# Eguana Reports – Servidor de Reportes Basado en Tecnología Java y XML

Roy Stalin Cox Sosa[1], David Fernando Pérez Mawyín[2], José Xavier Pérez Sigüenza[3], Luis Ángel Muñoz[4]
Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 vía Perimetral, Apartado 09-01-5863. Guayaquil, Ecuador
[1]roy_wolf2002@hotmail.com, [2]vertebreaker@hotmail.com, [3]jxperezs@hotmail.com, [4]juniormunoz@gmail.com

## Resumen

*Eguana Reports es un sistema de reportes desarrollado por los estudiantes del Tópico de Graduación "Desarrollo de Aplicaciones Transaccionales con Java y XML".*

*El sistema demuestra el uso extenso de tecnología Java EE (Java Enterprise Edition). Bajo esta plataforma se puede encontrar, dentro del proyecto, varias herramientas de código abierto integradas con el fin de alcanzar un objetivo. Asimismo, este documento puede constituir una guía práctica para aquellos desarrolladores interesados en Java EE.*

*Eguana Reports, como módulo, es la solución a la necesidad de reportes del sistema de E-Guana. E-Guana es una iniciativa de implementación de un sistema E-Procurement, desarrollada en módulos por los demás integrantes del tópico de graduación. A pesar de esto, este sistema de reportes no depende de los otros módulos para su funcionamiento.*

*Eguana Reports es una aplicación independiente. Puede integrarse con cualquier otra aplicación que utilice una fuente de datos y generar reportes personalizados a partir de estos.*

**Palabras Claves:** *Eguana Reports, Java EE, XML, reportes, Jasper Reports, Struts, Spring, Hibernate.*

## Abstract

*Eguana Reports is a reporting system developed by students of the Graduate Topic "Transactional Application Development with Java and XML".*

*The system shows the extensive use of technology Java EE (Java Enterprise Edition). Under this platform can be found, within the project, several open source tools integrated in order to achieve a goal. This paper also can be a practical guide for those developers interested in Java EE.*

*Eguana Reports, as a module, is the solution to the need for reporting of E-Guana. E-Guana is an initiative for an E-Procurement system implementation, developed in modules by the other students on the topic of graduation. Despite this, the reporting system does not depend on other modules to function.*

*Eguana Reports is a standalone application. It can integrate with any application using a data source and generate custom reports from these.*

# 1. Introducción

El objetivo de este trabajo es demostrar la capacidad de la plataforma J2EE y el movimiento Código Abierto (Open Source, en inglés), además de la integración de cada uno de los componentes utilizados. Este proyecto fue desarrollado completamente con herramientas y plataformas de código abierto.

El proyecto se construye con tecnología de código abierto, el estándar J2EE y XML, para probar su integración y funcionalidad para iniciativas útiles y de necesidades reales en un ambiente empresarial.

Eguana Reports presenta una base tecnológica que permite, de manera dinámica, obtener reportes personalizables. Los reportes pueden ser diseñados por los usuarios, pudiendo elegir los campos a usar, el formato de salida o la fuente de datos. De esta manera, si se desea consultar información de diferentes aplicaciones, se tiene una fuente única de estos reportes.

Al tener que mantener una fuente única de acceso a la información se pueden liberar recursos y hacer más rentable a la organización.

# 2. Justificación y Objetivos

## 2.1. Objetivos

Este proyecto tiene como objetivos:

- Utilizar herramientas de código abierto (open source), ya que son de fácil acceso y de bajo o cero costos de adquisición, lo que aminora los costos del proyecto y del mantenimiento de la tecnología.
- Crear reportes en formato PDF, HTML, XLS, CSV. Formatos comúnmente usados y de fácil exportación, en caso de ser necesario, a sistemas externos a la organización.
- Obtener reportes a partir de plantillas (formatos predefinidos). Estas plantillas son previamente grabadas en el servidor.
- Proveer un módulo para crear reportes personalizados.
- Tener un módulo propio para administrar usuarios, reportes y fuentes de datos. Esto nos permitirá organizar, restringir y delegar el acceso a los datos y su presentación.
- Definir un esquema de seguridad básica para el acceso a los reportes.
- Unificar la fuente de datos y de reportes dentro de una empresa.
- Proveer un método de integración con sistemas existentes.
- Permitir al equipo de desarrollo enfocar los recursos al sistema y no a módulos de reporte.

