aplicación o como librerías para diferentes lenguajes de programación. También posee un entorno visual muy intuitivo. Entre sus ventajas se pueden citar su compatibilidad con muchas bases de datos, su integración con otras herramientas de Business Objects y con herramientas de Microsoft, además que puede exportar a formatos propios de Microsoft. Además, cuenta con soporte ofrecido por Business Objects.

Sus desventajas son que, debido a su licencia como software propietario, no se tiene acceso al código fuente de éste. Su precio es muy elevado para un programador principiante, y su exceso de funciones puede confundir a los programadores principiantes. Además, si se usa con plataformas diferentes al sistema Microsoft Windows, sólo se puede usar con el lenguaje Java, perdiendo gran parte de su funcionalidad, ya que fue diseñado pensando en la tecnología .NET, de Microsoft.

4 TECNOLOGIA UTILIZADA

A lo largo de este capítulo se hablará brevemente de las diferentes tecnologías que se utilizaron en la creación del prototipo de reportador desarrollado para este estudio.

4.1 JAVA API FOR XML PROCESSIONG (JAXP)

La JAXP, como su nombre lo indica, es una API para el uso de XML en aplicaciones Java. Esta API provee la capacidad de analizar (parse) y de validar documentos XML. Las dos APIs básicas para el análisis de documentos son la interfaz DOM (Document Object Model) y la SAX (Simple API for XML). Aparte de las interfaces de análisis, esta API provee una interfaz para XSLT, que provee transformaciones estructurales y de datos sobre un documento XML. Para el desarrollo del prototipo, se usó por la interfaz para XSLT, que provee el paquete javax.xml.transform, y que está disponible en la versión 1.2 de esta API, que hace parte de Java 1.5.

4.2 XML

XML es la sigla de eXtensible Markup Language (lenguaje de marcado extensible), creado por el W3C (World Wide Web Consortium). Aunque es un lenguaje de marcado, al igual que HTML, tiene muchas más aplicaciones que HTML. XML en sí no es un lenguaje, sino que es un metalenguaje, o sea, es un

lenguaje que permite crear lenguajes adecuados para usos determinados. Los elementos que lo componen pueden dar información sobre lo que contienen, no necesariamente su estructura o presentación física, como ocurre con HTML. XML, aunque nació para ser aplicado en Internet como sucesor de HTML, donde se separa la presentación del contenido, también es ampliamente utilizado como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de datos entre aplicaciones. XML, al igual que HTML se basa en archivos de texto plano donde se usan etiquetas para delimitar los elementos de un documento. Sin embargo, XML define estas etiquetas en función del tipo de datos que está describiendo y no de la apariencia final que tendrán en pantalla o en la copia impresa, además de permitir definir nuevas etiquetas y ampliar las existentes.

XML es una tecnología sencilla que tiene a su alrededor otras tecnologías que la complementan y la hacen mucho mas grande y con unas posibilidades mucho mayores. Esta tecnología tiene un papel muy importante en la actualidad ya que tiende a la compatibilidad entre sistemas, es la tecnología que permitirá compartir la información de una manera segura, fiable y fácil. Debido a esto, se han desarrollado varios lenguajes basados en XML para ampliar sus aplicaciones. Entre estos podemos encontrar XSLT, XHTML (una extensión de HTML que cumple las reglas de XML) y XSL-FO, entre otros. Además, existen también versiones para usos específicos, como MathML (fórmulas matemáticas), SVG (gráficos vectoriales) y RSS (sindicación de noticias).

A continuación se ilustra un ejemplo de documento XML:

```
1 <?xml version="1.0" encoding="utf-8"
standalone="yes"?>
2 <biblioteca>
3       <libro>
4             <titulo>El Hobbit</titulo>
5             <autor>J. R. R. Tolkien</autor>
6             <genero>Fantasía</genero>
7       </libro>
8 </biblioteca>
```

Tabla 4.1. *Ejemplo de documento XML.*

La primera línea indica al analizador XML que este documento es un documento XML, y además provee otros atributos como son la versión del estándar XML que se está usando, la codificación (el código de caracteres, en este caso se usó Unicode), y por último si el documento es independiente o si va acompañado de una DTD (Document Type Definition). Si el documento estuviera asociado a una DTD para demostrar su validez, el atributo *standalone* tuviera el valor "no", y en la segunda línea se especificaría el origen de la DTD, mediante una dirección URL.

La siguiente línea, <*biblioteca*> indica el inicio del elemento principal (la raíz) del documento, que para el caso propuesto sería una representación de una biblioteca. A continuación se hace una representación de los elementos que componen la biblioteca (*libros*), y cada libro incluye una serie de datos que describen las propiedades específicas de ese libro (*nombre*, *autor* y *género*). Pueden haber tantos libros como se desee, pero sólo puede haber una biblioteca (esto porque en XML un documento debe tener solamente una raíz).

XML es de vital importancia para este trabajo monográfico, ya que es el formato principal de exportación del prototipo desarrollado, además las plantillas de formato para los reportes HTML están escritas con XSLT, lenguaje del cual se hablará a continuación.

4.3 XSLT

XSLT o XSL Transformations es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML, como por ejemplo PDF. Hay que resaltar que el documento XML original no es modificado, más bien se crea un nuevo documento basado en el contenido de un documento existente. Las hojas de estilo XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla: unidas al documento fuente a transformar, esas reglas de plantilla alimentan a un procesador de XSLT, el cual realiza las transformaciones deseadas colocando el resultado en un archivo de salida o, como en el caso de una pagina web, directamente en un dispositivo de presentación, como el monitor de un usuario.

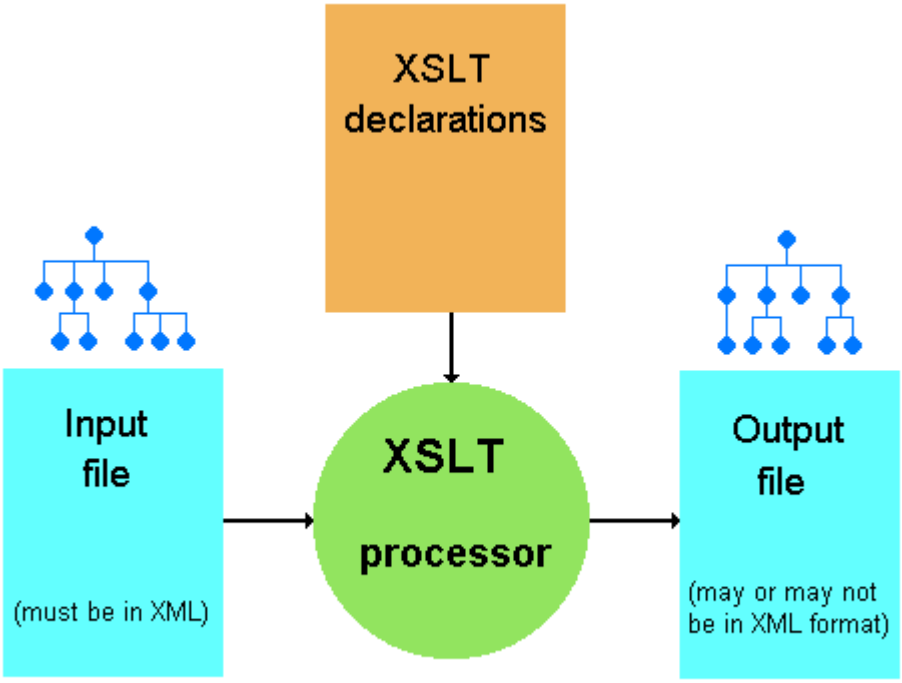

Imagen 4.1. *El proceso de transformación XSLT.*

Actualmente, XSLT es muy usado en la edición web, generando páginas HTML o XHTML. La unión de XML y XSLT permite separar contenido y presentación, aumentando así la productividad. Para este estudio monográfico, se usó XSLT para escribir las plantillas para los reportes.

Lo que se muestra en la siguiente página es parte del código de la plantilla XSLT que se incluye con el prototipo realizado para este estudio monográfico:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3       <xsl:output method="html" indent="yes"/>
4       <xsl:template match="/">
5               <html>
6               <xsl:apply-templates select="report"/>
7               </html>
8       </xsl:template>
9
10      <xsl:template match="report">
11              <head>
12                      <title><xsl:value-of select="title"/> - Generado con
JCReports</title>
13              </head>
14              <body>
15                      <xsl:apply-templates select="image"/>
16                      <xsl:apply-templates select="title"/>
17                      <xsl:apply-templates select="date"/>
18                      <xsl:apply-templates select="comments"/>
19                      <xsl:apply-templates select="table"/>
20              </body>
21      </xsl:template>
22 ...
23 </xsl:stylesheet>
```

Tabla 4.2. *Ejemplo de documento XSLT. Este es parte del código de la plantilla incluida con el prototipo.*

La primera línea define al documento como un documento XML, esto es así porque toda hoja de estilo XSLT es un documento XML. La segunda línea define el documento como una hoja de estilo XSLT, con su número de versión y el espacio de nombres donde obtener las etiquetas que conforman el lenguaje. La tercera línea define el tipo de salida que se va a obtener, así como si se quiere que la salida esté adecuadamente indentada.

Las líneas 4 a 8 definen una plantilla XSLT. El atributo *match* se usa para asociar la plantilla con un elemento del documento XML. Por ejemplo, si en el documento XML se tiene un elemento <*libro*>, en la hoja de estilo XSLT puede haber una plantilla con la declaración <*xsl:template match="libro"*>. También se puede usar el elemento / como valor del atributo *match* para referirse al documento entero. En la línea 6 se puede observar otra sentencia muy importante de XSLT, <*xsl:apply-templates*>. Esta sentencia, cuando se acompaña del atributo *select*, aplica la plantilla que tenga el atributo *match* correspondiente. Por ejemplo, si se tiene una plantilla declarada como <*xsl:template match="libro"*> y se use la sentencia <*xsl:apply-templates select="libro"*>, el procesador XSLT reemplaza esta sentencia con el contenido de la plantilla diseñada para el elemento *libro*. Si no se especifica un elemento mediante *select*, se aplicarán todas las plantillas de los elementos hijos del elemento donde se halle contenida esta sentencia.

Por último, la sentencia <*xsl:value-of="elemento" /*> se utiliza para recuperar el valor de un elemento XML. Regresando al ejemplo de la biblioteca, si se crea una plantilla para el elemento *libro*, y dentro de ésta aparece la sentencia <*xsl:value-of select="titulo" /*>, el procesador XSLT reemplazará esta sentencia por el valor del elemento *titulo*, y en el resultado final se tendría “El Hobbit”.

4.4 HTML

El HTML, acrónimo inglés de HyperText Markup Language (lenguaje de etiquetado de documentos hipertextual), es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo Internet Explorer, Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

Originalmente definido por Tim Berners-Lee y desarrollado posteriormente por el Grupo de Trabajo en Ingeniería de Internet (Internet Engineering Task Force, IETF), HTML es actualmente un estándar internacional (ISO/IEC 15445:2000). Las especificaciones posteriores de HTML son mantenidas por el World Wide Web Consortium (W3C).

La última versión de HTML es la 4.01, y no va a haber nuevas versiones de HTML. La herencia de HTML se mantiene en XHTML, una reimplementación de HTML 4 como una aplicación XML 1.0 y supone la base para la evolución de HTML.

Un ejemplo de código HTML sería el siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
 <html lang="es">
  <head>
   <title>Ejemplo</title>
  </head>
  <body>
   <p>ejemplo</p>
  </body>
 </html>
```

Tabla 4.3. *Ejemplo de documento HTML.*

La etiqueta *<!DOCTYPE>* al inicio indica cuál es la versión del estándar HTML que utiliza el documento, verificando que el contenido se apegue a la Definición de Tipo de Documento (DTD, Document Type Definition) que se especifica al interior de la etiqueta. La etiqueta *<html>* indica el inicio del documento HTML, y su atributo *lang* indica el idioma del contenido del documento. A continuación se presenta la cabecera del documento, iniciando con la etiqueta *<head>* y finalizando con *</head>*. En la cabecera se incluye el título del documento, que será mostrado en el borde de la ventana del navegador utilizado. Después aparece el cuerpo del documento, delimitado entre las etiquetas *<body>* y *</body>*. En el ejemplo, el cuerpo sólo está conformado por un párrafo (incluido entre las etiquetas * * y * *). Finalmente, la etiqueta *</html>* indica el final del documento.

HTML fue incluido en este estudio monográfico ya que es uno de los formatos a los que puede exportar el prototipo desarrollado.

4.5 XERCES

Xerces es una familia de paquetes de software de análisis y manipulación de documentos XML, y parte del proyecto XML de Apache. Xerces proporciona análisis y generación de documentos XML de gran calidad. En particular la versión para Java-XML fue uno de los primeros analizadores XML para Java y es una de las implementaciones más populares en el mundo Java.

Xerces es una implementación completa de la Java API for XML Processing (JAXP). Esta librería implementa varias APIs estándar para el análisis de XML, entre ellas DOM, SAX y SAX2. Además, Xerces en su versión 2.8.0 incluye las siguientes características:

- XML 1.0, tercera edición y XML 1.1, primera edición.
- Soporta espacios de nombres de XML 1.0 y XML 1.1.
- XML (XML Inclusions, Xinclude), versión 1.0.
- Modelo de Objetos de Documento (DOM, Document Object Model) niveles 2 y 3.
- Simple API for XML (SAX) 2.0.2, núcleo y extensiones.
- Java APIs for XML Processing (JAXP) 1.3.
- Segunda edición de los tipos de datos y estructuras de XML Schema 1.0.

Xerces es usado en este proyecto como una dependencia, ya que Xalan, el procesador de hojas de estilo XSLT necesita un parser XML para analizar la hoja de estilo y posteriormente aplicar la transformación a HTML.

4.6 XALAN

Xalan es un componente popular de código abierto de la fundación Apache (Apache Software Foundation) que implementa el lenguaje de transformaciones de XML (XSLT) y el lenguaje de consultas XML (Xpath). Este procesador está disponible como librería para los lenguajes de programación C++ y Java.

Xalan-Java es un procesador XSLT que sirve para transformar documentos XML en HTML, texto plano, u otros tipos de documento XML. Implementa las Transformaciones XSL (XSL Transformations, XSLT) versión 1.0 y el lenguaje XML Path (XPath) versión 1.0 y puede ser usado desde la línea de comandos, en un applet o un servlet, o como un módulo en otro programa.

Xalan-Java implementa la interfaz javax.xml.transform, de la API JAXP (Java API for XML Processing) 1.3. Esta interfaz proporciona un marco de trabajo (framework) modular y es un API estándar para ejecutar transformaciones XML, y utiliza propiedades del sistema para determinar qué transformador, así como qué analizador (parser) XML debe usar.

