Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Título: ***"Creación de componente de generación de reportes para ABCD 3.0"***

Autor:

Gerardo Díaz Rodríguez

Tutor:

Ing. Oigres Alvarez Pérez

La Habana, junio 2016

"Año 58 de la Revolución"

"Sueña y serás libre en espíritu, lucha y serás libre en vida"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____días del mes de _____del año _____.

Gerardo Díaz Rodríguez	Oigres Alvarez Pérez
_____	_____
Autor	Tutor

DATOS DE CONTACTO

Graduado de Ingeniero en Ciencias Informáticas en la Facultad 2 de la Universidad de las Ciencias Informáticas (UCI), en el año 2008. Profesor de la misma que labora en el Departamento de Práctica Profesional del Centro de Informatización de la Gestión Documental impartiendo las asignaturas de Proyecto de Investigación y Desarrollo IV, VI y VII. Posee 7 años de experiencia en el desarrollo de aplicaciones empresariales implementadas en java. Actualmente, es arquitecto del proyecto Automatización de Bibliotecas y Centros de Documentación 3.0.

Correo: oaperez@uci.cu

DEDICATORIA

Dedico este trabajo de diploma a toda mi familia y en especial a mi madre María Victoria Rodríguez Cardentey quien siempre ha estado a mi lado apoyándome con esta meta.

AGRADECIMIENTOS

Quisiera agradecer a mi familia en general por todo su apoyo durante toda mi vida como estudiante. A mi tío Roje, a mi tía Pilar "mi gordi y segunda madre", a mi tía Marielena gracias por ser como eres. A mi abuela Zenaida por siempre transmitirme esas buenas energías que tiene en todo momento, a mis hermanos y primos. Quisiera darles un agradecimiento especial a mis padres, a mi papá Gerardo y a ese otro padre que la vida me regaló y a quien doy gracias por criarme como su hijo y apoyarme en toda mi carrera, a ti Sergio gracias. A mi madre la persona que siempre me ha apoyado desde que nací y quien ha sido mi guía durante toda mi vida, a ella le regalo este título. A Adela Sánchez profesora, amiga y madre desde el preuniversitario por toda su ayuda y apoyo en mi carrera. Quiero agradecer a todos aquellos profesores que contribuyeron a mi formación y que me ayudaron a salir adelante en cada momento de mi carrera como estudiante. A mi tutor Oigres por confiar en mí para la realización de este trabajo y por ayudarme en todo momento en esta difícil tarea, gracias profe. A todos mis amigos del preuniversitarios y a los de la universidad por brindarme su apoyo en las buenas y en las malas cuando lo necesité. A todos aquellos que me han ayudado de una forma u otra a llegar aquí, gracias.

Gerardo Díaz Rodríguez

RESUMEN

El componente generación de reportes para el Sistema de Automatización de Bibliotecas y Centros de Documentación es un proyecto sustentado sobre las necesidades de la Arquitectura Base Orientada a Servicios de exportar información existente en el sistema en formatos pdf y xls. El objetivo principal de esta investigación es la creación de un componente que se integre en dicha arquitectura y que permita exportar los reportes en los formatos propuestos.

Para la realización del componente se emplearon las herramientas de Itext, Apache POI y JFreeChart para construir los reportes en los formatos pertinentes, así como el lenguaje de modelado UML para representar los diagramas correspondientes al mismo. En el desarrollo de la investigación se siguieron los pasos que propone el Proceso Unificado de Desarrollo de Software como metodología rectora. Para comprobar el correcto funcionamiento de las funcionalidades del componente se le realizaron pruebas de caja negra a cada funcionalidad.

Como resultado final de este trabajo se obtuvo un sistema que permite exportar la información en reportes con formatos de documento pdf y xls. Los reportes pueden incluir tablas comparativas, párrafos, imágenes, gráficas estadísticas como son las gráficas de barras, de pastel y de línea, en cualquier orden o cantidad, cumpliéndose de esta forma los objetivos propuestos.

Palabras clave:

Arquitectura ABOS, Componente, Reporte

TABLA DE CONTENIDOS

INTRODUCCIÓN

Los reportes son documentos que presentan de manera estructurada y resumida datos que son relevantes. Antiguamente se confeccionaban en formato duro, ya que no existía ninguna herramienta capaz de realizarlos. En la actualidad con el desarrollo alcanzado el empleo del formato digital ha venido ganando terreno debido a su facilidad, así como la presencia de aplicaciones que son capaces de generar reportes automáticamente. Dichas aplicaciones permiten obtener la información deseada con mayor rapidez ofreciendo un mejor nivel de detalle y flexibilidad.

A los reportes se le pueden realizar ciertas operaciones como imprimirlo, enviarlos por email, guardarlos en un archivo, entre otras. Los mismos pueden ser un material creado por una empresa, organización o un trabajo de clases que permita dar una mayor información sobre un tema en específico. Los reportes pueden hacer uso de una gran variedad de elementos para cumplir con su cometido. Así, por ejemplo, es posible el uso de gráficos de barras, de curvas sobre ejes cartesianos, de tablas comparativas entre otras.

Actualmente en la Universidad de las Ciencias informáticas (UCI) se desarrolla el proyecto de Automatización de Bibliotecas y Centros de Documentación (ABCD) cuyo objetivo es la informatización de los procesos existentes en las bibliotecas. Este proyecto está montado sobre la arquitectura ABOS (Arquitectura Base Orientada a Servicios) implementada en java. En la misma se necesita de la transformación y presentación de la información existente en el sistema ABCD 3.0 en formatos (pdf, xls). Por estos antecedentes se plantea a manera de **situación problemática** que:

- La arquitectura ABOS no contiene el componente para exportar la información a pdf y xls.
- En la actualidad no se puede obtener la información que se procesa en el sistema ABCD 3.0.
- Se necesita que la información contenga una estructura claramente definida, pero que a la vez pueda ser configurable para los distintos tipos de formatos a exportar el reporte, así como el tipo de hojas y estructura del documento.
- El sistema debe presentar la información conformada por párrafos, imágenes, tablas comparativas y gráficas de comparación como son: gráficas de barras, gráficas de pastel y gráficas de líneas.
- Los reportes que se obtengan deben mostrar la información que se requiera en cualquiera de sus combinaciones y repeticiones.

Sobre la base de estos antecedentes se identificó como **problema científico:** ¿Cómo exportar información en formatos pdf y xls en la arquitectura ABOS?

INTRODUCCIÓN

Para el desarrollo de la investigación se tomó como **objeto de estudio** la generación de reportes en java.