## 2.2. Justificación de la tecnología Java EE

Plataforma Java Edición Empresarial (Java EE) (Java Platform Enterprise Edition, en inglés), conocida formalmente como J2EE, se construye sobre la sólida base de la Plataforma Java Edición Estándar (Java SE) (Java Platform Standard Edition). Es una plataforma ampliamente utilizada para el desarrollo de aplicaciones empresariales multicapas, y es considerada el estándar industrial para implementar arquitecturas empresariales orientadas a objetos.

El nombre J2EE es usado hasta la versión Java EE 1.4. En versiones posteriores el término usado es Java EE. De aquí en adelante usaremos el término Java EE.

Java EE, además, constituye un conjunto de estándares, o colección de especificaciones y normas, para el desarrollo e implementación de aplicaciones distribuidas.

En este aspecto, y ya que las especificaciones deben ser aprobadas en consenso por comités de expertos, las aplicaciones tienen el potencial de ser *escalables* y de incluir otras características como *manejo de transacciones, concurrencia, seguridad y tolerancia a fallas*.

**2.2.1. Componentes Java EE.** Las aplicaciones Java EE están formadas por componentes. Un componente Java EE es una unidad de software funcional que se ensambla a una aplicación Java EE con sus clases y archivos relacionados y que se comunica con otros componentes. La especificación Java EE define los siguientes componentes:

- Aplicaciones clientes y applets.- Componentes que se ejecutan en el cliente.
- Componentes de tecnología JSP (Java Server Pages) y Java Servlet.- Componentes Web que se ejecutan en el servidor.
- Componentes EJB (Enterprise JavaBeans).- Componentes de negocios que se ejecutan en el servidor.

Los componentes Java EE están escritos en lenguaje Java.

Un componente Java EE, para ensamblarse a una aplicación Java EE, debe cumplir con las especificaciones Java EE

**2.2.3. Contenedores Java EE.** La arquitectura Java EE hace que las aplicaciones Java EE sean fáciles de implementar, gracias a que se construye a partir de componentes e independencia de plataforma. Esta arquitectura abstrae al desarrollador de las complicaciones de implementar manejo de transacciones, concurrencia, multihilos y otros detalles de bajo nivel en aplicaciones distribuidas.

De aquí que un servidor Java EE provee los servicios necesarios a todo componente bajo el término de Contenedor Java EE. No es necesario que

el desarrollador cree nuevos componentes, simplemente usa los que provee el servidor, y concentra sus esfuerzos en resolver problemas del negocio.

### 2.3. Arquitectura de aplicación MVC (Modelo-Vista-Controlador)

En el paradigma MVC (Modelo-Vista-Controlador) las entradas del usuario, el modelo de aplicación y la retroalimentación visual están explícitamente separados y manejados cada uno por una entidad especializada.

- Vista (View).- Maneja la salida textual/gráfica que se presenta al usuario.
- Controlador (Controller).- Maneja las entradas del usuario, el procesamiento de datos y la lógica de la aplicación, y delega los resultados al Modelo o la Vista.
- Modelo (Model).- Maneja el comportamiento y los datos dentro del dominio de la aplicación. Responde sobre su estado (generalmente a la Vista), y responde a instrucciones (generalmente del Controlador).

Esta separación de tareas beneficia en la reducción de código duplicado, centralizando control y haciendo que la aplicación sea más fácil de modificar.

## 3. Análisis

Eguana Reports se ha pensado como un servidor de reportes.

Un servidor de reportes tiene la misión de proveer a los usuarios una manera fácil de crear y generar reportes, utilizar reportes previamente parametrizados y diseñados, además de permitir hacerlo rápidamente y de manera segura. Un atributo importante también es poder compartir los reportes con otros usuarios afines, por ejemplo, por departamentos de una empresa.