Xalan-Java también implementa la interfaz javax.xml.xpath de JAXP 1.3, la cual provee un API para la evaluación de expresiones XPath y el acceso al ambiente de evaluación.

Xalan-Java también soporta SAX 2 y DOM nivel 3.

Por defecto, Xalan-Java usa Xerces-Java, pero puede ser configurado mediante las propiedades del sistema para trabajar con otros parsers XML. La entrada puede ser presentada en forma de flujo de etiquetas XML (desde un URI, un flujo de caracteres o de bytes, u otra transformación), un InputStream de SAX, o un DOM Node (Nodo DOM).

Xalan-Java ejecuta las transformaciones especificadas en la hoja de estilos XSL y empaqueta una secuencia de eventos SAX que pueden ser serializadas a un flujo de salida (OutputStream), a un Writer, o puede ser usado para construir un árbol DOM, o como entrada para otra transformación.

Entre las características de Xalan se encuentran:

- Incluye un intérprete para ser usado en un entorno de depuración o como herramienta, y un compilador (XSLTC) para ser utilizado en un entorno de ejecución de alto desempeño.
- Implementa las siguientes especificaciones W3C: XSL Transformations (XSLT) Version 1.0 y XML Path Language (XPath) Version 1.0.
- Implementa Java API for XML Processing (JAXP) 1.3, y también permite SAX 2 y DOM nivel 3.
- Implementa la API de XPath en JAXP 1.3.
- Puede ser configurado para trabajar con cualquier analizador XML, como por ejemplo Xerces-Java, o cualquiera que implemente JAXP 1.3.

- Puede procesar flujos (streams), SAX o DOM como entradas, y permite salidas de flujos, SAX o DOM.
- Las transformaciones permiten ser encadenadas (la salida de una transformación puede ser la entrada de otra).
- Puede ser ejecutado desde la línea de comandos para permitir las transformaciones archivo por archivo.
- Puede ser usado en un servlet para transformar documentos XML en HTML y servir los resultados a los clientes.

5 ESTUDIO DEL PROTOTIPO DE REPORTADOR QUE SE VA A REALIZAR

5.1 CARACTERISTICAS

El generador de reportes que se va a realizar a lo largo de esta monografía es sólo un prototipo, y como tal sólo va a cubrir las funcionalidades básicas de un generador de reportes. Se escogió como lenguaje de programación el lenguaje Java, porque es un lenguaje independiente de la máquina, lo que significa que un programa en Java funciona en cualquier sistema operativo sin necesidad de ser recompilado, siempre que haya una máquina virtual Java (JVM) para el sistema operativo correspondiente. Esto es una ventaja, porque el reportador puede funcionar en los sistemas operativos más modernos y populares, como son Linux y Windows. Además, el lenguaje ofrece una gran cantidad de clases que son de mucha ayuda a la hora de diseñar el reportador, lo cual evita tener que escribir código para crear funciones y estructuras básicas, conllevando con esto a un ahorro de tiempo significativo en la etapa de implementación.

El prototipo funcionará como una aplicación, pero se intentará implementar los paquetes de la forma más modular posible, para que las clases que conforman el núcleo del reportador puedan ser usadas como librerías que puedan ayudar a los programadores a crear sus propios reportes.

El prototipo tendrá como origen de datos las bases de datos, y el motor de bases de datos escogido para interactuar con el prototipo es MySQL, debido a su facilidad de uso y a que su licencia permite usarlo libremente para proyectos no comerciales.

Los formatos de exportación escogidos son XML y HTML. Estos formatos fueron escogidos porque son estándares recomendados por el W3C, además pueden ser visualizados por un navegador web, lo que permite que puedan ser desplegados con facilidad en un sitio Web, y hacen innecesarias herramientas especiales para visualizar los reportes, ya que los sistemas operativos modernos incluyen por lo menos un navegador web. Hay una razón especial para escoger XML, y es que, debido a que su finalidad es almacenar contenido, se pueden usar reportes exportados a XML como una entrada de otros sistemas, por ejemplo bases de datos o aplicaciones especializadas.

5.2 ANALISIS

En el documento de análisis básicamente se listan los requerimientos del sistema a desarrollar. A cada requerimiento se le asigna un código único para diferenciarlos, así como una prioridad, que indica el grado de importancia del requerimiento para el sistema. Estos requerimientos se dividen en funcionales y no funcionales. Los requerimientos funcionales de un sistema, según Ian Sommerville[1], describen la funcionalidad o los servicios que se espera que éste provea. Mientras, los requerimientos no funcionales son aquellos que no se refieren directamente a las funciones específicas del sistema, sino a las propiedades de éste. Ejemplos de requerimientos no funcionales son la fiabilidad, la seguridad y la interoperabilidad con otros sistemas.

Los requerimientos funcionales son explicados detalladamente mediante casos de uso. Cada caso de uso contiene los siguientes elementos:

- Una identificación y un nombre.
- Una corta descripción del caso de uso.
- Los actores que intervienen en él.
- Una prioridad.
- Los requerimientos no funcionales que satisfacen (si se satisfacen uno o más requerimientos no funcionales).

1. Sommerville, Ian. Requerimientos funcionales y no funcionales. En: Ingeniería de software, 6ª ed. p. 100.

- Una condición.
- Un accionamiento, que no es más que una descripción del evento que activa el caso de uso.
- Un flujo de eventos principal.
- Uno o varios flujos alternativos, que representan los pasos a seguir en caso que hayan errores en el flujo principal.
- Una poscondición, que indica el resultado de la ejecución del caso de uso.

En esta etapa se utilizan varios diagramas UML, para ayudar al lector a comprender el análisis:

- Diagrama de caso de uso, para mostrar las interacciones de los actores con los casos de uso del sistema.
- Diagrama de actividades, que ilustra el flujo de ejecución del sistema.

5.2.1 REQUERIMIENTOS DEL SISTEMA

Requerimientos Funcionales	***Descripción del requerimiento***
E0-01	El sistema debe conectarse a un servidor de bases de datos MySQL.
E0-02	El usuario debe especificar los parámetros de conexión.
E1-01	El sistema debe permitir al usuario elegir la base de datos que va a utilizar.

Requerimientos Funcionales	*Descripción del requerimiento*
E1-02	El sistema debe mostrar al usuario las tablas y los campos de la base de datos.
E2-01	El sistema debe permitir al usuario seleccionar los campos de la base de datos que el usuario desee.
E3-01	El sistema debe permitir al usuario remover los campos que éste haya escogido y que ya no desee utilizar.
E4-01	El sistema debe desconectarse del servidor de bases de datos cuando el usuario lo desee o al cerrar el programa.
E5-01	El sistema debe exportar la información al formato XML.
E5-02	El sistema debe exportar la información al formato HTML.
A1-01	El sistema debe permitir al usuario insertar información adicional al reporte. Esta información adicional consiste de: • Título • Comentarios • Imagen
A1-02	El sistema debe insertar automáticamente en el reporte la fecha de creación de éste.
A2-01	El sistema debe permitir al usuario seleccionar una plantilla de exportación.
N1-01	La sección de selección de campos y la sección de información adicional deben estar en pestañas distintas.

Requerimientos Funcionales	*Descripción del requerimiento*
N2-01	Las opciones del sistema deben ser accesibles a través de una barra de menú.

Tabla 5.1. *Requerimientos funcionales del sistema.*

Requerimientos no funcionales	*Descripción del requerimiento*
E0-100	El programa debe presentar comodidad al usuario al seleccionar la base de datos.
E1-100	El programa debe ser seguro. No debe modificar el contenido de las bases de datos.
A1-100	El programa debe ser portable. Debe funcionar en la mayor cantidad de sistemas operativos posibles.

Tabla 5.2. *Requerimientos no funcionales del sistema.*

5.2.2 CASOS DE USO DEL SISTEMA

Identificación y nombre del caso de uso	E0 - Conectar a un servidor de bases de datos.
Descripción	Este procedimiento se encarga de hacer una conexión a un servidor de bases de datos, dados unos parámetros de conexión.
Actores	Usuario, sistema, servidor de bases de datos.
Prioridad	Esencial.
Requerimientos no funcionales	E0-100.
Condición	El sistema debe estar iniciado. Debe existir un servidor de bases de datos MySQL en la red.
Accionamiento	El usuario debe hace clic sobre la opción “Conectar” en el menú respectivo.
Flujo de eventos	1. El usuario ingresa los parámetros de conexión que le son solicitados. Estos parámetros son: ● Motor de bases de datos. ● Servidor. ● Puerto. ● Base de datos. ● Usuario. ● Contraseña. 2. El usuario presiona el botón “Aceptar”. 3. El sistema envía los parámetros de conexión al servidor.
Flujo alternativo	Si los parámetros ofrecidos por el usuario son erróneos o el servidor de bases de datos no está disponible, se muestra un mensaje de error. Si el usuario presiona el botón ”Cancelar” la conexión no se realiza.
Poscondición	Se ha realizado una conexión con la base de datos.

Tabla 5.3. *Caso de uso E0.*

Identificación y nombre del caso de uso	E1 - Seleccionar una base de datos.
Descripción	Este procedimiento se encarga de usar la base de datos que haya especificado el usuario.
Actores	Usuario, sistema, servidor de bases de datos.
Prioridad	Esencial.
Requerimientos no funcionales	E0-100.
Condición	El sistema debe tener una conexión activa con el servidor de bases de datos.
Accionamiento	El usuario debe hace clic sobre la lista de bases de datos que muestra el sistema.
Flujo de eventos	1. El usuario hace clic sobre una base de datos de la lista de bases de datos que se le presenta. 2. El sistema envía una petición al servidor para usar la base de datos seleccionada por el usuario, y también pide una descripción de cada tabla que conforma la base de datos. 3. El sistema muestra un lista en forma de árbol que muestra la estructura (las tablas y campos) de la base de datos seleccionada.
Flujo alternativo	
Poscondición	Se ha seleccionado una base de datos.
Comentarios	Si se selecciona una base de datos vacía, el sistema no mostrará error, pero el árbol se mostrará vacío.

Tabla 5.4. *Caso de uso E1.*

Identificación y nombre del caso de uso	E2 – Seleccionar campos para consulta.
Descripción	Este procedimiento se encarga de agregar campos de la base de datos para la consulta que se usa para generar el reporte.
Actores	Usuario, sistema.
Prioridad	Esencial.
Requerimientos no funcionales	
Condición	El sistema debe tener una conexión activa con el servidor de bases de datos. El usuario debe escoger una base de datos para consultar.
Accionamiento	El usuario debe hace clic sobre la lista de tablas y campos que conforman la base de datos.
Flujo de eventos	1. El usuario hace clic sobre una de las tablas que conforman la base de datos de la lista que se le presenta. Al seleccionar una tabla esta muestra los campos que la conforman. 2. El usuario hace doble clic en los campos que desee consultar. 3. El sistema agrega los campos seleccionados a una lista y muestra la lista al usuario.
Flujo alternativo	
Poscondición	Se han seleccionado los campos para consulta.

Tabla 5.5. *Caso de uso E2.*

Identificación y nombre del caso de uso	E3 – Remover campos para consulta.
Descripción	Este procedimiento se encarga de remover campos de la consulta que se usa para generar el reporte.
Actores	Usuario, sistema.
Prioridad	Esencial.
Requerimientos no funcionales	
Condición	El sistema debe tener una conexión activa con el servidor de bases de datos. El usuario debe hacer seleccionado por lo menos un campo para consulta.
Accionamiento	El usuario debe seleccionar el campo a remover.
Flujo de eventos	1. Después de seleccionar el campo a remover, el usuario debe hacer clic en el botón “Remover”. 2. El sistema crea una nueva lista de campos, pero sin el campo seleccionado. 3. El sistema muestra la nueva lista al usuario.
Flujo alternativo	
Poscondición	Se ha removido un campo para consulta.

Tabla 5.6. *Caso de uso E3.*

Identificación y nombre del caso de uso	E4 – Desconectar del servidor de bases de datos.
Descripción	Este procedimiento se encarga de cerrar la conexión al servidor de bases de datos.
Actores	Usuario, sistema.
Prioridad	Esencial.
Requerimientos no funcionales	
Condición	El sistema debe tener una conexión activa con el servidor de bases de datos.
Accionamiento	El usuario debe hace clic sobre la opción “Desconectar” en el menú respectivo.
Flujo de eventos	1. El sistema limpia todas las listas de bases de datos. 2. El sistema muestra las listas vacías. 3. El sistema cierra la conexión
Flujo alternativo	Si el sistema está desconectado del servidor de bases de datos, le informará al usuario mediante un mensaje de error.
Poscondición	Se ha cerrado la conexión con el servidor de bases de datos.
Comentarios	Este caso de uso se ejecuta de manera automática al cerrar la aplicación.

Tabla 5.7. *Caso de uso E4.*

Identificación y nombre del caso de uso	A1 – Agregar información adicional al reporte.
Descripción	Este procedimiento se encarga de insertar información adicional en el reporte. Esta información adicional es suministrada por el usuario.
Actores	Usuario, sistema.
Prioridad	Alta.
Requerimientos no funcionales	
Condición	Ninguna.
Accionamiento	El usuario debe hacer clic sobre la pestaña “Información adicional”.
Flujo de eventos	• El usuario escribe un título para el reporte en la caja de texto diseñada para tal fin. • El usuario hace clic sobre el botón “Buscar”, si desea insertar una imagen. Si desea seguir este paso, ocurren los siguientes pasos: 1. El sistema muestra un cuadro de diálogo para seleccionar la imagen a insertar. 2. El usuario busca y selecciona la imagen deseada. 3. El sistema cierra el cuadro de diálogo y escribe la ruta a la imagen en una caja de texto. • El usuario escribe comentarios sobre el reporte. • El sistema inserta la fecha de creación del reporte como información adicional.
Flujo alternativo	
Poscondición	Se ha escrito la información adicional del reporte, que será exportada junto al reporte.
Comentarios	Excepto la inserción de la fecha del reporte, todos los pasos descritos son opcionales. La ejecución de este caso de uso queda a discreción del usuario.