Por tanto, el **objetivo general** es el desarrollo del componente de generación de reportes para la arquitectura ABOS que permita exportar los reportes en formato pdf y xls.

Como **campo de acción** se seleccionó la generación de reporte en la arquitectura ABOS.

Para su cumplimiento se desglosaron los siguientes **objetivos específicos:**

- Valorar el estado las herramientas que utiliza java para generar reportes e identificar las características que estas poseen en la solución de la propuesta.
- Desarrollar el componente generación de reporte para pdf y xls respectivamente.
- Validar el correcto funcionamiento del componente integrado en la arquitectura ABOS.

Para dar cumplimiento a los objetivos propuestos se plantean las siguientes **tareas de investigación:**

- Análisis de la arquitectura ABOS para comprender la estructura y el funcionamiento de los componentes en la misma.
- Caracterización de frameworks y librerías utilizadas en la generación de reportes en java para la selección de las herramientas.
- Análisis de los patrones de diseño para la implementación del componente generación de reporte.
- Estudio del método de prueba de caja negra para comprobar el correcto funcionamiento del componente.

Los **métodos** utilizados en la investigación del **nivel teórico** son:

Analítico-sintético Posibilitaron esencialmente la obtención de la información teórica, necesarias para dar cumplimiento a las tareas de la investigación y se extrajeron los principales aspectos de cada una de ellos, alcanzando así conocimientos generalizados.

Modelación: Facilitó el diseño del componente regeneración de reportes, representando los procesos que engloba la aplicación.

Del **nivel empírico** se utilizó el siguiente:

- **Observación:** Se utilizó para observar y analizar el comportamiento del componente generación de reportes en cuanto a su funcionalidad.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS SOBRE LA ARQUITECTURA ABOS Y LAS HERRAMIENTAS DE GENERACIÓN DE REPORTE

Introducción

En el presente capítulo se presenta el estudio del estado del arte realizado, donde se analiza la arquitectura ABOS y el funcionamiento de los componentes en la misma. Además, se hace alusión a las principales características de las herramientas existentes a nivel internacional para la generación de reportes en java y el aporte que puedan brindar durante la implementación. Se presenta también la metodología de desarrollo de software como metodología rectora del proceso de desarrollo y las principales características de las herramientas utilizadas para el desarrollo del componente.

1.1 Arquitectura ABOS

La Arquitectura Base Orientada a Servicios sustenta el desarrollo de software del sistema ABCD 3.0. Se distingue por ser una arquitectura que permite el desarrollo modular mediante la implementación de componentes conocidos como bundles. Los bundles en la misma, según su funcionalidad pueden ser agrupados en algunas de las capas: Dominio, Acceso a Datos, Servicios de Negocio y Presentación.

Para el uso de base de datos relacionales la arquitectura sugiere el uso del patrón modelo de dominio, mapeado con anotaciones de JPA y el ORM EclipseLink (1). En la capa de acceso a datos se utiliza Spring Data para la generación de las funcionalidades CRUD de cada una de las entidades del dominio.

Las clases del dominio constituyen una capa transversal que es la encargada de transportar los datos entre las otras capas de la arquitectura. La capa de servicios de negocio es la encargada de encapsular la lógica de negocio de cada módulo y ocultar para las capas superiores todas las dependencias de acceso a datos. La capa de presentación se encarga de utilizar la capa de servicio de negocio y de proveer los componentes de presentación, validación, localización, entrada, reportes, etc. En la misma se utiliza RAP (2) para la construcción de las interfaces de usuarios y se utiliza el patrón Modelo-Vista-Controlador para gestionar la interacción entre los eventos de las vistas y las acciones a acometer, implementadas en la capa de servicios.

Cada bundle encapsula la implementación de las interfaces que provee que solo pueden ser accedidas mediante el consumo de los servicios que publica en tiempo de ejecución en la máquina virtual de java (3). Para la configuración de los servicios en cada bundle se especifica en la carpeta META-INF dentro de la carpeta Spring con XML. La descripción se realiza de acuerdo a lo establecido por Spring Dynamic

Modules (Spring DM) (4). El entorno de ejecución para cada bundle de la arquitectura es el framework Equinox 3.8, que implementa la especificación de Open Services Gateway Initiative (OSGI 4.3) y forma parte del entorno de despliegue conformado por el servidor Virgo 3.6.3. A modo de resumen se presenta una síntesis de la arquitectura en la Figura 1:

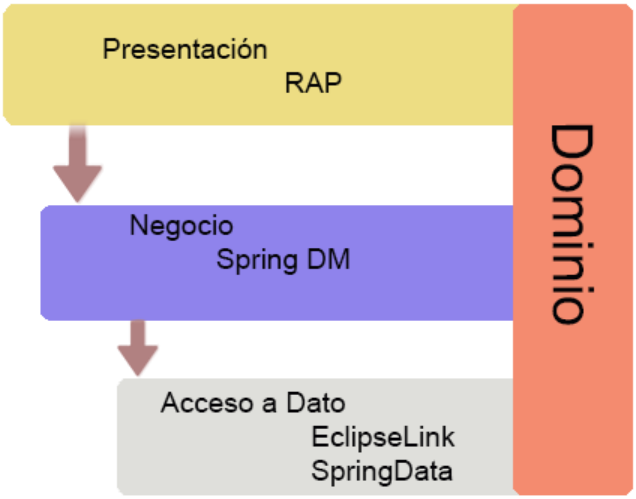


Figura 1: *Arquitectura ABOS*

1.1.1 OSGI

OSGI es un sistema (o framework) modular para Java que establece las formas de crear módulos y la manera en que estos interactuaran entre sí en tiempo de ejecución. Fue creado en marzo de 1999. Solventa los problemas del tradicional "classloader" de la máquina virtual y de los servidores de aplicaciones Java. En OSGI, cada módulo tiene su propio classpath separado del resto de classpath de los demás módulos y la carga de clases ocurre de manera independiente. Este framework proporciona a los desarrolladores un entorno orientado a servicios y basado en componentes, ofrece estándares para manejar los ciclos de vida del software. OSGI, proporciona un marco de trabajo java de uso general, seguro y administrado que soporta el despliegue dinámico de aplicaciones conocidas como "bundles" o módulos (5).

Algunas de las características que componen este marco de trabajo:

- Es un sistema de módulos para la plataforma java.
- Incluye reglas de visibilidad, gestión de dependencias y versionado de los bundles.

- La instalación, arranque, parada, actualización y desinstalación de bundles se realiza dinámicamente en tiempo de ejecución sin tener que detener por completo la plataforma.
- Se trata de una arquitectura orientada a servicios.
- Los servicios pueden ser registrados y consumidos dentro de la máquina virtual de java.