De manera global se puede enfocar de la siguiente manera: Eguana Reports tiene su núcleo en JasperReports, un poderoso motor de reportes muy utilizado hoy en día. El problema es que hay que conocer los conceptos y aspectos técnicos cuando se crea y ejecuta reportes en JasperReports. Eguana Reports soluciona este problema abtrayendo al usuario del trabajo pesado. El usuario no necesita ser experto para diseñar y ejecutar reportes. Además Eguana Reports provee un esquema de administración para organizar, centralizar y dar seguridad al repositorio de reportes y personas que requieren acceso.

En resumen, Eguana Reports se encarga de facilitar la Administración, el Diseño y la Ejecución de reportes basados en JasperReports.

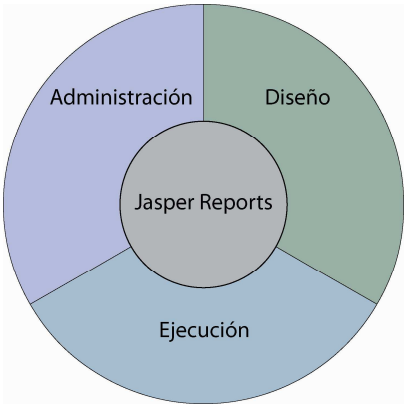


**Figura 1.** Enfoque de Eguana Reports

### 3.1. Funcionalidad

La aplicación se divide en pequeños módulos que cumplen una funcionalidad específica. Las funciones deseadas dentro de Eguana Reports son:

- Administración Eguana Reports.- Para agregar fuentes de datos, reportes, usuarios y grupos, y cualquier otra tarea de administración general. Cada una de estas tareas constituye en sí un pequeño módulo dentro de este.
- Creación y validación de plantilla de reporte.- la plantilla es un diseño en xml que se compila para crear un reporte listo para usar.
- Acceso a fuentes de datos.- Dependiendo del tipo de fuente de datos, o del modo de acceso (JDBC, JNDI), crea una conexión para que el generador de reportes la use.
- Generación de reportes.- Obtiene una conexión a la fuente de datos y crea un reporte con la información basándose en un diseño de reporte previamente creado y validado.
- Conversión de formato de presentación.- Convertir el formato a HTML, XLS, PDF, CSV.
- Entrega de reporte.- Entrega de reporte por un medio determinado: pantalla o archivo. Esta función se relaciona con la conversión de formato.
- Control de acceso de usuarios y grupos.- Se crean grupos de usuarios que tienen acceso a cierto grupo de reportes. Se debe validar el acceso al momento de querer generar un reporte.

El siguiente sería el orden del proceso desde la creación hasta la entrega del reporte al usuario final:

1. Crear una plantilla (diseño de reporte) válida, con el uso de un editor de plantilla (Ej.: iReports, JasperAssistant).
2. La plantilla creada se asocia a un usuario o un grupo que tendrá permiso para su uso en la generación de reporte.
3. El usuario de otra aplicación (por ejemplo Eguana E-Procurement) que desea generar un reporte específico hará una petición al generador

de reportes de Eguana Reports, haciendo referencia a un diseño previamente creado en forma de plantilla.

4. Verificando el control de acceso (grupo-usuario-reporte) válido.
5. El generador de reporte obtiene una conexión a la fuente de datos, previamente configurada (JNDI, JDBC), de donde se extrae la información que contendrá el reporte.
6. Según el diseño de reporte se genera el reporte.
7. Convierte el formato de presentación del reporte, de ser necesario (a HTML, PDF, XLS, entre otros).
8. Se lo entrega al usuario final.
9. Este esquema internamente hace uso de varias tecnologías y herramientas Java EE, descritas en el capítulo anterior, para lograr tareas específicas.

### 3.2. Diagrama de clases

El diagrama de clases es básico para el diseño del sistema y sus componentes. Representan el modelo de la realidad. El objeto primordial es Reporte.