Tabla 5.8. *Caso de uso A1.*

Identificación y nombre del caso de uso	A2 – Seleccionar plantilla de exportación.
Descripción	Este procedimiento permite al usuario escoger una plantilla que permita darle un formato de presentación a los reportes exportados a HTML.
Actores	Usuario, sistema.
Prioridad	Alta.
Requerimientos no funcionales	
Condición	Ninguna.
Accionamiento	El usuario debe hacer clic sobre la opción “Seleccionar plantilla” en el menú de plantillas.
Flujo de eventos	1. El sistema muestra un cuadro de diálogo, preguntando al usuario la plantilla a seleccionar. 2. El usuario selecciona la plantilla deseada. 3. El sistema cierra el cuadro de diálogo y guarda la ruta de la plantilla seleccionada.
Flujo alternativo	
Poscondición	Se ha seleccionado una plantilla de exportación.
Comentarios	Este caso de uso sólo es relevante si el usuario desea hacer una exportación a HTML. Si el usuario hace una exportación a XML, el sistema no utilizará la plantilla.

Tabla 5.9. *Caso de uso A2.*

Identificación y nombre del caso de uso	E5-01 – Exportar reporte a XML.
Descripción	Este procedimiento se encarga de exportar la información que haya seleccionado el usuario al formato XML.
Actores	Usuario, sistema, servidor de bases de datos.
Prioridad	Esencial.
Requerimientos no funcionales	
Condición	El sistema debe estar conectado a un servidor de bases de datos. El usuario debe haber escogido por lo menos un campo para exportación.
Accionamiento	El usuario debe hacer clic sobre la opción “XML” en el menú de exportación.
Flujo de eventos	1. El sistema muestra un cuadro de diálogo, para que el usuario escoja el lugar donde va a guardar el reporte. 2. El usuario escribe el nombre de archivo que desee y hace clic sobre el botón “Guardar”. 3. El sistema crea un nuevo archivo en blanco para el reporte. 4. El sistema inserta la información adicional que el usuario haya ingresado. 5. El sistema hace una consulta al servidor de bases de datos, pidiendo el contenido especificado por el usuario. 6. El servidor envía al sistema la información solicitada. 7. El sistema inserta en el archivo la información recibida del servidor de bases de datos. 8. El sistema cierra el archivo.
Flujo alternativo	Si por algún motivo el archivo de destino no se puede crear (falta de permisos, disco lleno), el sistema informará mediante un mensaje de error y la exportación no se realizará.
Poscondición	Se ha exportado un reporte a XML.
Comentarios	Si el archivo de destino existe, el sistema lo sobreescribirá.

Tabla 5.10. *Caso de uso E5-01.*

Identificación y nombre del caso de uso	E5-02 – Exportar reporte a HTML.
Descripción	Este procedimiento se encarga de exportar la información que haya seleccionado el usuario al formato HTML.
Actores	Usuario, sistema.
Prioridad	Esencial.
Requerimientos no funcionales	
Condición	El sistema debe estar conectado a un servidor de bases de datos. El usuario debe haber escogido por lo menos un campo para exportación. El usuario debe haber seleccionado una plantilla de exportación.
Accionamiento	El usuario debe hacer clic sobre la opción “HTML” en el menú de exportación.
Flujo de eventos	1. El sistema muestra un cuadro de diálogo, para que el usuario escoja el lugar donde va a guardar el reporte. 2. El usuario escribe el nombre de archivo que desee y hace clic sobre el botón “Guardar”. 3. El sistema ejecuta el caso de uso E5-01 (exportación a XML) para crear un archivo de entrada temporal. 4. El sistema lee la plantilla que el usuario haya especificado. 5. El sistema aplica un proceso de transformación al archivo de entrada usando la plantilla seleccionada. 6. El sistema guarda el archivo de salida con el nombre escogido por el usuario. 7. El sistema cierra el archivo de salida y elimina el temporal.
Flujo alternativo	Si por algún motivo el archivo de destino no se puede crear (falta de permisos, disco lleno), el sistema informará mediante un mensaje de error y la exportación no se realizará.
Poscondición	Se ha exportado un reporte a HTML.
Comentarios	Si el archivo de destino existe, el sistema lo sobreescribirá.

Tabla 5.11. *Caso de uso E5-02.*

5.2.3 DIAGRAMAS

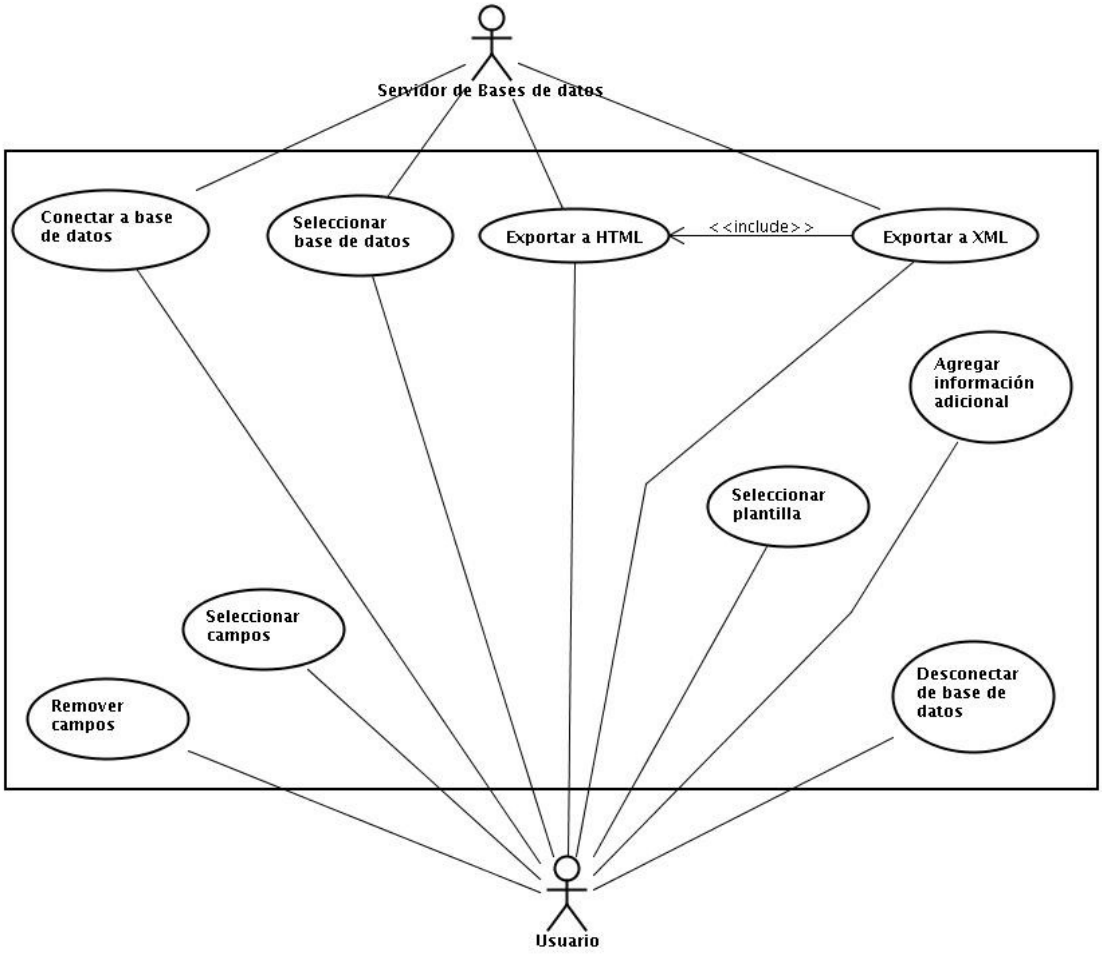

Imagen 5.1. *Diagrama de casos de uso.*

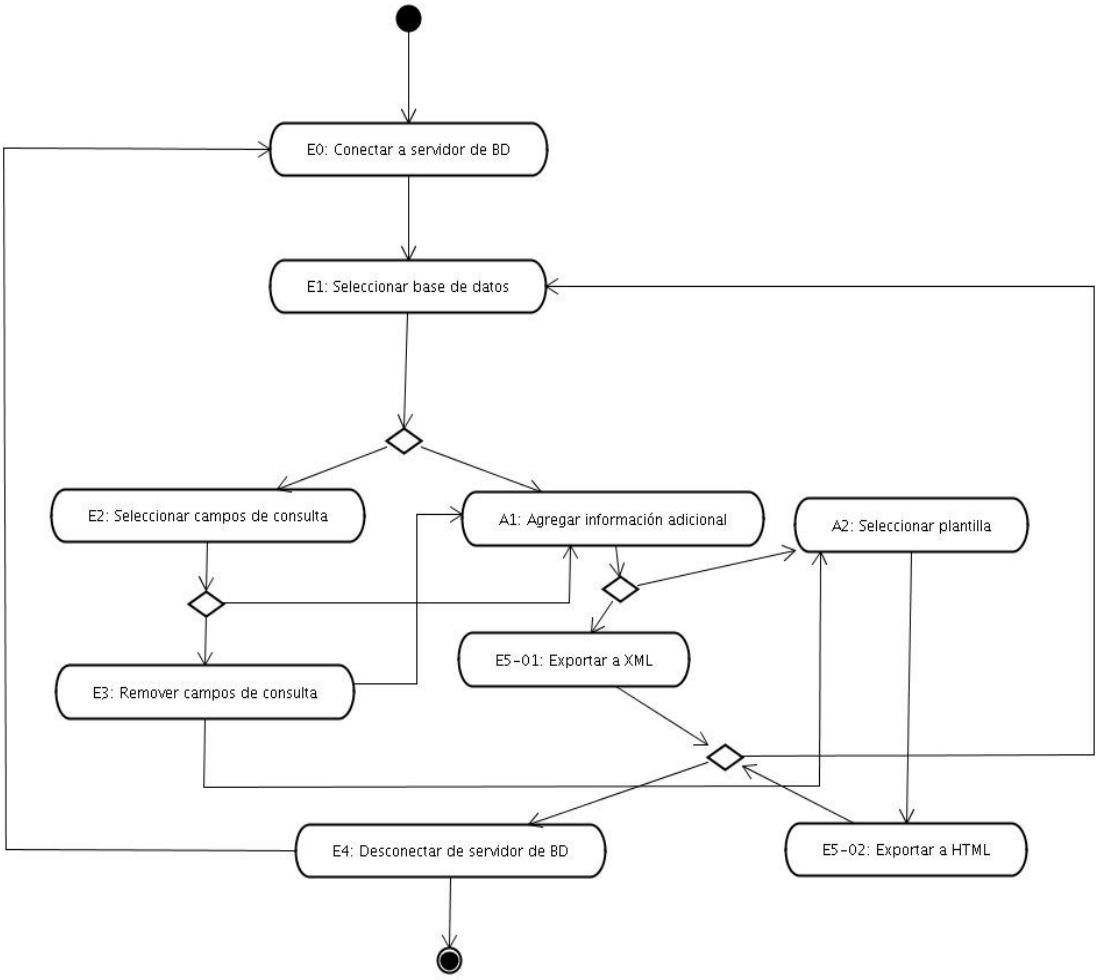

Imagen 5.2. *Diagrama de actividades.*

5.3 DISEÑO

El diseño es la etapa del proceso de la ingeniería de software que antecede a la implementación. Estas actividades constituyen la construcción del sistema. El diseño se subdivide en diseño del sistema y diseño de objetos[2]. Esta etapa del proceso de la ingeniería de software suele documentarse con diagramas UML, con el fin de mostrar de forma exacta la estructura del sistema.

El diseño del sistema es el primer paso en esta etapa, y se enfoca en la descomposición del sistema en partes manejables. Durante el diseño del sistema nos enfocamos en los procesos, estructuras de datos y componentes de software y hardware necesarios para implementarlo. El reto del diseño del sistema es que hay que satisfacer muchos criterios y restricciones conflictivos cuando se descompone el sistema[3].

Durante el diseño de objetos se refinan los modelos de análisis y de diseño del sistema, y se identifican nuevos objetos. Para esto se identifican objetos personalizados, se ajustan los componentes hechos y se hace una especificación precisa de cada interfaz de subsistema y clase. Como resultado, el modelo de diseño de objetos puede dividirse en conjuntos de clases que pueden ser implementados por desarrolladores individuales[4].

2. Bruegge, Bernd y Dutoit, Allen H. Diseño del sistema. En: Ingeniería de software orientado a objetos, 1ª ed. p. 168.
3.Ibid, pag. 168.
4. Op. cit., p. 233.

5.3.1 DISEÑO DEL PROTOTIPO

Al llegar a la etapa de diseño, se decidió dividir el sistema en tres módulos principales:

- Un módulo que se encarga de interactuar con la base de datos.
- Un módulo encargado de hacer las exportaciones.
- Un módulo que solamente agrupe de las GUI (interfaces gráficas de usuario).

Los módulos fueron creados con base en la funcionalidad que ofrecen, con el fin de agrupar en ellos las clases que intervienen para obtener la funcionalidad que se desea, y que además puedan integrarse con facilidad a los demás módulos que conforman el sistema. Esta división en módulos también permite la reutilización de las clases que lo conforman para otros proyectos de software.

Aparte de los módulos principales, existen unas clases que se encargan de las transformaciones XSLT, que hacen parte de la JAXP y son implementadas por Xalan, el procesador de hojas de estilo XSLT usado para el desarrollo de este proyecto.

A continuación se muestran las clases que conforman cada módulo y la funcionalidad que aportan:

- Módulo de bases de datos: Está conformado únicamente por la clase *Database*, que se encarga de: cargar el driver de la base de datos, conectarse a la base de datos, desconectarse de la base de datos, enviar consultas al servidor de bases de datos y chequear el estado de la conexión de la base de datos. La clase *Database* implementa el patrón de diseño Singleton, lo que asegura que sólo exista un objeto de esta clase durante la ejecución del sistema.
- Módulo de exportación: Se compone de las clases *Report*, *XMLExporter* y *HTMLExporter*, y de la interfaz *Exporter*. La interfaz *Exporter* sólo provee el método export(), el cual se encarga de exportar el reporte actual al formato especificado por las clases que implementan esta interfaz. Las clases *XMLExporter* y *HTMLExporter* implementan la interfaz *Exporter*, y sobreescriben el método export() para exportar el reporte a XML o HTML, respectivamente. Ambas clases tienen como atributos un objeto de la clase *Report*, una cadena, que almacena el query que obtiene la información del servidor de bases de datos y una *Database*. La clase *HTMLExporter* incluye además un atributo *File* que guarda la información de la plantilla XSLT a utilizar. La clase *Report* sirve como un contenedor que almacena los datos adicionales de cada reporte. Estos datos son el título del reporte, los comentarios, la ruta a la

imagen (si el reporte usa una imagen) y la fecha de creación del reporte.

- Módulo de las interfaces gráficas de usuario: Se compone de las clases *MainWindow*, que como su nombre lo indica es la ventana principal con la que va a interactuar el usuario, la clase *ConnectDialog*, que es un diálogo donde se introducen los parámetros de conexión a la base de datos, y la clase *AboutBox*, la cual es simplemente un cuadro de diálogo con datos acerca del sistema.