Principales beneficios que proporciona esta tecnología:

- Reutilización del código.
- Simplifica los proyectos en los que participan muchos desarrolladores en diferentes equipos.
- Se puede utilizar en sistemas más pequeños.
- Gestiona los despliegues locales o remotos.
- Se trata de una herramienta fácilmente ampliable.
- No es una tecnología cerrada.

1.1.2 Bundle

Un bundle es una aplicación empaquetada en un fichero jar, que se despliega en una plataforma OSGI. Cada bundle contiene un fichero de metadatos organizado por pares de nombre clave: valor donde se describen las relaciones del bundle con el mundo exterior a él. (Por ejemplo, que paquetes importan o que paquetes exportan). Este fichero viene dado con el nombre de MANIFEST.MF y se encuentra ubicado en el directorio META-INF. La principal característica de los bundles es que brindan servicios a otros bundles para que estos los consuman y a su vez pueden consumir servicios de otros bundles (5).

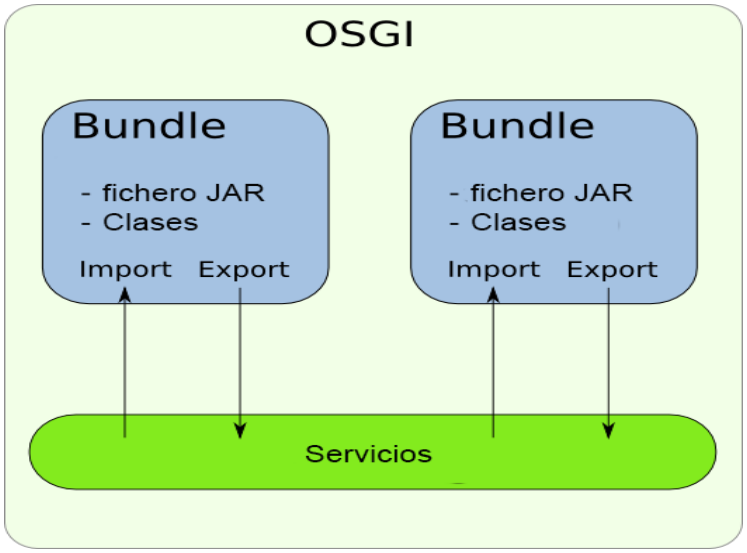


Figura 2: *Bundle*

1.1.3 Spring

Está compuesto por un conjunto de módulos, de los cuales se pretende tomar aquellos que faciliten la implementación del trabajo en cuestión. Spring está basado en el principio de inyección de dependencia. Esta técnica hace externa la creación y el manejo de las dependencias de las clases, con lo que se logra una mayor limpieza y claridad en el código. Es un framework que ha venido a revolucionar la manera de programar aplicaciones en java debido a la facilidad de crear componentes reutilizables (4).

1.1.4 Spring DM

Spring DM es un módulo dinámico para plataformas de servicios OSGI. Ofrece una solución modular para el desarrollo de aplicaciones basadas en Spring que se pueden ejecutar en un entorno de OSGI. El objetivo de esta tecnología es hacer que OSGI y Spring trabajen juntos de una manera simple y que se puedan aprovechar los servicios que ofrece OSGI (6). La combinación de Spring DM y OSGI provee las siguientes características (4):

- Mejor separación de la lógica de la aplicación en módulos.
- La habilidad de desplegar diferentes versiones de los módulos.
- La habilidad de descubrir dinámicamente y usar servicios provistos por otros módulos en el sistema.
- La posibilidad de instalar, modificar y desinstalar módulos en un sistema que se ejecuta.
- Puede usar Spring dentro y entre los módulos.
- Un modelo de programación simple y familiar para los desarrolladores de aplicaciones.
- Posibilita la configuración de servicios con xml con lo cual no es necesario implementar interfaces de OSGI en cada bundle para la exportación y consumo de servicios.

La configuración de los servicios en cada bundle se realiza mediante las especificaciones de Spring DM. La misma proporciona las pautas necesarias para configurar los servicios que se exportan y la integración con los servicios que se consumen. A continuación, se muestra una imagen de la configuración realizada en un bundle configurado para importar y exportar servicios:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/osgi"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xsi:schemaLocation="http://www.springframework.org/schema/osgi
        http://www.springframework.org/schema/osgi/spring-osgi.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <service ref="xlsReportGenerator" interface="cu.uci.abos.reports.XLSReportGenerator" />
    <service ref="xlsDocumentGenerator" interface="cu.uci.abos.reports.ReportTemplate" />