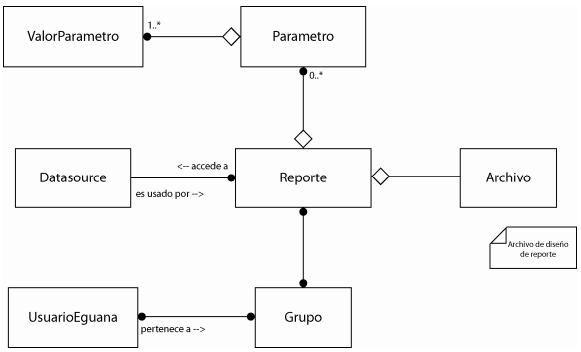


**Figura 2.** Diagrama simplificado de objetos

### 3.3. Casos de uso

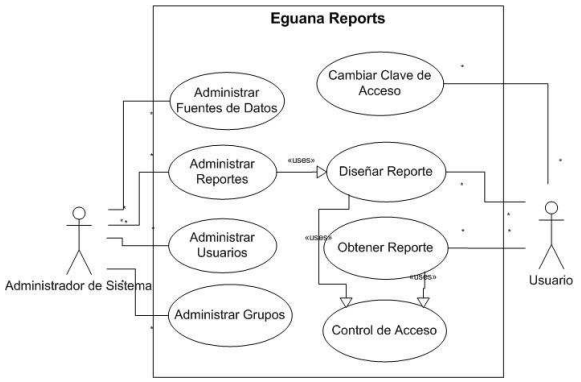


**Figura 3.** Diagrama de casos de uso

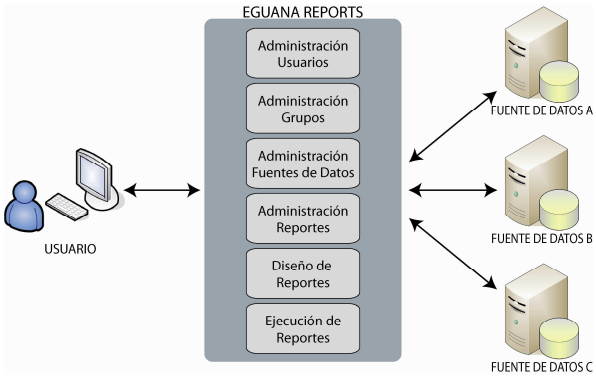


**Figura 4.** Esquema general de funcionamiento

## 4. Diseño

El modelo de aplicación Java EE define un modelo de N-capas para una aplicación distribuida, un concepto desde el punto de vista general de una aplicación de acuerdo a la funcionalidad y al lugar de ejecución de cada capa. Aquí se incluye la base de datos como capa, y la aplicación del cliente como parte de otra capa. La arquitectura MVC define un modelo para separar los datos de una aplicación y mantener una estructura interna ordenada (en tres partes: modelo, vista y controlador).

Podemos resumir en el siguiente gráfico cómo se aplica el modelo multicapas de Java EE junto con el modelo MVC en nuestra aplicación.

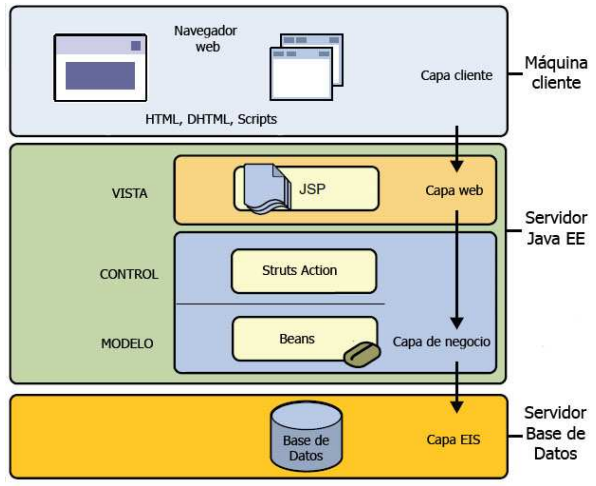


**Figura 5.** MVC y modelo de aplicación Java EE

Según esto aplicamos la tecnología y componentes necesarios para cada funcionalidad en cada capa. Nuestra aplicación queda detallada en tres capas: capa de persistencia y modelo, capa de lógica y control, capa de vista. A continuación se muestra un bosquejo general.