En los siguientes diagramas se muestran las clases que conforman el sistema, los atributos y operaciones (métodos) que cada clase posee, las relaciones entre las clases y los módulos que conforman el sistema, vistos como paquetes.

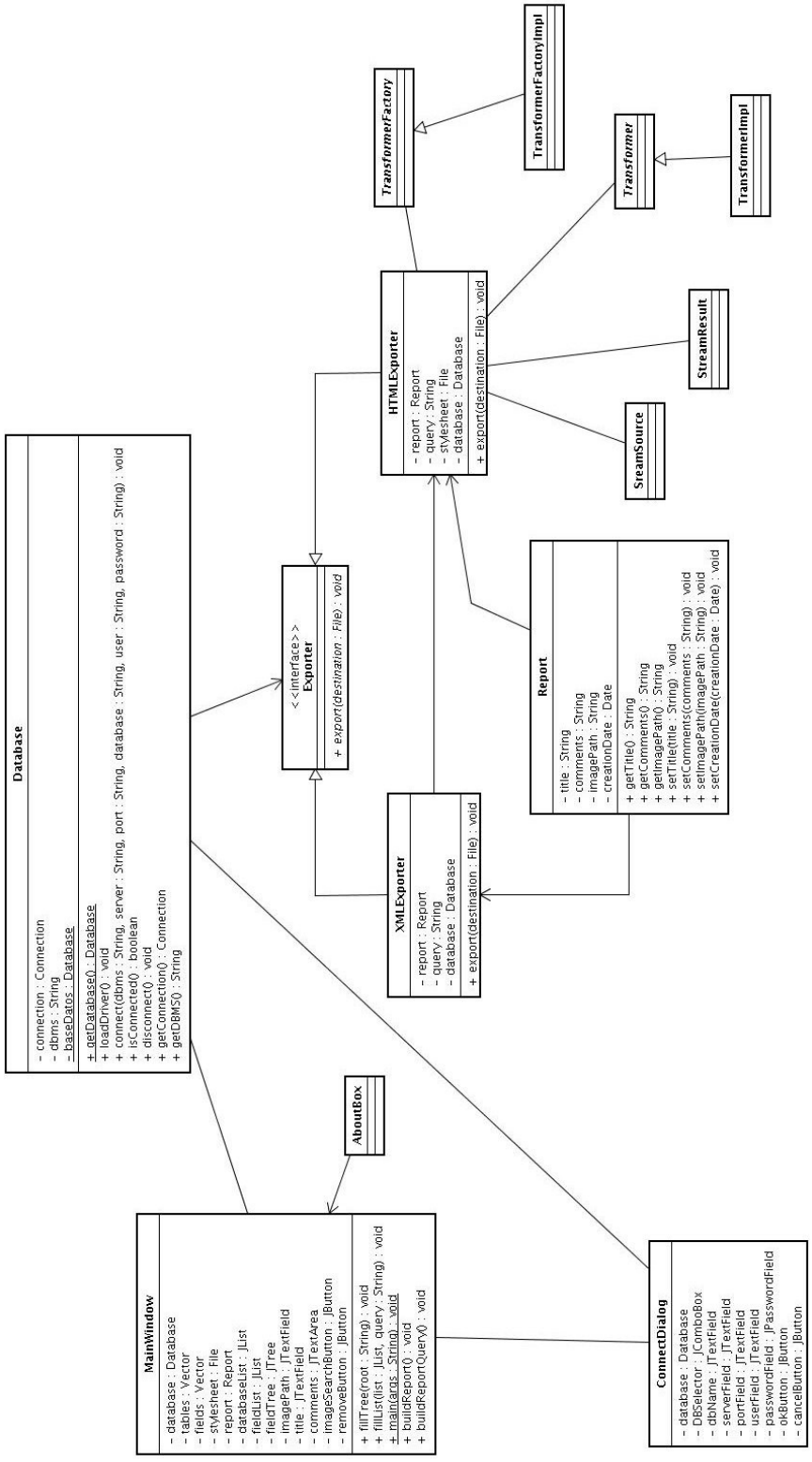

Imagen 5.3. *Diagrama de clases.*

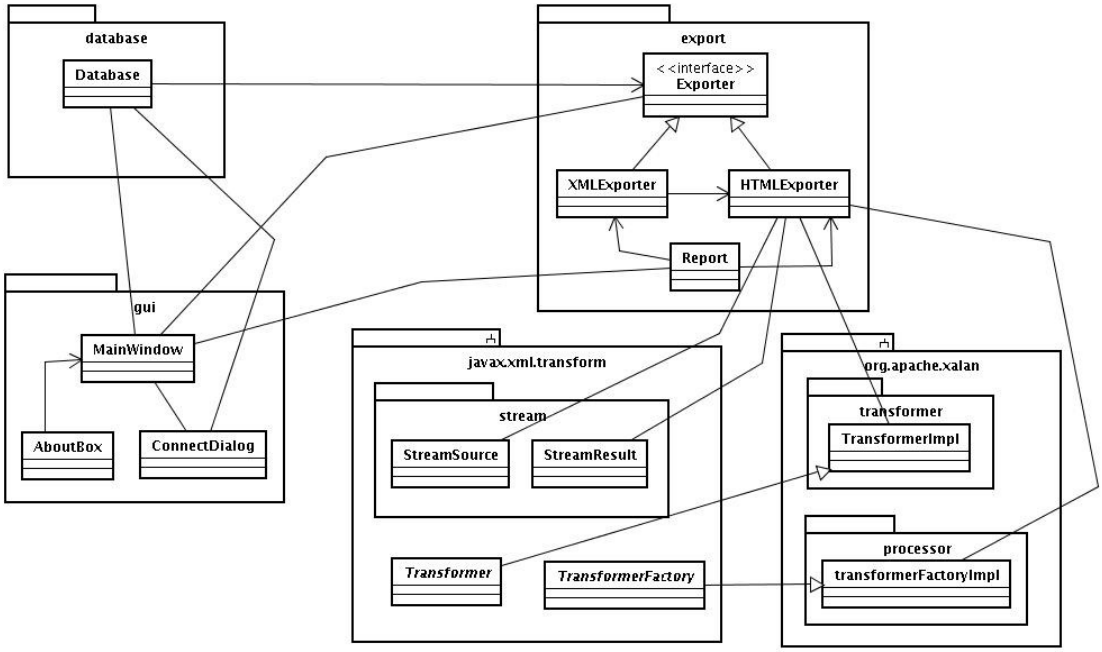

Imagen 5.4. *Diagrama de paquetes.*

Los diagramas de secuencia aquí mostrados ilustran los pasos que se siguen para la ejecución exitosa de un caso de uso, así como los objetos que intervienen en la ejecución del caso de uso. Quizás los objetos más importantes son los objetos de control, ya que se encargan de administrar la ejecución del resto del caso de uso. El nombre de estos objetos inicia en Control para mayor claridad, y en la mayoría de los casos el control es la ventana principal de la aplicación, o sea, un objeto de la clase *MainWindow*.

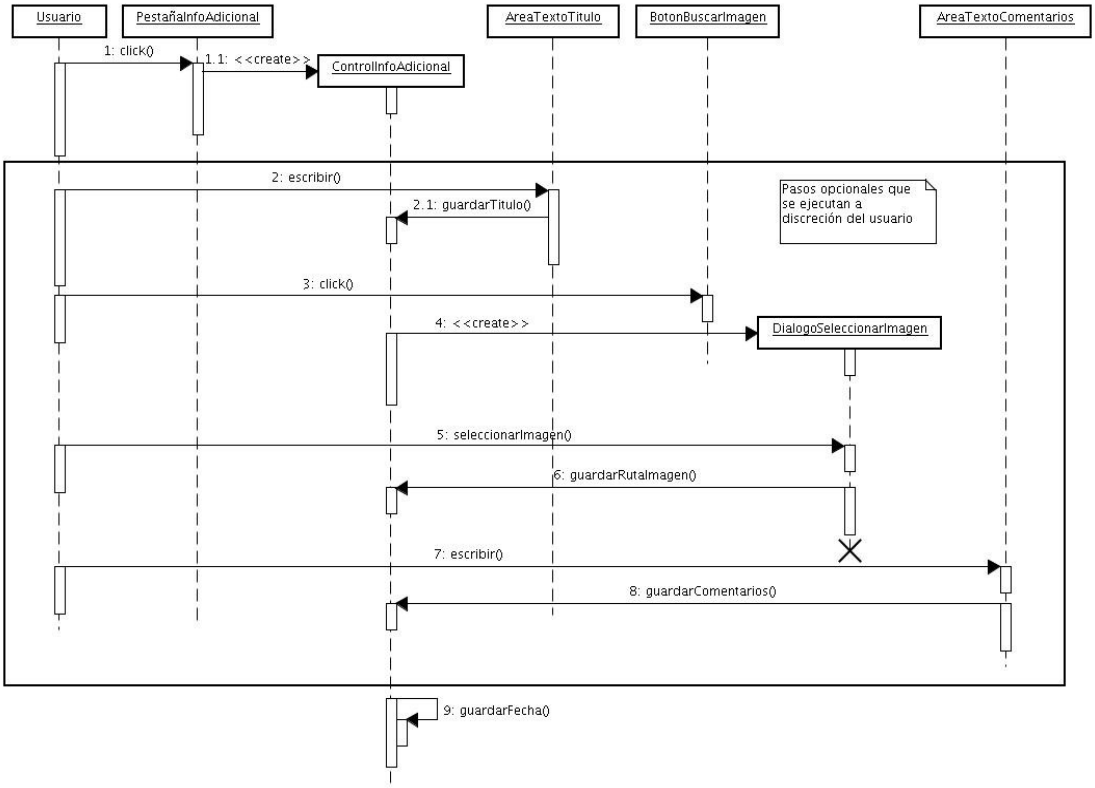

Imagen 5.5. *Diagrama de secuencia para el caso de uso A1.*

Para el caso de uso A1 – Agregar información adicional al reporte, el orden de ejecución es el siguiente: el usuario hace clic en una pestaña que permite el acceso a la información adicional, representada por el objeto

PestañaInfoAdicional, y se crea el objeto *ControlInfoAdicional*. Los pasos que siguen a este son totalmente opcionales, a discreción del usuario, y se pueden ejecutar en cualquier orden. El diagrama sugiere un orden, donde el usuario introduce título, ruta de imagen y comentarios en ese orden estricto. Primero, el usuario escribe el título del reporte en un *AreaTextoTitulo*, y se guarda en el *ControlInfoAdicional*. Igualmente ocurre con los comentarios, con la diferencia que el usuario escribe el comentario en un *AreaTextoComentarios*. Para la selección de una imagen, el usuario hace clic en un *BotonSeleccionarImagen*, y el *ControlInfoAdicional* crea un *DialogoSeleccionarImagen*, donde el usuario selecciona la imagen deseada, para finalmente guardar la ruta de la imagen en el *ControlInfoAdicional*. El último paso, donde se guarda la fecha de creación del reporte es obligatorio y se ejecuta automáticamente, sin interactuar con el usuario.

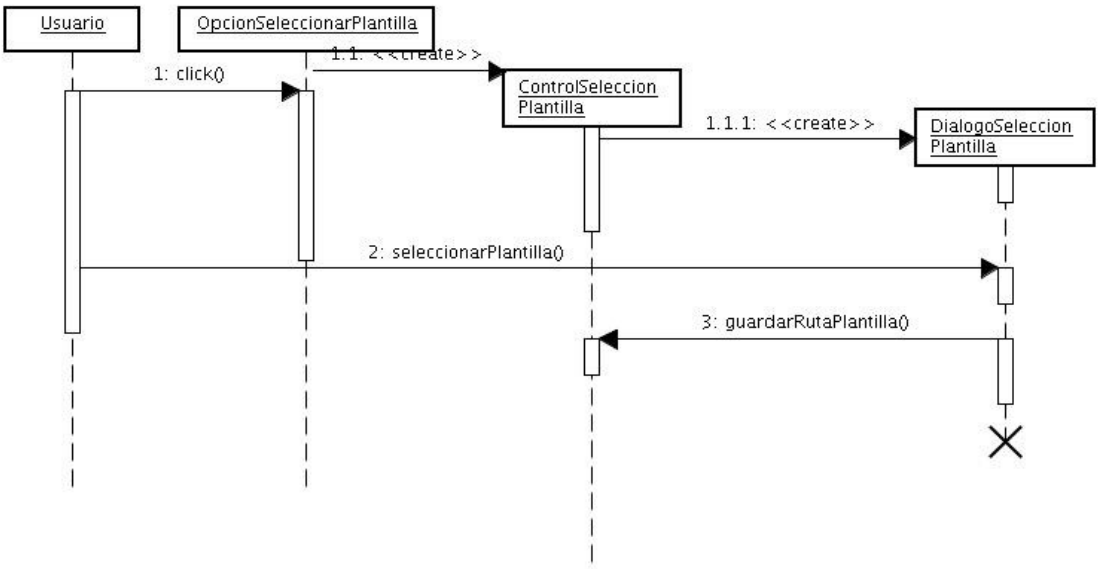

Imagen 5.6. *Diagrama de secuencia para el caso de uso A2.*

En el siguiente diagrama, que muestra el caso de uso A2 – Seleccionar plantilla de exportación, se muestra cómo el usuario inicia el caso de uso haciendo clic en un objeto *OpciónSeleccionarPlantilla*. A continuación, se crea un objeto *ControlSeleccionPlantilla*, que crea un *DialogoSeleccionPlantilla*, que como su nombre lo indica, es un cuadro de diálogo que permite seleccionar una plantilla de exportación. El usuario selecciona la plantilla deseada en el cuadro de diálogo, y el *ControlSeleccionPlantilla* guarda la ruta a la plantilla seleccionada.