</beans:beans>
```

Figura 3: *Configuración de un bundle para exportar servicios*

1.2 Herramientas para la Generación de Reportes

Son muchas las herramientas reconocidas a nivel internacional para la generación de reportes, sin embarro, teniendo en cuenta que el lenguaje utilizado en la arquitectura ABOS es java se impone centrar el estudio en aquellas herramientas escritas en dicho lenguaje y no sean privativas. De ellas existen cinco herramientas que cumplen con lo antes planteado y a la vez son de las más utilizadas para el desarrollo de software: JasperReports, Itext, JFreeChart, Apache POI y BIRT.

1.2.1 JasperReports

JasperReports está considerada como una de las mejores bibliotecas de software libre para generar reportes. Se puede incluir fácilmente en cualquier aplicación Java para la creación de sofisticados reportes. Su propósito principal es el de crear documentos preparados para imprimir en una forma simple y flexible (7).

Además de los datos en forma de texto, JasperReports es capaz de generar reportes profesionales incluyendo imágenes y gráficos, entre sus funcionalidades más importantes se encuentran (7):

- Posee un diseño de informe flexible.
- Puede presentar los datos de forma textual o gráficamente.
- Los desarrolladores pueden proporcionar datos de múltiples maneras.
- Acepta datos de múltiples fuentes.
- Puede generar marcas de agua.
- Puede generar informes integrados.
- Es capaz de generar los reportes en múltiples formatos.

Descripción de las funcionalidades descritas:

Posee un diseño de informe flexible

JasperReports permite separar los datos en secciones opcionales dentro del reporte. Dichas secciones incluyen el título del reporte, el encabezado de página, la sección del cuerpo del reporte, el pie de página y el sumario. Además, con él se pueden crear capas dinámicas basadas en el contenido del reporte, por ejemplo, en dependencia del valor de un campo del reporte, los datos pueden ocultarse o mostrarse en el mismo, incluso pueden agruparse en secciones lógicas, por ejemplo, si se crea un reporte de carros, es posible agrupar los datos por modelo, año, o una combinación de estas o cualquier otra pieza de datos mostrada en el reporte. Agrupar datos proporciona un mejor control del diseño del reporte. La definición de datos agrupados puede usarse además para calcular valores subtotales basados en el subconjunto de datos agrupados en el reporte, además pueden ser usados como definiciones para los gráficos.

Puede presentar los datos de forma textual o gráficamente

JasperReports provee la habilidad de mostrar los datos del reporte textualmente o gráficamente haciendo uso de los gráficos. Permite usar expresiones para generar reportes que muestren datos de forma dinámica, lo cual significa que los datos que no son pasados directamente al reporte o almacenados en alguna parte, son calculados a partir de los datos contenidos en el origen de datos o en los parámetros del reporte.

Los desarrolladores pueden proporcionar datos de múltiples maneras

JasperReports permite a los desarrolladores pasar los datos al reporte por parámetros, los cuales pueden ser instancias de cualquier clase de Java o definida en la aplicación. Los datos pueden ser pasados además usando clases especiales llamadas datasources (origen de datos). Los parámetros y los datasources se pueden combinar para lograr una mayor flexibilidad.

Acepta datos de múltiples fuentes

JasperReports genera reportes usando cualquier sistema de bases de datos relacional soportado por JDBC. Sin embargo, no está limitado a las bases de datos solamente. Se pueden generar reportes desde un gran número de datasources, incluyendo archivos XML, Plain Old Java Objects (POJOs), cualquier clase que implemente la interfaz java.util.Map y cualquier clase que implemente la interfaz javax.swing.TableModel.

Puede generar marcas de agua

JasperReports es capaz de generar imágenes o texto de fondo en los reportes que genere. Estas imágenes de fondo pueden servir como una especie de marca de agua para los reportes. Una marca de agua es como una imagen secundaria que se establece debajo de las imágenes principales. Las marcas de agua pueden utilizarse para marcar los reportes o con propósitos de seguridad, pues hacen difícil la posibilidad de falsificar informes. Todas las páginas del reporte tienen la misma marca de agua, la cual les da un aspecto coherente a los mismos.

Puede generar informes integrados

Otra de las funcionalidades de JasperReports es que permite crear informes integrados, o reportes dentro de reportes. Los informes integrados simplifican el diseño del reporte significativamente, permitiendo extraer las secciones complejas de los reportes a reportes separados e incorporar los mismos dentro del reporte maestro.

Es capaz de generar los reportes en múltiples formatos

Los reportes generados con JasperReports pueden ser exportados a un gran número de formatos, incluyendo PDF (Formato de Documento Portable), XLS (Excel), RTF (Formato de Texto Enriquecido, el cual puede ser leído y editado por la mayoría de los procesadores de Word, incluyendo , pero no limitado a Microsoft Word, OpenOffice.org Writer, StarOffice Writer y WordPerfect), HTML (Lenguaje de Marcado de HiperTexto), XML (Lenguaje de Marcado Extensible), CSV (Valores Separados por Coma) y texto plano.

JasperReports utiliza como diseñador visual de reportes a iReport.

iReport

E s una aplicación **de código abierto** basado en **Java** que permite a diseñadores y desarrolladores en general diseñar y modelar los reportes visualmente. Con **iReport** es posible dar soporte de internacionalización, orígenes de datos con JDBC y JavaBeans a los reportes. Además de la gran facilidad para crear nuevos reportes usando asistentes de configuración y plantillas, con la opción de poder tener una pre visualización de la exportación de los reportes en PDF, HTML, XLS, CSV, TXT y XML. (8)