### 4.1. Capa de Persistencia y Modelo

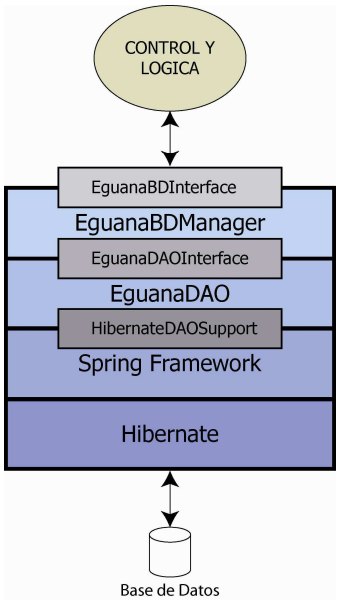


**Figura 6.** Capa de persistencia y modelo

### 4.2. Capa de Vista

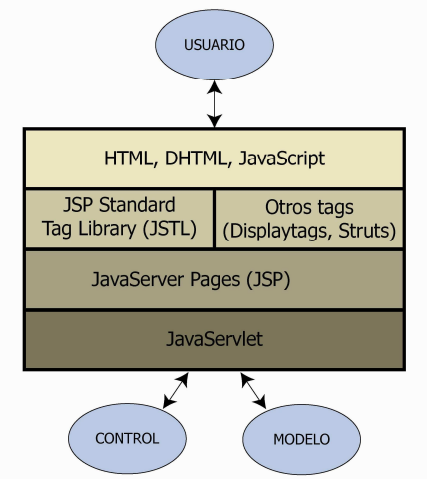


**Figura 7.** Capa de vista

### 4.3. Capa de Lógica y Control

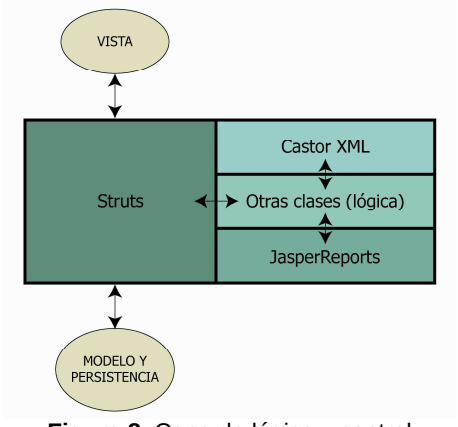


**Figura 8.** Capa de lógica y control

## 5. Implementación

### 5.1. Diseño de reportes

Existen dos formas de proveer un archivo JRXML al sistema:

La primera es crear el diseño en una herramienta visual, como iReport, y luego cargándolo al sistema con la interfaz para creación y mantenimiento de reportes.

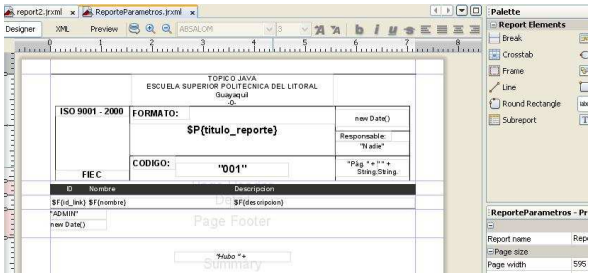


**Figura 10.** Diseño de reporte con iReports

La segunda es crear el diseño mediante la interfaz de Eguana Reports. Este diseñador no es visual, pero es un prototipo que sirve para generar reportes básicos de una manera muy sencilla y dinámica. El nivel de conocimientos necesarios para diseñar un reporte con Eguana Reports es menor.