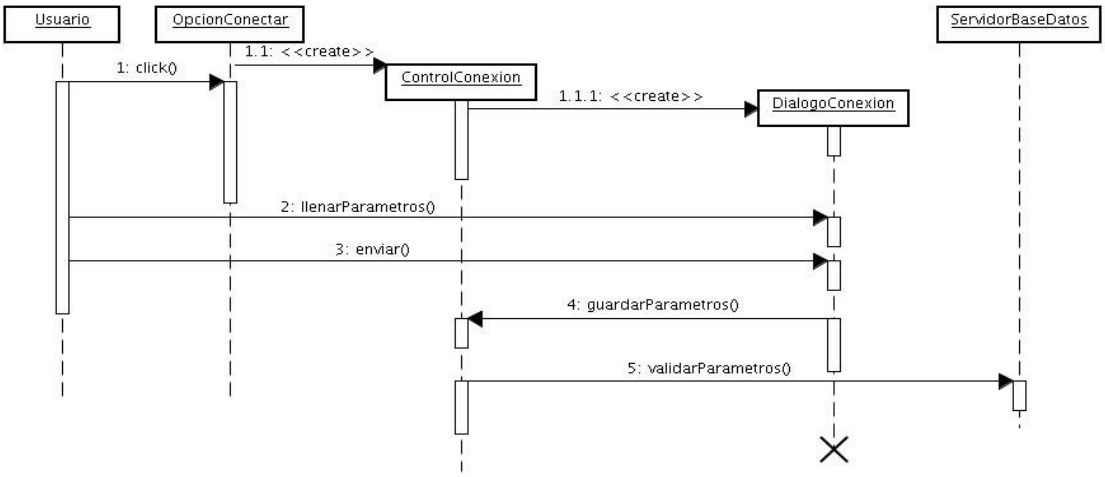

Imagen 5.7. *Diagrama de secuencia para el caso de uso E0.*

El caso de uso E0 – Conectar a un servidor de bases de datos se ejecuta de la siguiente manera: el usuario inicia el caso de uso haciendo clic en un objeto *OpcionConectar*, que hace parte de la ventana principal. Entonces se crea un objeto *ControlConexion*, el cual crea un objeto *DialogoConexion*, el cual es un objeto de la clase *ConnectDialog*. El usuario introduce los parámetros de la conexión que desea realizar (motor de bases de datos, servidor, puerto, base de datos, usuario y contraseña). Cuando el usuario haya finalizado, envía los datos, que se almacenan en el *ControlConexion*, y éste verifica los parámetros con el *ServidorBaseDatos*. Si los datos son válidos, el *DialogoConexion* se destruye, finalizando el caso de uso. Estos pasos a seguir se ilustran en el siguiente diagrama:

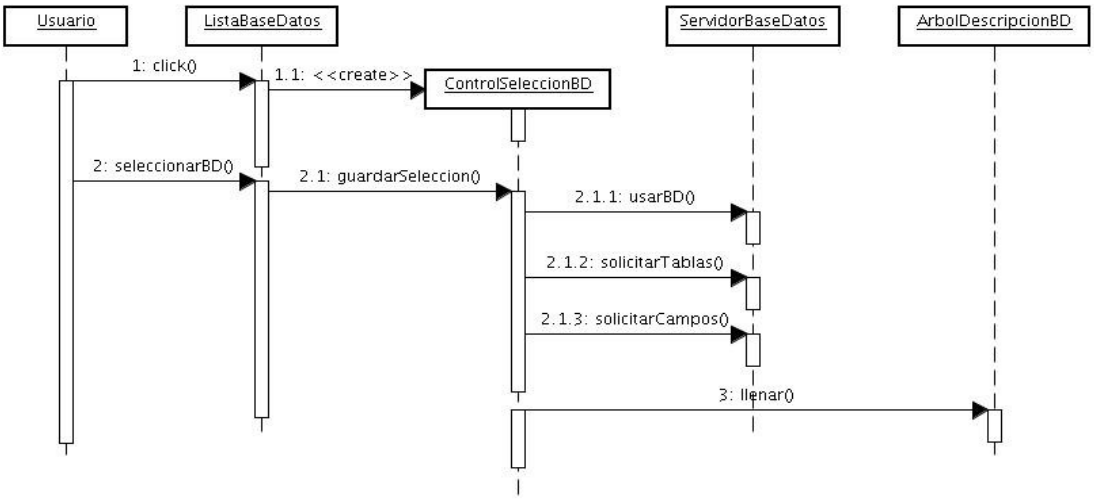

Imagen 5.8. *Diagrama de secuencia para el caso de uso E1.*

En el diagrama que se muestra a continuación, que ilustra el caso de uso E1 – Seleccionar una base de datos, se inicia el caso de uso cuando el usuario hace clic en un objeto *ListaBaseDatos*, el cual muestra en la ventana principal una lista de las bases de datos disponibles para el usuario. Al hacer clic, se crea un *ControlSeleccionBD*. A continuación, el usuario selecciona de la *ListaBaseDatos* una de las bases de datos que tiene disponibles. La opción seleccionada se guarda en el *ControlSeleccionBD*, el cual hace una petición al *ServidorBaseDatos* para usar la base de datos seleccionada por el usuario, seguida por dos tipos de peticiones, que solicitan las tablas que conforman la base de datos, así como los campos que conforman cada tabla de la base de datos. Finalmente, el *ControlSeleccionBD* le envía la información a un objeto *ArbolDescripcionBD*, el cual muestra en la ventana principal un árbol que contiene la estructura de la base de datos seleccionada.

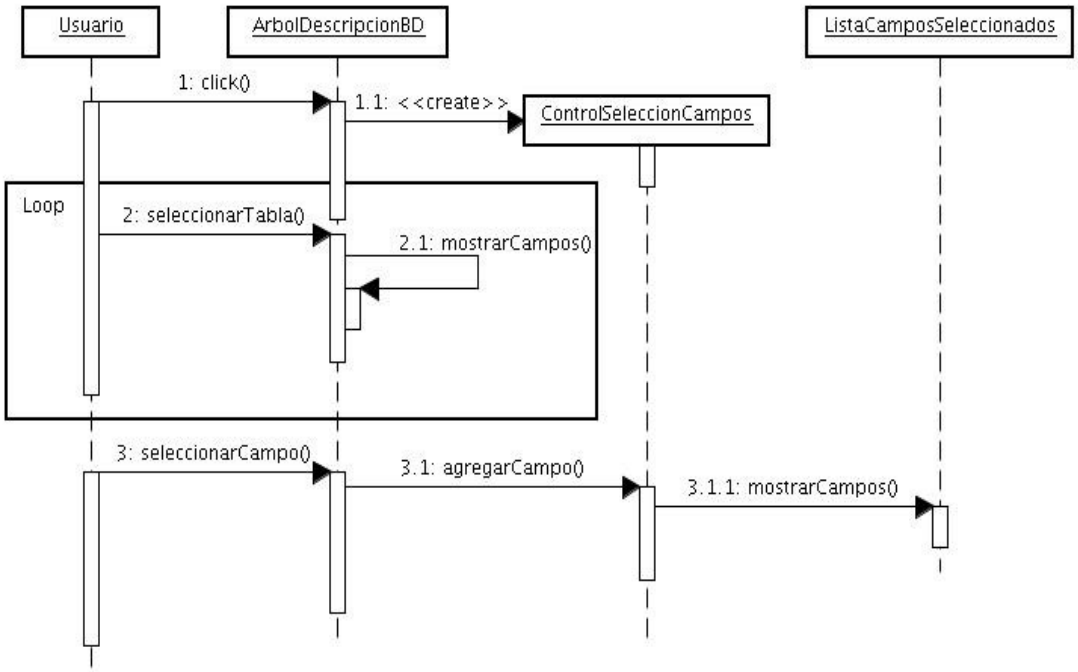

Imagen 5.9. *Diagrama de secuencia para el caso de uso E2.*

Para el caso de uso E2 – Seleccionar campos para consulta, el orden de ejecución es el que se muestra a continuación: para dar inicio al caso de uso, el usuario hace clic en un *ArbolDescripcionBD* y posteriormente se crea un objeto *ControlSeleccionCampos*. A continuación, el usuario selecciona una tabla del árbol, y automáticamente se muestran los campos que conforman esa tabla. Este paso se repite cada vez que el usuario selecciona una tabla distinta. Cuando el usuario ha hecho una elección, procede a seleccionar en el árbol un campo perteneciente a la tabla, y este campo se guarda en el *ControlSeleccionCampos*. Para finalizar, el *ControlSeleccionCampos* envía el campo seleccionado a un objeto *ListaCamposSeleccionados*, el cual es una lista que se muestra en la ventana principal, y como su nombre lo indica, contiene los campos que el usuario haya seleccionado.

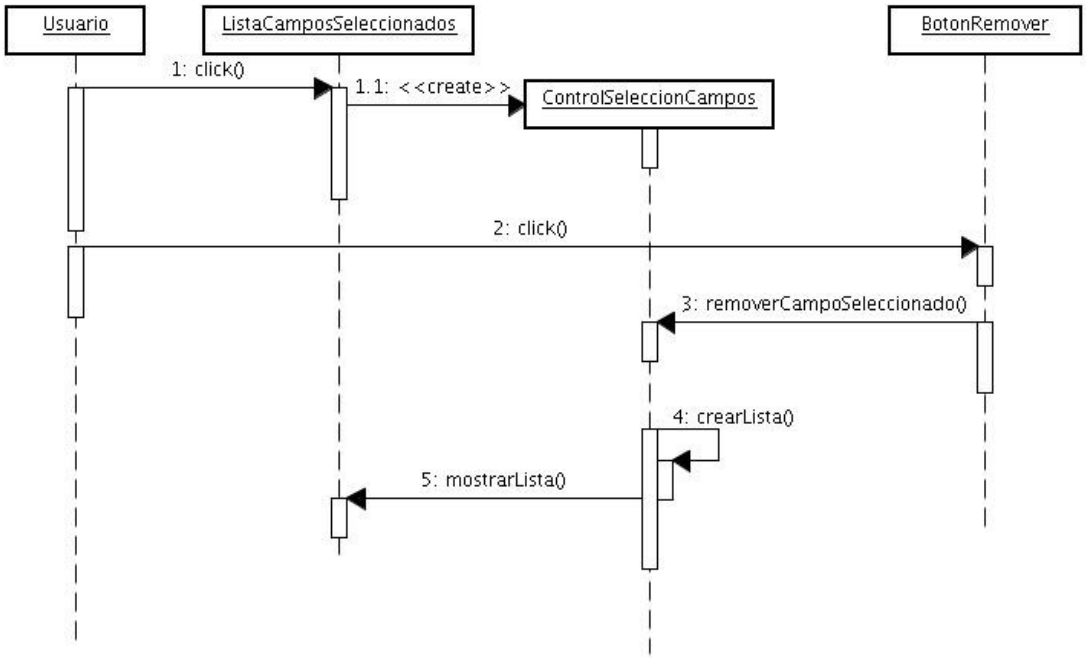

Imagen 5.10. *Diagrama de secuencia para el caso de uso E3.*

Para el siguiente diagrama, que ilustra el caso de uso E3 – Remover campos para consulta, se muestra cómo el usuario inicia el caso de uso haciendo clic en un objeto *ListaCamposSeleccionados*, específicamente en el campo que desea remover. Inmediatamente se crea un objeto *ControlSeleccionCampos*. A continuación, el usuario hace clic en un objeto *BotonRemover*, que hace parte de la ventana principal y que como su nombre lo indica, tiene la función de remover el campo seleccionado por el usuario. El botón envía un mensaje al *ControlSeleccionCampos* para que remueva el campo seleccionado. El *ControlSeleccionCampos* vuelve a crear la lista de campos, y se la envía al objeto *ListaCamposSeleccionados* para que éste la muestre.

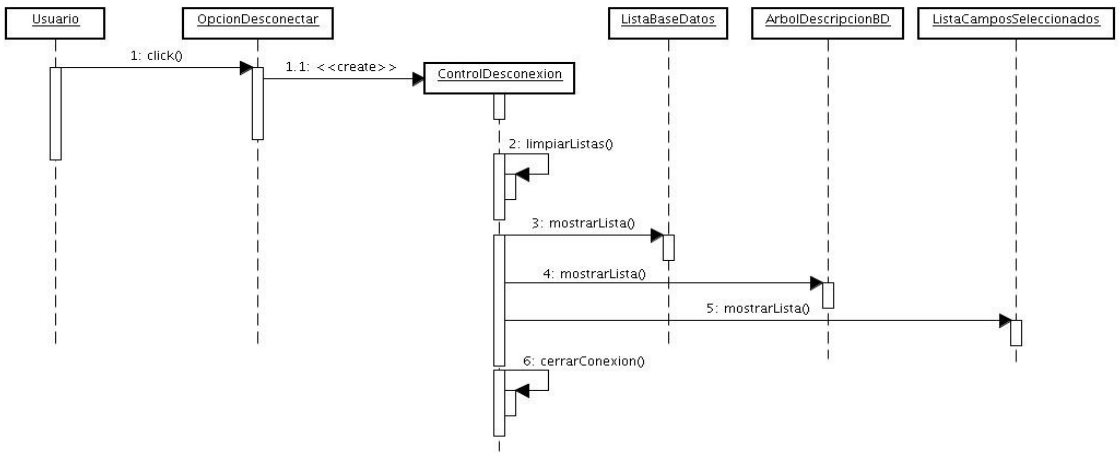

Imagen 5.11. *Diagrama de secuencia para el caso de uso E4.*

En el diagrama que se muestra a continuación, que ilustra el caso de uso E4 – Desconectar del servidor de bases de datos, se inicia el caso de uso cuando el usuario hace clic sobre un objeto *OpcionDesconectar*, que hace parte de la ventana principal, y a continuación se crea un objeto *ControlDesconexión*. El *ControlDesconexión* procede a limipar las listas de bases de datos, y envía las listas vacías a los objetos *ListaBaseDatos*, *ArbolDescripciónBD* y *ListaCamposSeleccionados* para que muestre al usuario las listas vacías, y finalmente cierra la conexión con el servidor de bases de datos.