La herramienta **iReport** es un constructor / diseñador de informes visual, poderoso, intuitivo y fácil de usar para JasperReports escrito en Java. Este instrumento permite que los usuarios corrijan

visualmente informes complejos con cartas, imágenes, informes integrados, etc. iReport está además integrado con JFreeChart, una de la biblioteca gráficas OpenSource más difundida para Java.

Características importantes de **iReport:**

- 100% escrito en JAVA y además OPENSOURCE y gratuito.
- Maneja el 98% de las etiquetas de JasperReports
- Permite diseñar con sus propias herramientas: rectángulos, líneas, elipses, campos de los textfields, cartas, subreports (informes integrados).
- Soporta internacionalización nativamente.
- Browser de la estructura del documento.
- Recopilador y exportador integrados.
- Soporta JDBC.
- Soporta JavaBeans como orígenes de datos (éstos deben implementar la interface JRDataSource).
- Incluye Wizard's (asistentes) para crear automáticamente informes.
- Tiene asistentes para generar los informes integrados.
- Tiene asistentes para las plantillas.
- Facilidad de instalación.

1.2.2 Descripción de la herramienta Itext

IText es una poderosa librería Java de código abierto que soporta la generación de documentos en los formatos HTML, RTF, XML y de documentos PDF. Cuenta con gran variedad de fuentes que pueden ser utilizados en el cuerpo del documento. Además, la estructura de iText permite generar cualquiera de los mencionados tipos de documentos (HTML, RTF, XML y PDF) con el mismo código. iText es una librería que contiene clases para generar documentos PDF con diferentes fuentes, generar tablas, establecer marcas de agua, etc (9).

La mayoría de las herramientas para la generación de documentos dan la posibilidad de generar archivos de extensión PDF. Algunas de estas herramientas son: JasperReports, iReport, CrystalReport y Oracle XML Publisher entre otras.

Así como existen numerosas herramientas para generar documentos PDF, es necesario conocer que prácticamente todas las que están soportadas con el lenguaje Java, utilizan la misma librería, iText. Al

delegarse las responsabilidades en las herramientas de ayuda al diseño de los documentos, por lo general se desconoce las bondades de la utilización directa de esta librería.
Entre las características que se pueden añadir a los documentos destacan:

- Márgenes
- Tablas
- Cabeceras
- Pies de páginas
- Numeración en las páginas
- Distintos tipos de letras
- Cifrado de documentos
- Soporte para varios tamaños de páginas y vistas
- Posibilidad de visualizar imágenes
- Marcas de agua
- Salto automático de página
- Posibilidad de colocar los objetos en cualquier parte del documento
- Páginas multicolumnas
- Firma digital con PKCS7

La ventaja que ofrece iText sobre otras librerías de generación de documentos que existen para la plataforma java, es que la clase que genera los PDF ya está compilada, no hay que hacer una transformación xslt cada vez que se genera el documento, por lo que la generación es mucho más rápida y se consumen muchos menos recursos del sistema.

1.2.3 JFreeChart

JFreeChart es una librería libre para gráficos escrita 100% en java que facilita mostrar gráficos con calidad profesional en las aplicaciones. Entre sus funcionalidades incluye el soporte de un alto rango de tipos de gráficos, un diseño flexible que permite crear los gráficos tanto del lado del cliente como del servidor y soporte a varios formatos de salida (11).

Entre las principales características se encuentran las siguientes:

- Una API bien documentada que da soporte a un amplio rango de tipos de gráfica.
- Diseño flexible y fácil de extender pensado para aplicaciones cliente - servidor
- Puede exportar en ficheros gráficos (JPEG, PNG) y archivos (PDF, EPS, SVG)

- Presentación de datos en Servlet y JSP
- Ofrece numerosas interfaces para el tratamiento de datos en base al gráfico que se desee crear
- Manejo de gráficos: funciones de zoom e impresión.

Se permite la representación gráfica en los siguientes formatos:

- Gráficos de tarta: 2d y 3d
- Gráficos de barras: regulares y en pilas. También ofrece la posibilidad de gráficos 3d
- Gráficos de líneas y áreas
- Gráficos de dispersión
- Gráficos de tiempos
- Gráficos combinados
- Diagramas de Gantt

1.2.4 Apache POI

Existe un grupo de proyectos con el nombre de Apache POI que proporcionan una serie de APIs en java para trabajar con los diferentes formatos de Microsoft Office. Concretamente la API que proporciona el soporte para acceder a los formatos de Microsoft Excel, es decir, a los documentos de hojas de cálculo. Estas APIs están escritas totalmente en java e implementan los estándares de los formatos que soportan xls y xlsx (12).

Apache POI es una API que permite a los programadores crear, modificar y visualizar archivos de Microsoft Office utilizando programas en java. Es una biblioteca de código abierto desarrollada y distribuida por Apache Software Foundation. Es compatible con todas las características básicas de las bibliotecas de Excel y como principales características presenta la presentación y extracción de textos.

El lenguaje de programación usado es Java, y al estar la librería escrita completamente en dicho lenguaje, hace que la solución sea multiplataforma. Apache POI cuenta con una documentación en línea completa y con gran cantidad de ejemplos que muestran cómo utilizar la mayoría de las funcionalidades que ofrece. Esta alternativa lleva usándose en diferentes proyectos tanto libres como privativos desde hace varios años, esto unido a la publicación frecuente de nuevas versiones y al soporte proporcionado por Apache y su comunidad da como resultado una solución con una actividad, soporte y madurez muy altas (12).

1.2.5 BIRT (Business Intelligence and Reporting Tools)