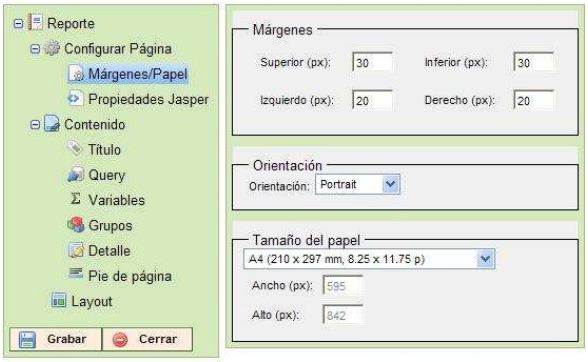


**Figura 11.** Diseñar un reporte con Eguana Reports

### 5.2. Ejecución de reportes

La idea aquí es que este proceso sea simplificado, y un usuario pueda obtener el reporte deseado con unos pocos clics.

Cuando un reporte se ejecuta aparece la interfaz para elegir el formato de salida e ingresar los parámetros del reporte.

Estos parámetros son definidos en el momento de creación y diseño del reporte, de modo que escoge según sea conveniente el valor a partir de elementos HTML, como listas, campo de textos y checkbox.

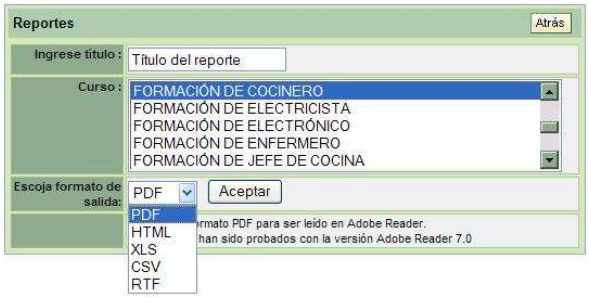


**Figura 12.** Ingreso de parámetros

El resultado de la ejecución de reportes creados en Eguana Reports lo obtenemos en el formato escogido en el paso previo (PDF, XLS, HTML, CSV, RTF). Por ejemplo, un reporte en PDF:

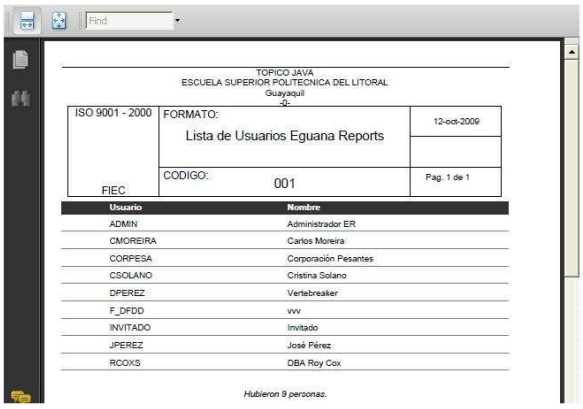


**Figura 13.** Ejemplo de reporte

## 6. Conclusiones

- El desarrollo de aplicaciones en Java y código abierto tiene sus ventajas y desventajas. La disponibilidad de software para cumplir cualquier objetivo es alta, pero es eso mismo lo que dificulta la tarea de elegir qué herramienta usar. La tarea de selección muestra que, a diferencia de los demás módulos de Eguana, no fue necesario usar EJBs para implementar la solución a nuestro modelo. En vez de eso, utilizamos Hibernate.
- En el mundo de código abierto las actualizaciones son más rápidas que con código propietario. Esto se debe a la cantidad de gente colaborando, muchos gratis y otros como parte de las fundaciones de código abierto.
- La rapidez hace que aparezcan nuevas herramientas día a día. Ya en la actualidad las versiones usadas en nuestro proyecto no son las últimas, e incluso existen otras tecnologías como JSF (Java Server Faces).
- Aunque la curva de aprendizaje es pronunciada, el uso de Java EE y XML en la arquitectura de la aplicación asegura que sea escalable y portable, y que se ajuste a las necesidades de crecimiento futuro en una empresa.
- Eguana Reports demuestra la versatilidad de JasperReports para generar reportes, que es una razón de que sea una herramienta muy utilizada hoy en día. Eguana Reports se basa en Jasper Reports para crear un ambiente más dinámico en la generación de un reporte, disminuyendo el tiempo que toma diseñar el mismo y obtener resultados.
- Generar reportes en varios formatos permite que este proyecto se adapte a las necesidades del cliente al momento de elegir el programa donde desea ver los reportes, de una manera sencilla y dinámica.
- Tener Eguana Reports, como servidor de reportes, permite a una empresa centralizar el almacenamiento y generación de reportes, optimizando los recursos, esfuerzos y organizando la tarea de reportería en un solo lugar.
- Eguana Reports provee un entorno seguro de acceso a los reportes mediante usuarios, grupos y permisos. Se centraliza el almacenamiento y generación de reportes. Se puede implementar un mejor esquema de seguridad. Además se pueden distribuir los reportes por grupos o departamentos de una empresa y controlar mejor el acceso a reportes.
- Con la creación del rol Administrador de Reportes en una empresa, es posible abstraer a los desarrolladores del problema de incluir reportes, o los mecanismos necesarios para generarlos, en cada módulo de la aplicación y así evitar la desorganización.
- El prototipo de diseñador de reportes incluido en Eguana Reports permite crear reportes básicos aún más rápido que tan sólo utilizando Jasper Reports, sin necesidad de programar y con pocos conocimientos en diseñar reportes.

## 7. Recomendaciones

- Con respecto al desarrollo con código abierto, Java EE y XML, se recomienda buscar la mayor cantidad de ayuda posible cuando no se está claro sobre las tecnologías involucradas
- Es recomendable que el desarrollador forme parte de un grupo que implemente soluciones en el mercado, en la vida real. La mejor forma de aprender es poner la teoría en práctica.
- Con respecto a nuestra aplicación, se recomienda su mejora constante en la aceptación de diseños de versiones distintas de JasperReports. JasperReports se actualiza constantemente y los formatos de diseño cambian.
- Se recomienda también mejorar el prototipo de diseñador. Por ahora se provee un diseño elemental. Aún así, la base para el crecimiento está estructurada desde el diseño gracias al uso de XML.

## 8. Referencias


[1] Sun Microsystems, The Java EE 5 Tutorial, https://www.sun.com/offers/docs/JavaEETutorial.pdf, Septiembre 2007.

[2] JBoss Community, Installation and Getting Started Guide, www.jboss.org, 2008.

[3] Wikipedia, articulos varios, www.wikipedia.org, 2009.

[4] Apache Software Foundation, The Apache Tomcat 5.5 Servlet/JSP Container, http://tomcat.apache.org/tomcat-5.5-doc/index.html, 2009.

[5] Sun Microsystems, MySQL 5.1 Reference Manual, http://www.mysql.com/doc , 2009

[6] The Apache Sofware Foundation, Struts, http://struts.apache.org, 2008

[7] Hibernate Community, Hibernate Reference Documentation, http://www.hibernate.org/documentation, 2007

[8] David R. Heffelfinger, JasperReports for Java Developers, PACKT Publishing, 2006

[9] Werner Guttmann, Castor 1.3 – Reference documentation, The Castor Project, 2009

[10] Gary Cernosek - IBM, A Brief History of Eclipse, http://www.ibm.com/developerworks/rational/library/nov05/cernosek/ , 2009

[11] Eclipse Foundation, Eclipse 3.1 Documentation, http://www.eclipse.org/documentation/, 2009

[12] Trygve Reenskaug, MVC – Xerox PARC 1978 – 79, http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html, 2009

[13] Trygve Reenskaug, “The Model-View-Controller (MVC), Its Past and Present”, University of Oslo, Agosto 2003

[14] Trygve Reenskaug, MODELS - VIEWS – CONTROLLERS (10-dic-1979), http://heim.ifi.uio.no/~trygver/, 2009