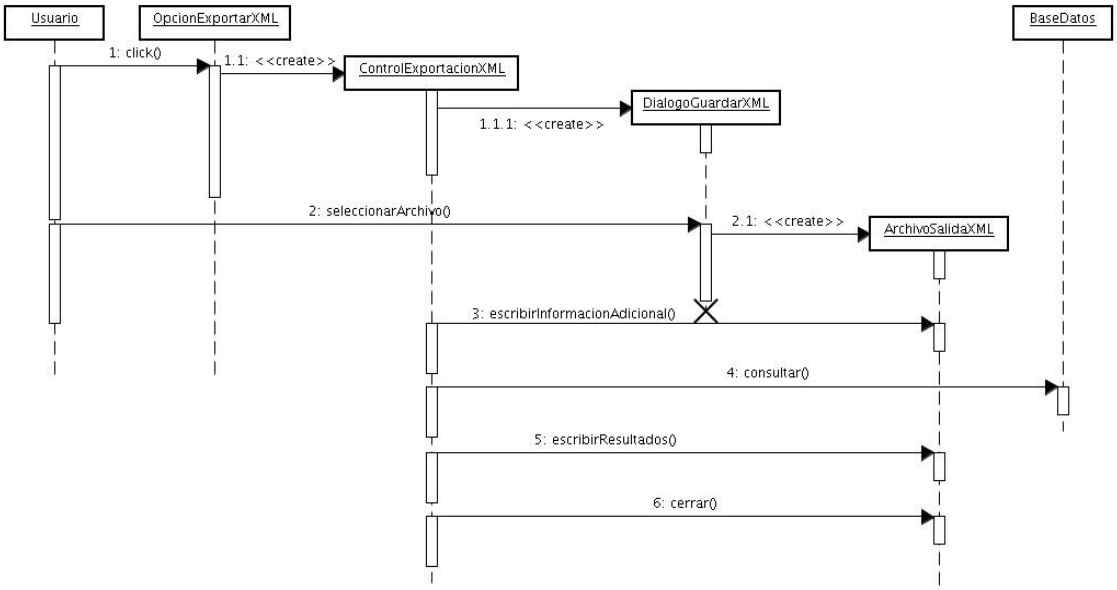

Imagen 5.12. *Diagrama de secuencia para el caso de uso E5-01.*

El caso de uso E5-01 – Exportar reporte a XML se ejecuta como se muestra a continuación: se inicia el caso de uso cuando el usuario hace clic en un objeto *OpcionExportarXML*, que se incluye en la ventana principal. Entonces se crea un objeto *ControlExportacionXML*, el cual crea un objeto *DialogoGuardarXML*, que es un cuadro de diálogo que permite guardar un archivo XML. El usuario ingresa el nombre de archivo que desee y confirma la operación. El *DialogoGuardarXML* crea un objeto *ArchivoSalidaXML*, el cual es el archivo destino donde se hará la exportación. El *ControlExportacionXML* escribe la información adicional del reporte en el *ArchivoSalidaXML*, después consulta a la base de datos y escribe los resultados de la consulta en el *ArchivoSalidaXML*, y finaliza cerrando el archivo.

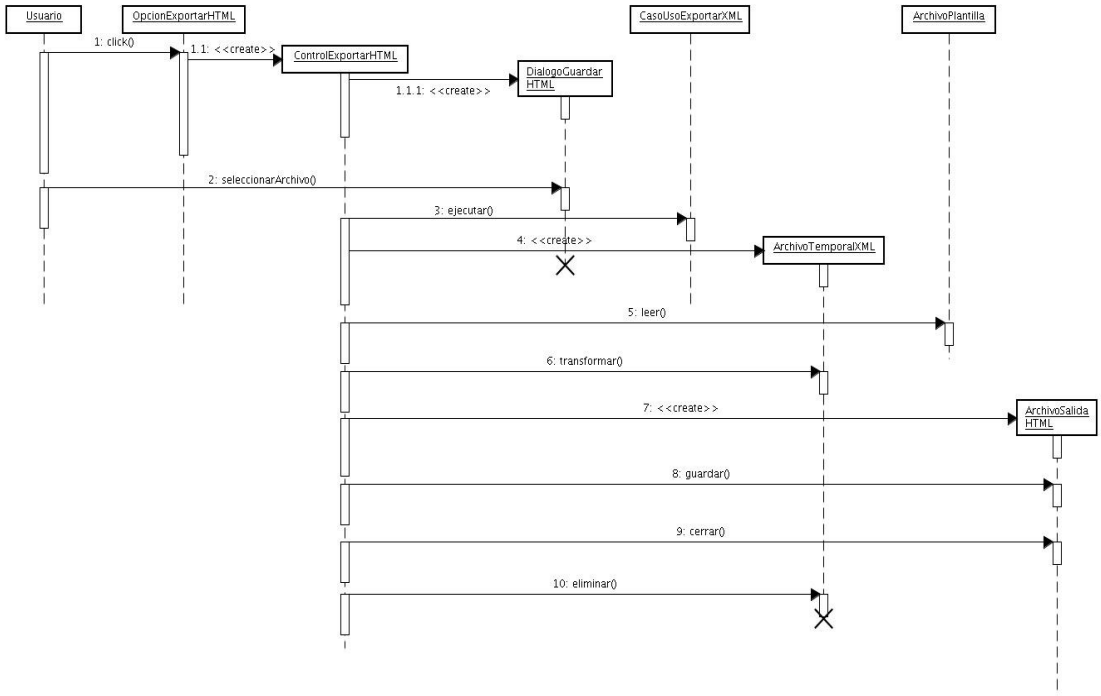

Imagen 5.13. *Diagrama de secuencia para el caso de uso E5-02.*

El caso de uso E5-02 – Exportar reporte a HTML se ejecuta de la siguiente manera: el usuario hace clic en un objeto *OpcionExportarHTML* para dar inicio al caso de uso, y se crea el objeto *ControlExportarHTML*.

El *ControlExportarHTML* crea un objeto *DialogoGuardarHTML*, el cual es un cuadro de diálogo que permite guardar archivos HTML. El usuario selecciona el archivo donde desea guardar el reporte y confirma la operación. El *ControlExportarHTML* ejecuta el caso de uso E5-01, y guarda el resultado en un *ArchivoTemporalXML*, el cual como su nombre lo indica es un reporte temporal exportado a XML.

El *ControlExportarHTML* lee un *ArchivoPlantilla*, que es la plantilla XSLT que el usuario haya especificado con anterioridad, y con la información obtenida transforma el *ArchivoTemporalXML*. El *ControlExportarHTML* crea un *ArchivoSalidaHTML*, que es el archivo destino donde se hará la exportación final. El *ControlExportarHTML* guarda en este archivo los resultados de la transformación que se hizo sobre el documento XML y cierra el archivo. Finalmente, el *ControlExportarHTML* elimina el archivo temporal.

6 CONCLUSIONES

Al concluir este estudio monográfico, se puede apreciar la importancia de los generadores de reportes tanto en el ámbito empresarial como para los desarrolladores de software, notando que el uso de estas herramientas está aumentando de una forma rápida y progresiva, debido a que a la hora de construir reportes, éstas permiten que el reporte se realice de una forma más ágil y productiva, ahorrándole tiempo y esfuerzo al usuario final.

Entre los aportes que brinda esta monografía se encuentran un estudio de los generadores de reportes en la actualidad, donde se explica qué es un generador de reportes, los beneficios que éstos proveen y un análisis de algunos de los generadores de reportes más importantes vigentes en el mercado actual.

Otro aporte realizado por esta monografía es la ilustración del proceso de desarrollo de un generador de reportes. Para esto se siguió el procedimiento de análisis y diseño de software, usando una mezcla del proceso de Programación Extrema (XP) junto con el Proceso Ágil Unificado (AUP), utilizando casos de uso y diagramas UML para ilustrar la estructura y la funcionalidad de un generador de reportes. A continuación se implementó un prototipo de generador de reportes usando una serie de tecnologías que fueron investigadas durante el proceso de realización del trabajo monográfico y que además cumpliera con las especificaciones de análisis y diseño hechas

previamente. Para terminar, se escribió un manual de usuario para el prototipo desarrollado, con el fin de que éste tenga una referencia acerca de la forma en que funciona el prototipo.

Esta monografía, a nivel tecnológico, introduce al lector en el uso de la tecnología Java, en los lenguajes XML / XSL para exportar y transformar información; y en el lenguaje HTML para mostrar información en un navegador web, haciendo uso de las APIs de Java para XML y la implementación de éstas hecha por la Apache Software Foundation.

Esta monografía contiene algunas recomendaciones que futuros desarrolladores pueden tomar en cuenta para extender el trabajo hecho durante el desarrollo de esta monografía, como pueden ser el uso de algunas herramientas de desarrollo o posibles características a incluir en el prototipo.

En general, este trabajo recopila y pone en práctica los conocimientos adquiridos durante el Minor de Ingeniería de Software y durante toda la carrera de Ingeniería de Sistemas.

7 RECOMENDACIONES

Es indispensable contar con herramientas CASE de diseño y desarrollo actualizadas para llevar a cabo la extensión de este proyecto. Debido a que este proyecto fue desarrollado por estudiantes para estudiantes, se recomienda encarecidamente el uso de software libre.

Para el desarrollo de este proyecto, se usó JUDE como herramienta de diseño de diagramas UML, ya que permite crear los diagramas de una manera muy sencilla, y permite la exportación e importación de código Java.

Las herramientas de desarrollo utilizadas fueron la plataforma Eclipse y el IDE NetBeans. Ambos son los entornos de desarrollo en Java más populares en la actualidad, ya que permiten la refactorización, generan código y hacen sugerencias para el uso correcto del lenguaje Java. Se usaron ambos ya que, mientras Eclipse genera un código más fácil de leer, NetBeans permite diseñar las GUI más rápidamente y de forma sencilla.

Para asegurar la portabilidad de la aplicación, el desarrollo fue llevado a cabo por completo bajo el sistema operativo Linux, mientras que las pruebas fueron hechas en los sistemas Linux y Microsoft Windows.

No es obligatorio el uso de estas herramientas para llevar a cabo este proyecto, ya que cualquier herramienta CASE puede servir para el mismo propósito, pero es ampliamente sugerido el uso de estas herramientas ya que son libres, lo que implica el bajo (incluso nulo) costo de las herramientas y el fácil acceso a éstas por parte de los estudiantes, y porque son herramientas que se pueden aprender a utilizar fácilmente. Todas estas herramientas están incluidas en el CD-ROM que acompaña a esta monografía.

Para extender la funcionalidad del prototipo, se sugiere la implementación de las siguientes características:

- Exportación a otros formatos de salida, por ejemplo PDF, XLS (Microsoft Excel), ODS (OpenDocument Spreadsheet). Sería interesante exportar al formato ODS, ya que el formato OpenDocument (usado por varias aplicaciones ofimáticas, entre ellas la suite OpenOffice.org, de creciente popularidad) es un estándar ISO (norma ISO/EIC 26300).
- Permitir el diseño visual del reporte. Para esto, se puede utilizar la API Java2D.
- Mostrar una vista previa del reporte antes de ser exportado, a petición del usuario.
- Añadir soporte para otras bases de datos, por ejemplo PostgreSQL, DB2 y Oracle.

- Diseñar varias plantillas de reporte. Incluso, se podría hacer un módulo para la creación visual de plantillas, ya que actualmente hay que tener conocimientos de XSLT para crear plantillas de reporte.
- Dotar al prototipo con la capacidad de imprimir los reportes.
- Uso de archivos como origen de datos para los reportes. Se podría usar archivos XML o de otro formato.
- Creación de un formato específico para los reportes, con el fin de permitir guardar los reportes para modificarlos posteriormente.

8 REFERENCIAS BIBLIOGRÁFICAS

SOMMERVILLE, Ian. Ingeniería de Software, 6ª ed. Pearson Educación, México, 2002. p. 100, 101.

BRUEGGE, Bernd y DUTOIT, Allen H. Ingeniería de Software Orientado a Objetos, 1ª ed. Pearson Educación, México, 2002. p. 168, 233.

Sitios web:

http://en.wikipedia.org/wiki/Xml

http://en.wikipedia.org/wiki/XSLT

http://en.wikipedia.org/wiki/HTML

http://en.wikipedia.org/wiki/Xalan

http://en.wikipedia.org/wiki/Xerces

http://en.wikipedia.org/wiki/JAXP

http://es.wikipedia.org/wiki/XML

http://es.wikipedia.org/wiki/XSLT

http://es.wikipedia.org/wiki/HTML

http://xalan.apache.org/

http://xerces.apache.org/

http://www.w3schools.com/

http://www.bitam.com.mx/TecAnalisis.htm#reportools

ANEXOS

A. MANUAL DE USUARIO

1. REQUERIMIENTOS BÁSICOS

- 8 MB de espacio en disco.
- Máquina virtual J2SE, versión 5.0 instalada. Si no tiene instalada la máquina virtual, puede ir al sitio http://java.sun.com/j2se/1.5.0/download.jsp para descargar la versión correspondiente a su sistema operativo e instalarla siguiendo las instrucciones de instalación.
- Un servidor de bases de datos MySQL (puede ser local o remoto).

1. INSTALACIÓN

Para instalar el prototipo, extraiga el contenido del archivo zip en el directorio de su preferencia.

2. EJECUCIÓN

Para ejecutar el programa, haga doble clic en el archivo JCReports.jar (si es usuario de Windows) o en el archivo jcreports (si usa un sistema UNIX). Si por alguna razón no puede ejecutar estos archivos mediante el doble clic, puede abrir una consola de comandos y en ella navegar hasta la localización del archivo .jar y ejecutarlo mediante la orden `java –jar JCReports.jar`.

3. USO DEL PROGRAMA

Al ejecutar el programa, aparece el entorno principal, el cual se observa a continuación:

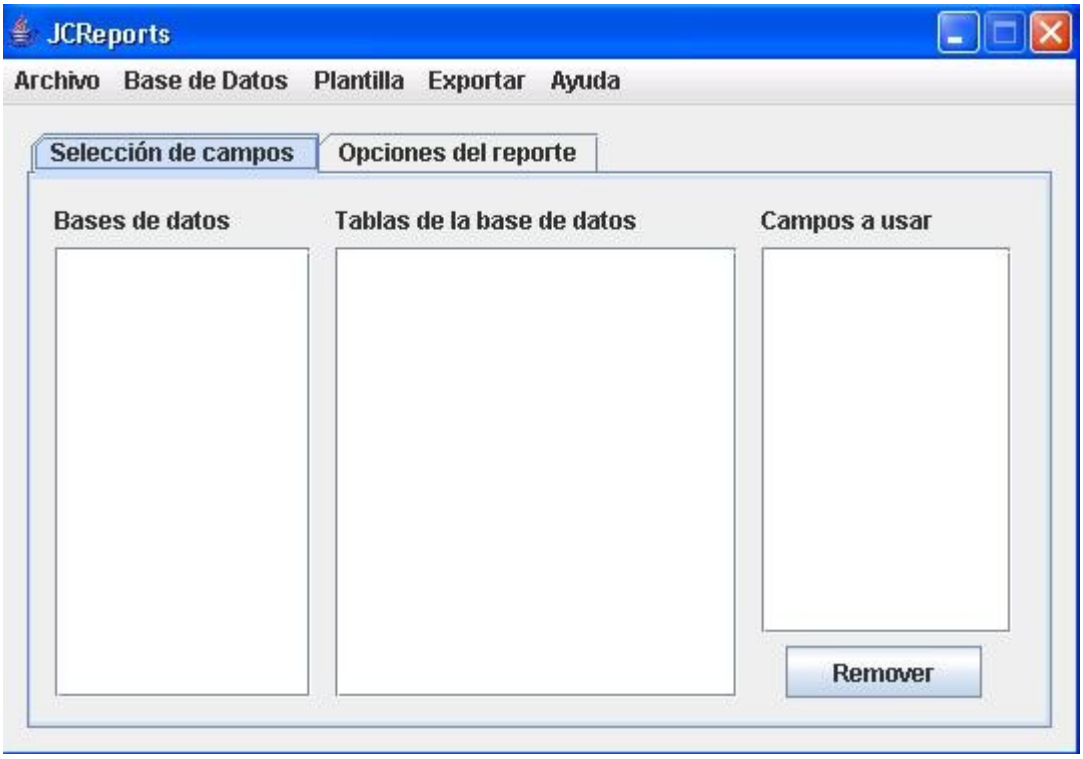

Imagen A.1. *Entorno principal del prototipo*

Para conectarse a una base de datos haga clic en la opción ***Conectar*** del menú ***Base de datos***, y aparecerá el siguiente cuadro de dialogo:

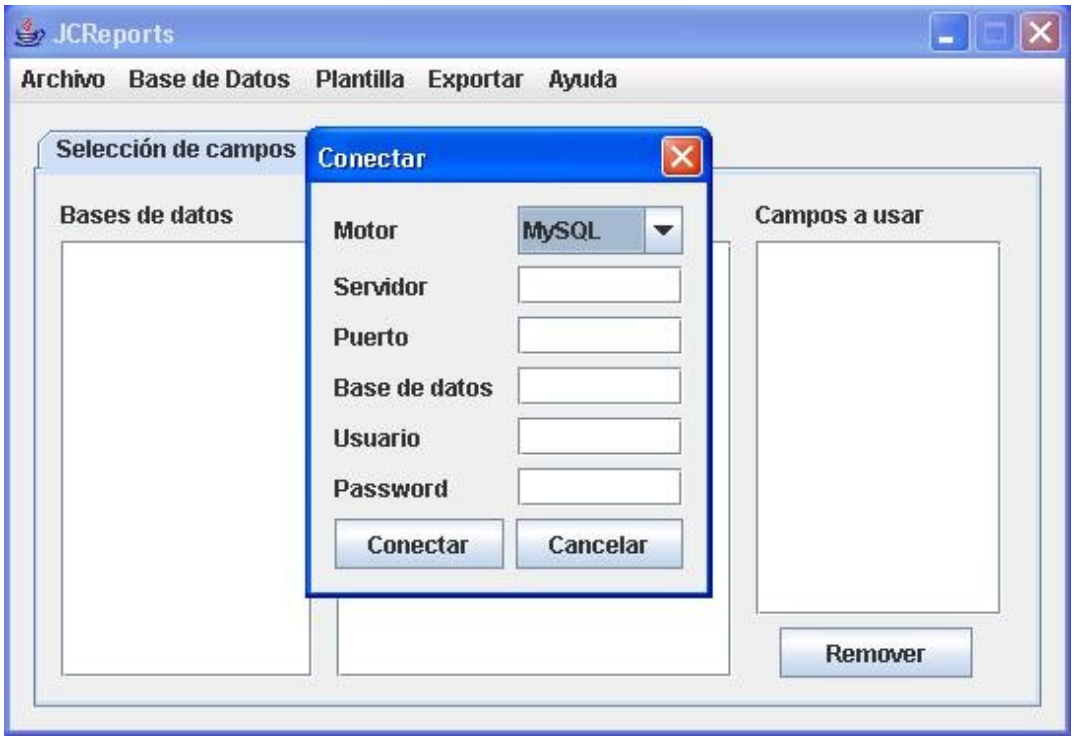

Imagen A.2. *Dialogo de conexión a una base de datos.*

A continuación introduzca los datos necesarios para establecer la conexión.

- En el campo ***Servidor*** escriba la dirección IP o el nombre del servidor donde está alojada la base de datos.
- En el campo ***Puerto*** escriba el número del puerto donde funciona el servidor de la base de datos (el valor por defecto es el 3306)
- En el campo ***Base de datos*** escriba el nombre de la base de datos a la que desea conectarse.
- En el campo ***Usuario*** escriba el nombre de un usuario valido de la base de datos y en el campo ***Password*** la respectiva contraseña.

Un ejemplo de cómo debe llenar este diálogo es el siguiente:

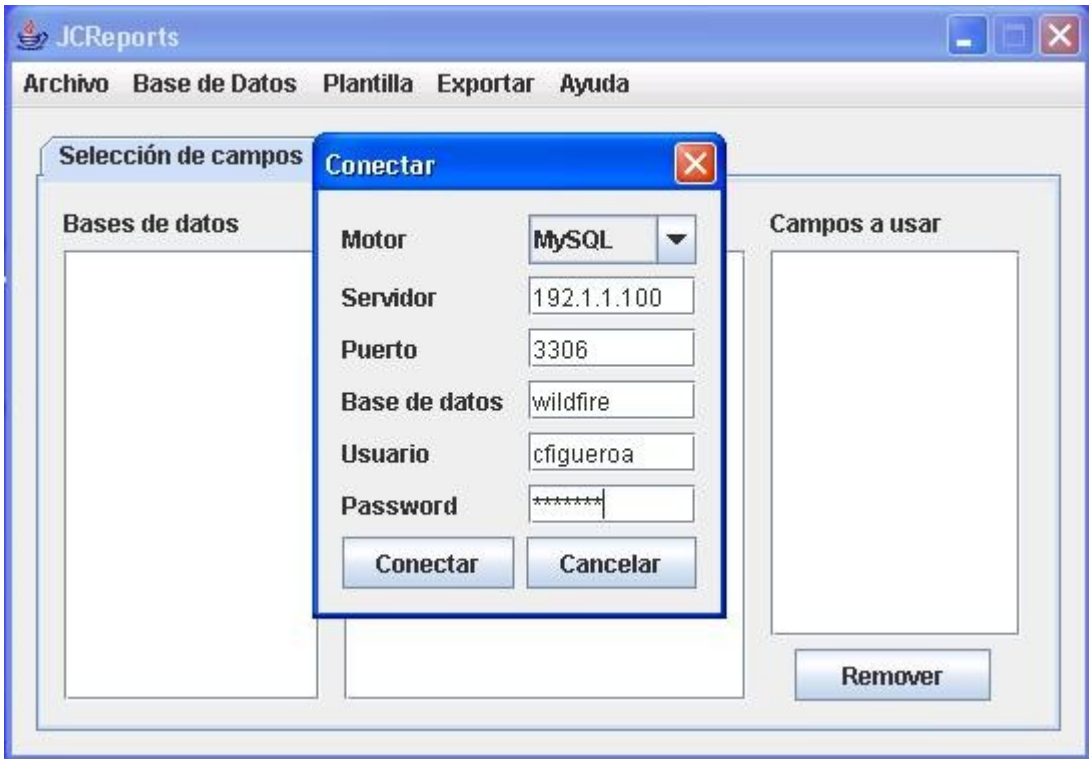

Imagen A.3 *Ejemplo de conexión a una base de datos.*

Finalmente haga clic en el botón ***Conectar***.

Al establecer la conexión con la base de datos en la lista ***Bases de datos*** aparecerán una lista con las bases de datos a las que usted tiene acceso.

Al seleccionar una base de datos aparecerán en la columna ***Tablas de la base de datos*** aparecerán las tablas que conforman la base de datos elegida anteriormente.

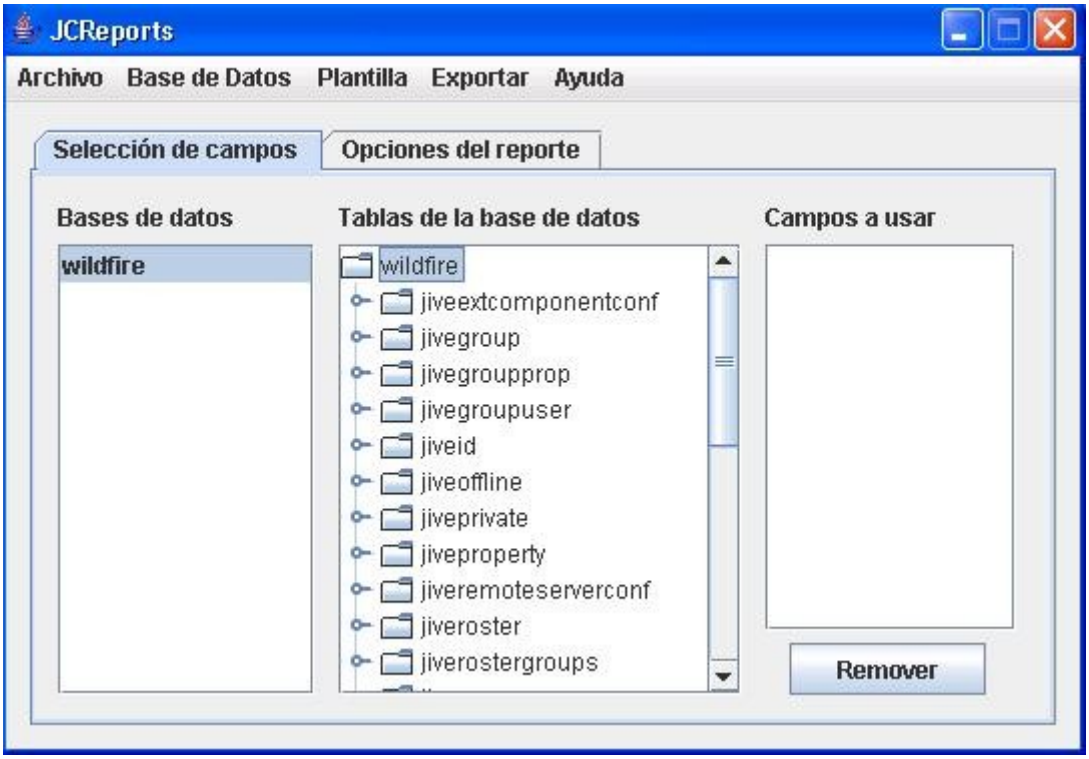

Imagen A.4. *Al seleccionar una base de datos aparecen las tablas que la conforman.*

Al seleccionar una tabla de la columna ***Tablas de la base de datos*** aparecerán los campos que contiene cada tabla.

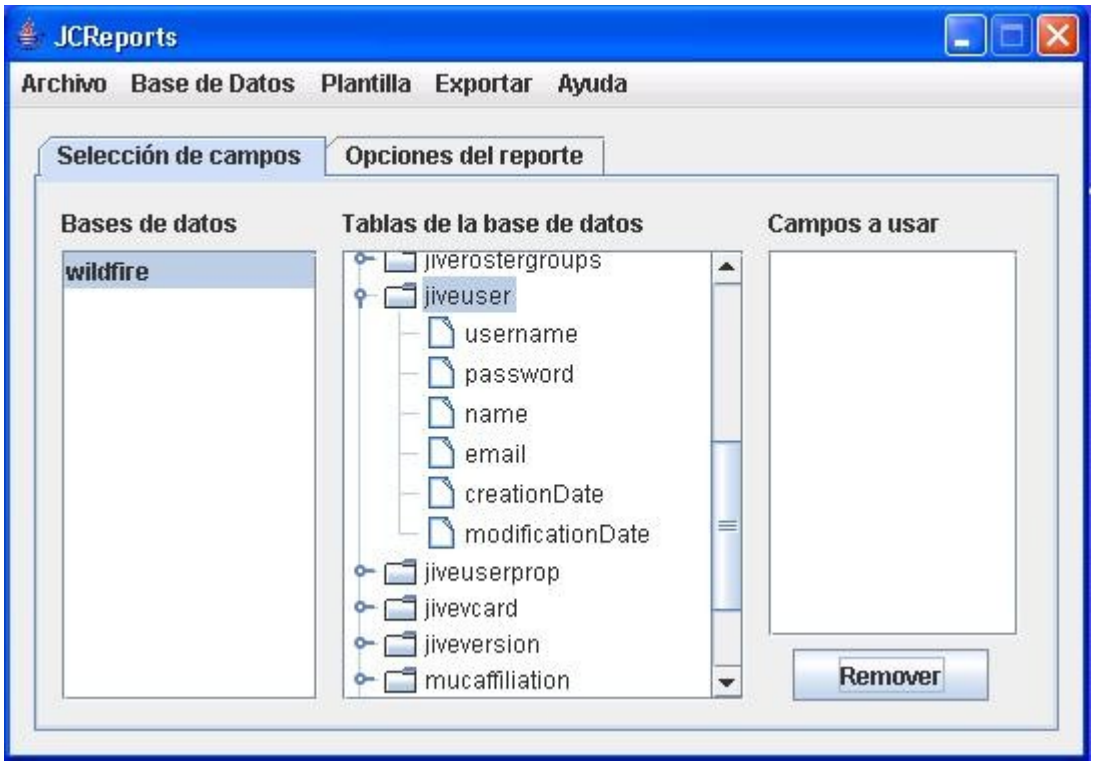

Imagen A.5. *Al seleccionar una tabla aparecen los campos que la conforman.*

Al hacer doble clic en cada campo que desee agregar al reporte este aparecerá en la columna ***Campos a usar.*** si decide no usar algún campo de los agregados, selecciónelo y haga clic en el botón ***Remover***.

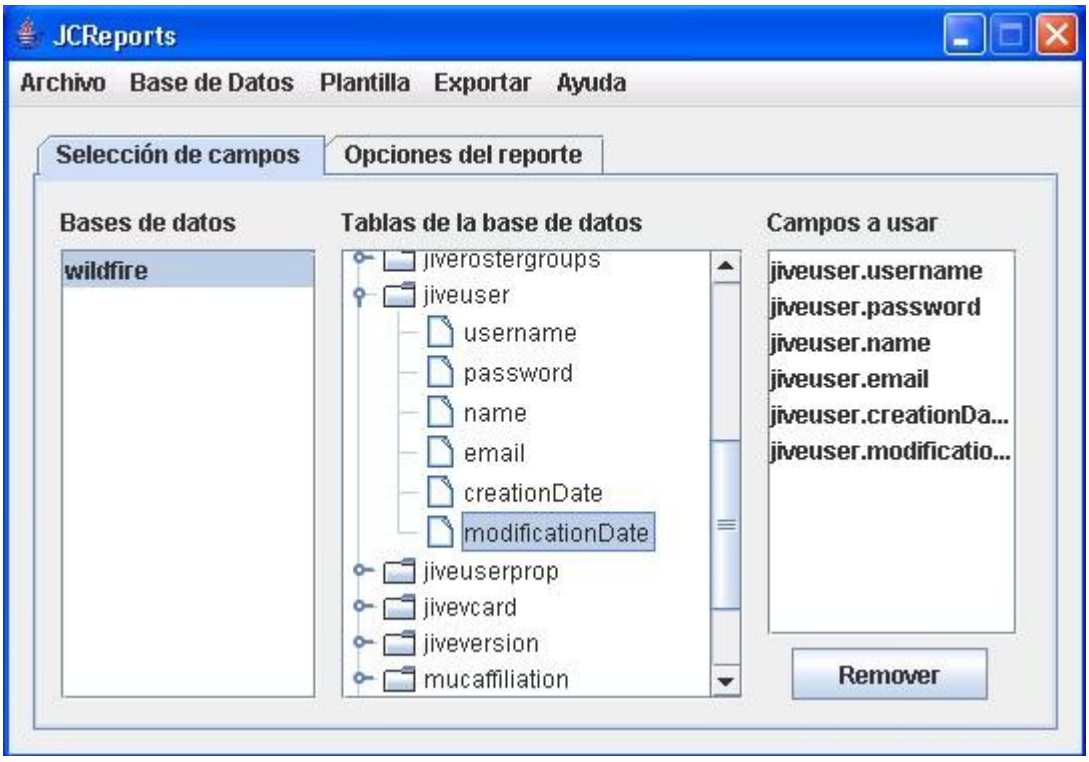

Imagen A.6. *Al hacer doble clic en un campo este se agrega a la columna Campos a Usar.*

A continuación proceda a especificar los detalles del reporte en la opción ***Opciones del reporte.*** Estas opciones son:

- Título.
- Imagen, en caso que desee agregar una imagen.
- Comentarios adicionales.