BIRT está basado en Eclipse, sistema de informes de código libre para aplicaciones web, especialmente aquellas basadas en java y JEE. BIRT provee un entorno gráfico de diseño de informes, empaquetado como un plug-in para Eclipse o como un diseñador de informes independiente. Posee un API de tiempo de ejecución, para integrar informes en la aplicación. Tiene dos componentes principales: un diseñador de informes basado en Eclipse y un componente en tiempo de ejecución que se puede añadir a un servidor de aplicaciones. El proyecto BIRT también incluye un motor de gráficos que es totalmente integrado en el diseño y se puede utilizar independientemente para integrar gráficos en una aplicación.

BIRT puede incluir gran variedad de reportes en la aplicación, genera reportes que cumplen con las normas de calidad impuestas. Puede usarse en cualquier plataforma donde eclipse corra. Posee variados formatos de salida como HTML, PDF, WORD y XLS. Los diseños de informes BIRT se hacen en XML y pueden acceder a ciertos números de fuentes de datos diferentes incluyendo base de datos SQL, JDO datastores, JFire Scripting Objects, POJOS y servicios Web. (12)

1.3 Selección de las herramientas de generación de reportes

En la bibliografía consultada se ha identificado que todas las herramientas descritas son potentes herramientas en el desarrollo de aplicaciones para generar reportes. Sin embargo, es de vital importancia tener en cuenta los formatos para generar los reportes en los componentes (pdf, xls) y que la principal responsabilidad de los componentes es la de dibujar los datos del sistema mediante una plantilla que sirva como estándar, sin que ello implique crear un reporte para cada caso independiente y sin realizar consultas a base datos, ni manejar estructura de datos complejas. Por estas razones se seleccionó a iText para el desarrollo del componente encargado de exportar la información en formato pdf. Esta librería es una de las más utilizadas para crear documentos pdf, además es la librería base que utilizan otras para generar reportes en pdf. Apache POI se seleccionó para el desarrollo del componente encargado de exportar los reportes en formato xls debido a sus prestaciones para el trabajo con formatos de Microsoft Office y la amplia documentación y foros existentes en internet. Para generar los gráficos en los componentes se seleccionó la librería de JFreeChart la cual proporciona una gran variedad de gráficos profesionales y varios formatos de salida.

1.4 Herramientas de desarrollo y modelado

A continuación, se describen las herramientas restantes utilizadas para el desarrollo del componente reporte. Estas tecnologías fueron seleccionadas de acuerdo a las ventajas que proporcionan y su amplia utilización en herramientas de esta índole.

1.4.1 Lenguaje de Modelado UML

El Lenguaje de Modelado Unificado (UML) es el lenguaje estándar para escribir diseños de software. El UML puede usarse para visualizar, especificar, construir y documentar los artefactos de un sistema de software intensivo (13). UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de base de datos y componentes de software reutilizables. Se puede usar en gran variedad de formas para soportar una metodología de desarrollo de software (tal como el Proceso Unificado Rational) pero no especifica en sí mismo qué metodología o proceso usar (14).

Existen varis aplicaciones para el modelado UML, son las denominadas herramientas CASE. Dichas herramientas son la aplicación de tecnologías informáticas a las actividades, técnicas y metodológicas propias del desarrollo. Su objetivo es acelerar el proceso para el que han sido diseñadas, automatizar o apoyar una o más fases del ciclo de vida del desarrollo del software. La principal ventaja de la utilización de estas herramientas es la mejora de la calidad de los desarrollos realizados y el aumento de la productividad.

La herramienta propuesta para el modelado en esta investigación es **Visual Paradigm 8.0** (VP), la cual es muy eficaz para visualizar y diseñar elementos de software. VP está orientado a la creación de diseños utilizando el paradigma de programación orientada a objetos. Además, ayuda a una rápida construcción de aplicaciones con calidad y a un menor costo. Permite modelar una gran variedad de diagramas, código inverso y generar documentación. Provee soporte para la generación de código, tiene soporte para diversos IDE de desarrollo, así como la posibilidad de realizarse la ingeniería inversa para aplicaciones realizadas en Java y otros lenguajes (16).

1.4.2 IDE de desarrollo Eclipse

Eclipse es un entorno de desarrollo de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para el desarrollo de aplicaciones. Fue desarrollado originalmente por IBM (International Business Machines). Actualmente eclipse es desarrollado por la

Fundación de Eclipse, una organización independiente sin ánimos de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. La versión del eclipse utilizada para el desarrollo de este proyecto es Eclipse Juno Service Release 2. Entre las características que posee esta herramienta podemos destacar las siguientes (17):

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración Con Ant
- Asistentes (winzards): para la creación de proyectos, clases, tests, entre otras
- Refactorización

1.4.3 Lenguaje de programación Java

Java es un lenguaje de programación orientado a objeto y de uso general, su sintaxis y semántica son suficientemente completas para poder abarcar programas de todo tipo. Es una de las plataformas que cuenta con mayor acogida para la construcción de aplicaciones multicapas. Su arquitectura ofrece conceptos básicos de componentes, herramientas y ambientes de desarrollo que permitan generar programas o páginas dinámicas desplegables para los distintos ámbitos de programación. Su principal característica es la capacidad de ejecutarse en cualquier máquina y sobre cualquier sistema operativo o arquitectura, manteniendo las facilidades básicas del lenguaje (3).