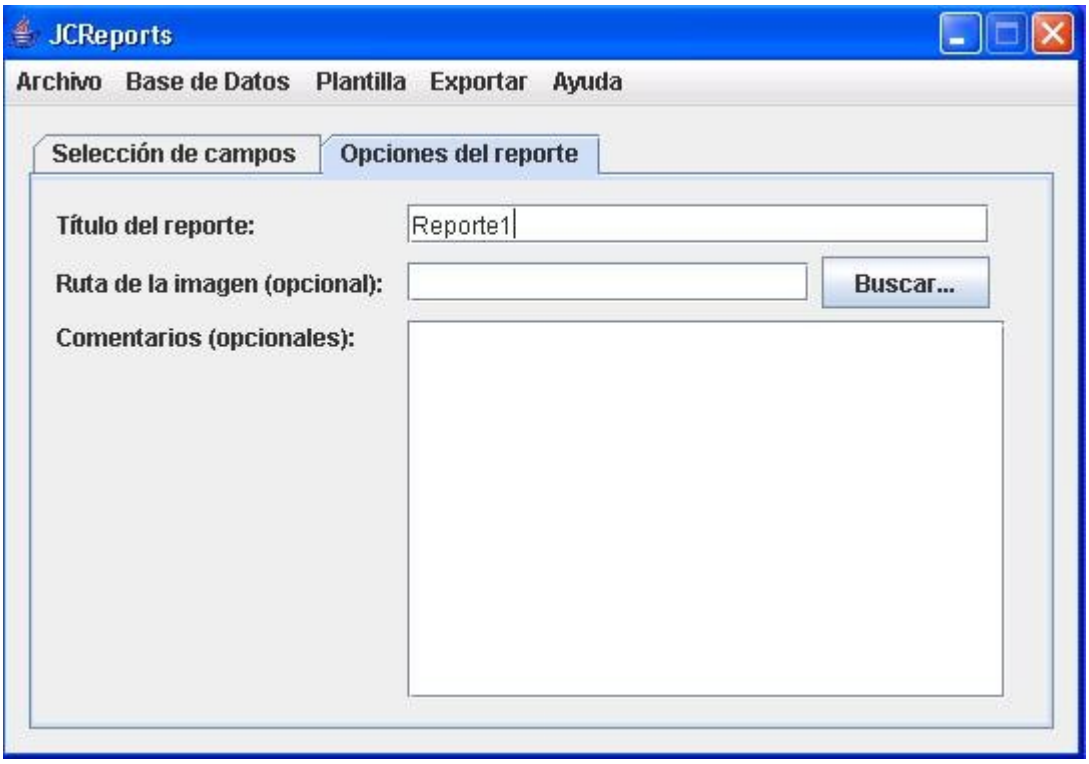

Imagen A.7. *Ingresando el titulo.*

Para agregar una imagen haga clic en el botón ***Buscar...***. Aparecerá un cuadro de diálogo para seleccionar la imagen. Busque la imagen de su preferencia y haga clic en ***Abrir***. La ruta de la imagen se mostrará en el cuadro de texto que se encuentra al lado del botón ***Buscar...***. Esta ruta no puede ser modificada a mano, por lo que, si desea cambiar la imagen seleccionada, deberá repetir el procedimiento.

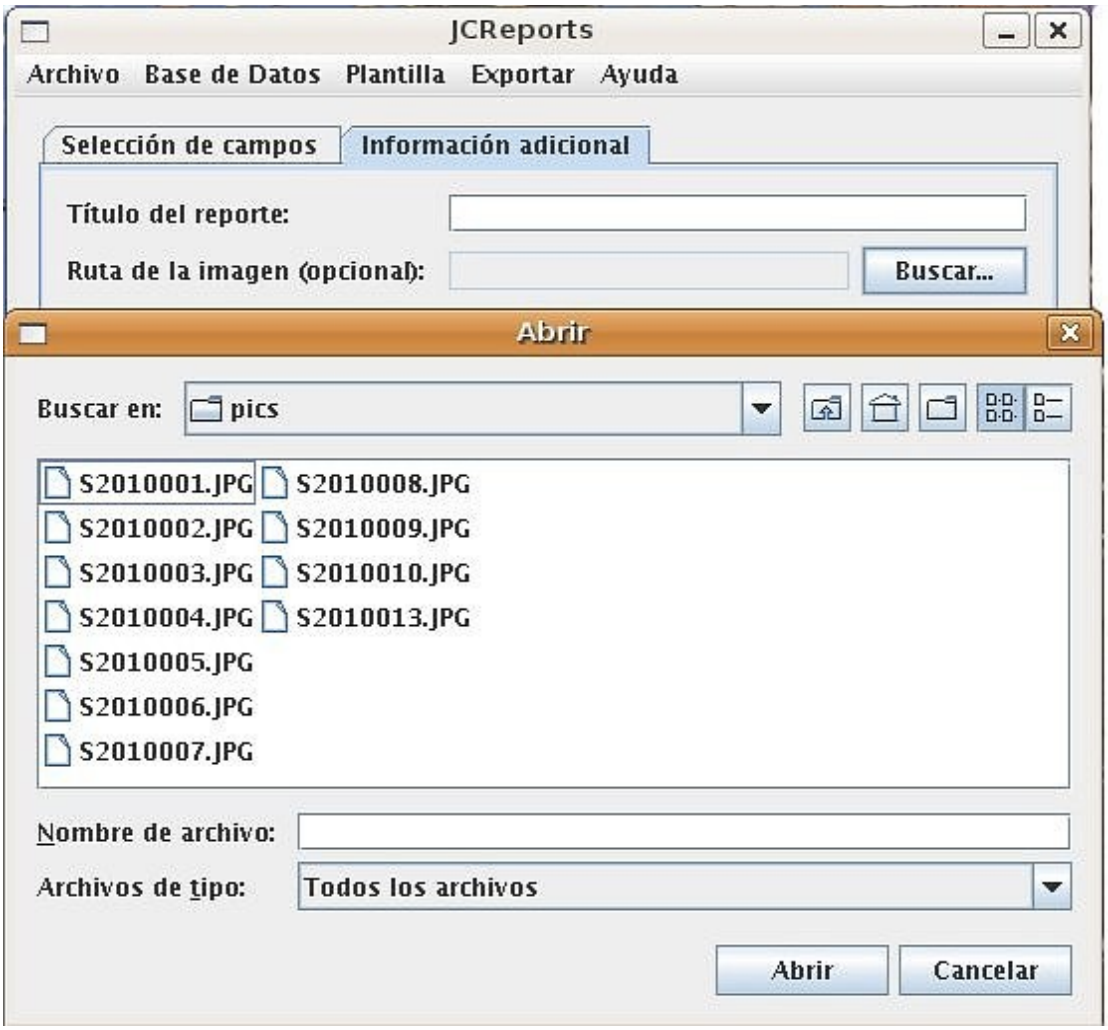

Imagen A.8. *Selección de imágenes.*

Para exportar el reporte, haga clic en el menú ***Exportar***. Aparecerán dos opciones de exportación: XML y HTML. Seleccione la opción de su preferencia, y aperecerá un cuadro de diálogo. Seleccione la ubicación donde desea guardar el archivo, escriba un nombre y haga clic en ***Guardar***. **IMPORTANTE: Si elige un archivo existente, el contenido del archivo será reemplazado.**

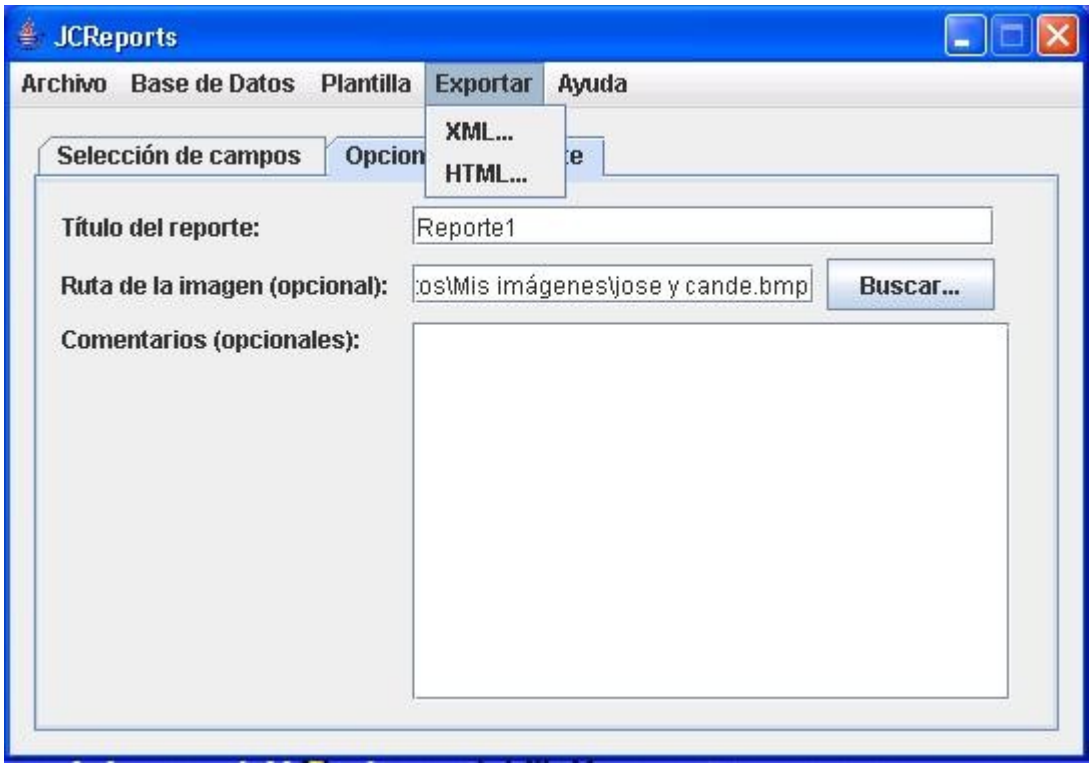

Imagen A.9. *Menú de exportación*

Tenga en cuenta que, si desea exportar al formato HTML, antes debe seleccionar una plantilla de exportación. Para hacer esto, haga clic en la opción ***Usar plantilla...*** del menú ***Plantilla***. Aparecerá un cuadro de diálogo, similar al visto en la opción para agregar imágenes. Seleccione la plantilla de su elección y haga clic en ***Abrir***.

Para visualizar los reportes que haya creado, puede utilizar su navegador favorito.

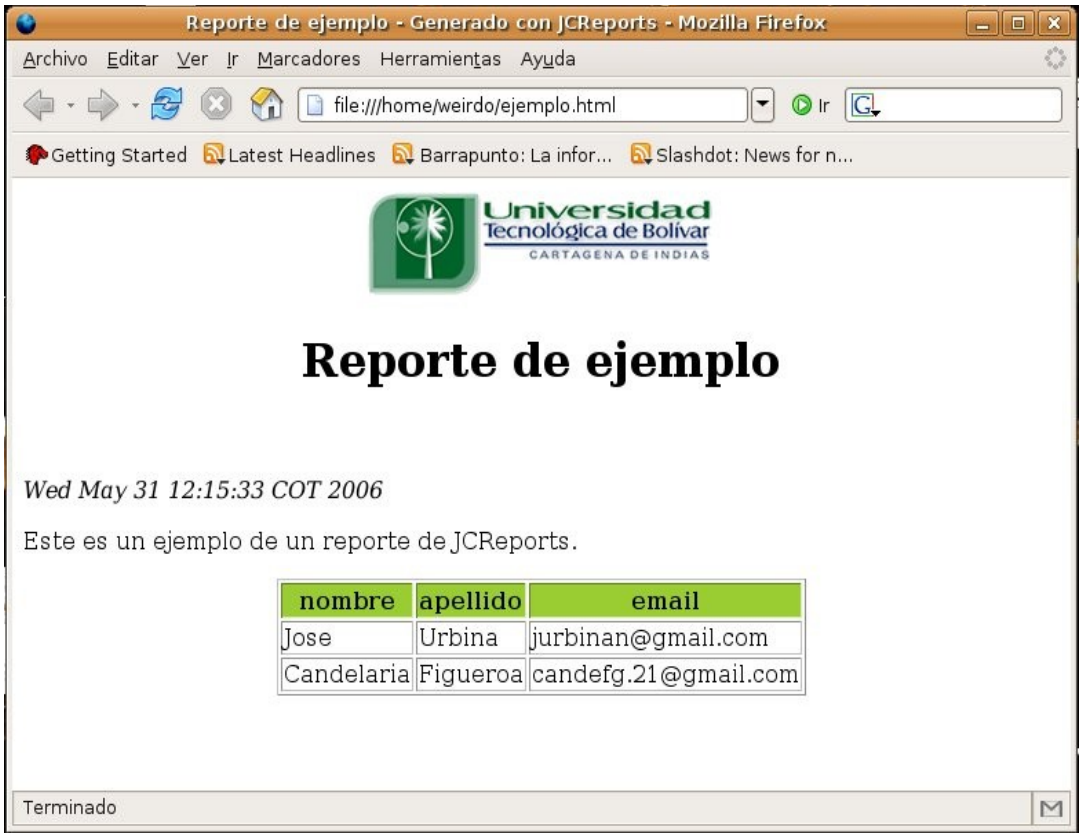

Imagen A.10. *Ejemplo de reporte visualizado con Mozilla Firefox.*