1.5 Metodología de desarrollo RUP

El Proceso Unificado de Desarrollo de Software (RUP) forma parte de las llamadas metodologías pesadas. Esta metodología permite una forma disciplinada de asignar tareas y responsabilidades de acuerdo a quién hace, qué, cuándo y cómo. Utiliza el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) como lenguaje de modelado, el cual está orientado a objeto y genera una gran documentación. Está caracterizado por tres elementos fundamentales que son: dirigido por caso de uso, iterativo e incremental y centrado en la arquitectura. (17)

- Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

- Iterativo e incremental: Cada fase se desarrolla en iteraciones. Se divide el trabajo en partes pequeñas o mini-proyectos, donde cada uno es una iteración que resulta en un incremento.
- Centrado en la arquitectura: La arquitectura describe los elementos del modelo que son importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

RUP describe cuatro fases como muestra la Figura 4: Inicio, Elaboración, Construcción y Transición:

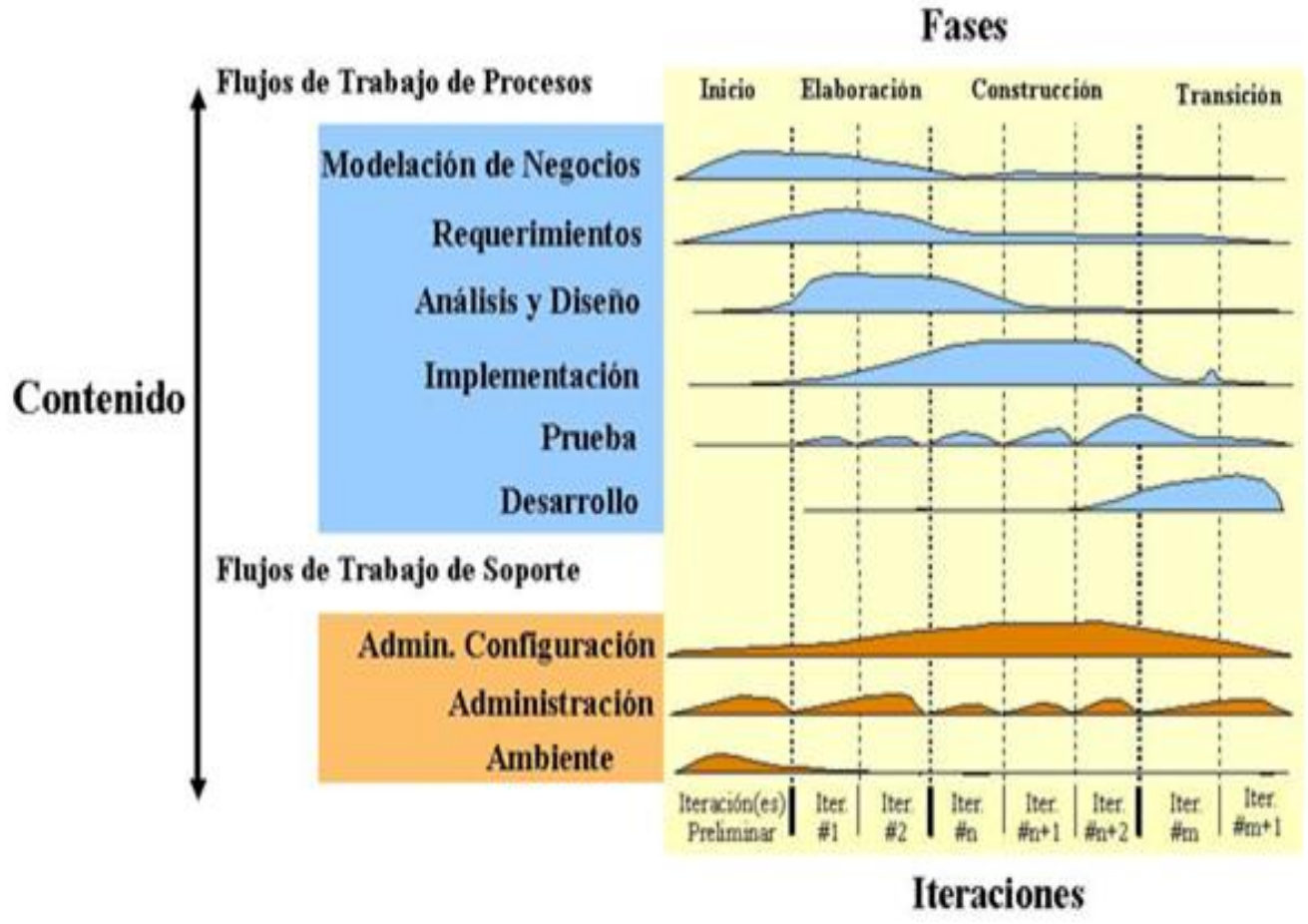


Figura 4: *Metodología RUP*

Los objetivos de cada uno de las fases son los siguientes:

- **Inicio:** Describir el negocio de delimitar el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- **Elaboración:** Definir la arquitectura del sistema y obtener una aplicación ejecutable que responde a los casos de uso que la comprometen.

- **Construcción:** Llevar a cabo la construcción del producto por medio de una serie de iteraciones en las cuales se seleccionan algunos casos de usos. Definir su análisis y diseño y proceder a su implementación y pruebas. En esta fase se realiza una pequeña cascada para cada ciclo, se realizan tantas iteraciones hasta que se termine la nueva implementación del producto.
- **Transición:** Garantizar que se tiene un producto preparado para la entrega al usuario.

En RUP se han agrupado las actividades en grupos lógicos definiéndose una serie de flujos de trabajo. Algunos de estos flujos son:

- **Modelamiento de negocio:** Describe los procesos del negocio, identificando quiénes participan y las actividades que requieren automatizar.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba:** Busca los defectos a lo largo del ciclo de vida del producto.
- **Despliegue:** Produce liberaciones del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

Conclusiones parciales

En el presente capítulo se realizó el estudio de la arquitectura ABOS con el objetivo de conocer el funcionamiento de los componentes en la arquitectura. Se analizaron los conceptos principales asociados a la investigación para seleccionar las herramientas, las librerías y la metodología de desarrollo adecuada para la construcción del producto final.

A raíz del estudio de la arquitectura se llegó a la conclusión de que la aplicación en cuestión seria desarrollada como bundle capaz de integrarse en la arquitectura ABOS. Después de estudiada la bibliografía sobre las herramientas de generación de reporte existentes, se decidió utilizar la librería Itext para el desarrollo de los reportes en formatos pdf, Apache POI para el desarrollo de los reportes en formato xls y la librería de JFreeChart para integrar las gráficas estadísticas en los reportes. Se

realizó también la caracterización de las herramientas para el desarrollo de la aplicación y la metodología de desarrollo de software RUP como metodología rectora del proceso de desarrollo.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN DEL COMPONENTE REPORTE

Introducción

En el presente capítulo se realiza una propuesta de solución para el componente generación de reportes a partir del problema planteado. En el mismo, se exponen los requisitos funcionales y no funcionales previos para dar una correcta solución a las necesidades del cliente. Se determinan los casos de uso correspondientes y se describen las principales funcionalidades del componente. Se presentan también, todos los artefactos generados en el transcurso de los flujos de trabajos de diseño e implementación del componente mediante los diagramas correspondiente a cada uno. Igualmente, se especifican los patrones de diseños y estándares de codificación utilizados en la concepción de la solución.

2.1 Propuesta de solución

Una vez realizado el análisis del estado del arte, elegidas las herramientas y la metodología a utilizar; se está en condiciones de plantear una solución al problema descrito.

La solución que se propone constituye un componente o bundle que se pueda insertar en la arquitectura ABOS utilizada por el proyecto ABCD 3.0. Dicho componente incluye las funcionalidades de generar y configurar los reportes dentro de la arquitectura. El componente para su funcionamiento se divide en tres componentes: un componente principal, uno para la generación de reportes en formatos pdf y otro para generar los reportes en formato xls. Se espera como resultado un componente flexible a cambios que puedan surgir y que facilite en gran escala el trabajo de aquellos que lo utilicen.

2.2 Modelo de dominio

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos del software. Utilizando la notación UML, un modelo de domino se representa con un conjunto de diagramas de clases en los que no se define ninguna operación (18). Puede mostrar:

- Objetos del dominio o clases conceptuales
- Asociaciones entre clases conceptuales
- Atributos de las clases conceptuales

Con el objetivo de tener una visión clara de las funcionalidades que debe tener el componente y la estructura con que debe contar se realizó el siguiente modelo de dominio:

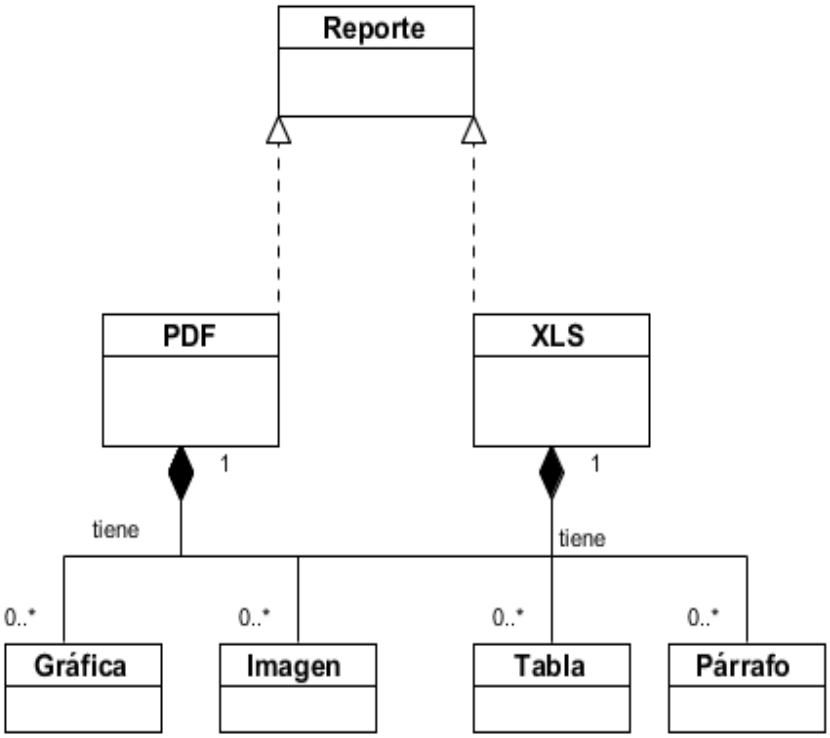


Figura 5: Modelo del dominio

En la figura 5 se representa a través de las clases conceptuales los principales conceptos que se manejan para alcanzar los objetivos trazados para este proyecto. Estos son:

- **Reporte:** Documento que se crea con la información requerida
- **PDF:** Reporte en extensión de pdf
- **XLS:** Reporte en extensión xls
- **(Tabla, Imagen, Gráfica, párrafo):** Elementos que conforman un reporte ya sea en formato pdf o xls

2.3 Especificación de los requisitos de software

La especificación de requisitos es un insumo fundamental en el desarrollo de software. Describen los servicios que ha de ofrecer un sistema y las restricciones asociadas a su funcionamiento. Es el área del conocimiento que explica cómo lograr la obtención, análisis, especificación y validación de los requisitos de software (14).

2.3.1 Requisitos funcionales

La especificación de requisitos es un insumo fundamental en el desarrollo de software. Describen los servicios que ha de ofrecer un sistema y las restricciones asociadas a su funcionamiento. Es el área

del conocimiento que explica cómo lograr la obtención, análisis, especificación y validación de los requisitos de software (14).

- RF1 Crear parámetros de reporte
- RF2 Configurar reporte
- RF3 Exportar reporte en formato pdf
- RF4 Exportar reporte en formato xls

Tabla 1: *Requisitos funcionales y su descripción*

Requisitos	**Descripción**
1 Crear parámetros de reporte	El sistema crea los parámetros (párrafos, tablas, imágenes, gráficas) pasándole los datos correspondiente a cada uno.
2 Configurar reporte	El sistema configura los reportes de acuerdo a los parámetros, estilo del documento y el formato a exportar el reporte.
3 Exportar reporte en formato pdf	El sistema permite exportar los reportes en formato de pdf
4 Exportar el reporte en formato xls	El sistema permite exportar los reportes en formato de xls

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempos, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican en el sistema en su totalidad. Pueden especificar también la utilización de una herramienta en particular, un lenguaje de programación o un método de desarrollo (19).

RNF1 Requerimiento de hardware: Para el correcto funcionamiento del sistema se debe tener:

- Memoria RAM: 2gb como mínimo
- Procesador: Dual Core o superior

El sistema donde se integre el componente debe cumplir como mínimo con los recursos antes planteados. De lo contrario no se garantiza el correcto funcionamiento del componente.

RNF 1 Usabilidad: El sistema será usado por personal con conocimientos previos en programación orientada a objeto y en el lenguaje java.

RNF 2 Usabilidad: El sistema exporta los reportes en Excel en versiones de Microsoft Office (97-2008). En las versiones superiores no se asegura su correcta visualización.

RNF 3 Requerimientos de software:

- Sistema operativo linux o Windows donde corra la máquina virtual de Java 7

El sistema no garantiza su funcionalidad sobre versiones anteriores de la máquina virtual de Java.

RNF4: Implementación: Se utiliza para la construcción del sistema el lenguaje Java.

2.4 Definición de los Actores

Un actor es cualquier cosa con comportamiento, incluyendo el propio sistema que se está estudiando cuando solicita los servicios de otro sistema. Los actores no son solamente roles que juegan personas, sino también organizaciones, software y maquinas (15).

En el componente interactúa un solo actor el cual se define en la siguiente tabla:

Tabla 2: *Descripción del actor*

Actor	**Objetivo**
Sistema	Es el encargado de solicitar los servicios de generar reporte en formato pdf y xls respectivamente.

2.5 Definición de los casos de usos

A partir del análisis de los requisitos funcionales del sistema se definieron los siguientes casos de usos:

- Configurar reporte
- Generar reportes en formato pdf
- Generar reporte en formato xls

2.5.1 Diagrama de casos de uso

Un diagrama de caso de uso explica gráficamente un conjunto de casos de uso de un sistema, los actores y la relación entre estos y los casos de uso. El diagrama tiene como objetivo ofrecer una clase de diagrama contextual que permita conocer rápidamente los actores externos de un sistema y las formas básicas en que los utilizan (14).

La figura 6 muestra el diagrama de casos de uso correspondiente al componente.

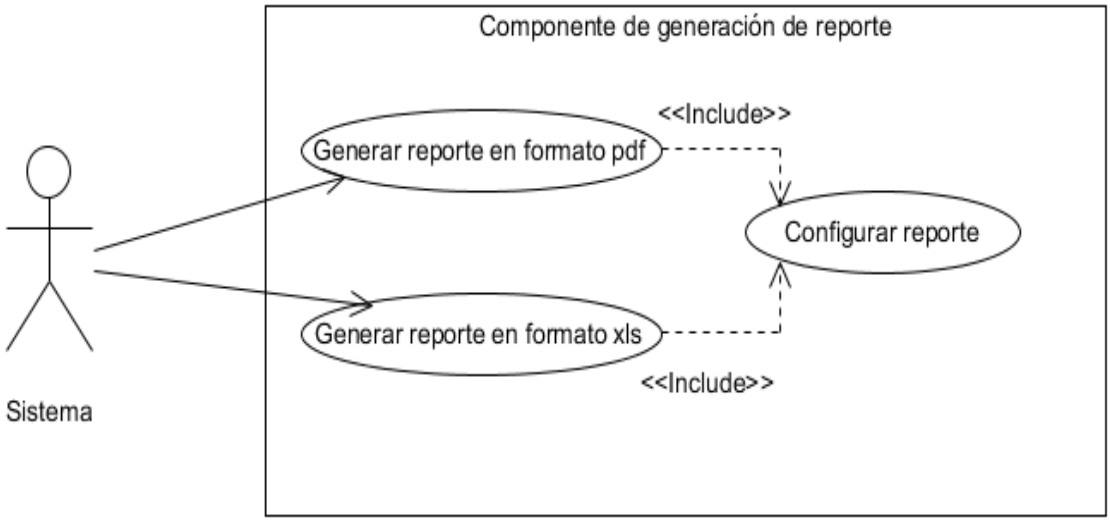


Figura 6: *Diagrama de casos de uso*

Las siguientes tablas muestran la descripción de los casos de uso identificados en el sistema.

2.5.2 Descripción de los casos de uso

Caso de uso Generar reporte en formato pdf:

Tabla 3: *Descripción del caso de uso exportar reporte en formato pdf*

Objetivo	Exportar un reporte en formato pdf
Actores	Sistema
Resumen	El caso de uso se inicia cuando el actor solicita el servicio de exportar reporte a pdf
Complejidad	Media

Prioridad	Media	
Precondiciones	El actor posee información para generar un reporte	
Postcondiciones	Se crea un documento con la información proporcionada por el actor en formato pdf.	
Flujo de eventos		
Flujo básico Generar reporte en formato pdf		
Actor		Sistema
1	Solicita el servicio de generar reporte en formato pdf	
2		Realiza un llamado al caso de uso configurar reporte donde se realiza la configuración necesaria para generar el reporte
3		Toma los datos de la configuración realizada
4		Crea un documento con cabecera, pie de página y aplicando el estilo seleccionado en la configuración
5		Añade al documento los elementos listados (imagen, tabla, párrafo, gráfica) en el orden que aparezca en la configuración
6		Devuelve el documento
7		Termina el caso de uso
Flujos alternos		
3.1 En el caso que no se realice la configuración del servicio		
Actor		Sistema
		No procede a generar el reporte

Caso de uso configurar reporte:

Tabla 4: *Descripción del caso de uso configurar reporte*

Objetivo	Configurar los reportes	
Actores	Sistema	
Resumen	El caso de uso se inicia cuando el sistema envía la configuración necesaria para generar el reporte en el formato que desee	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El actor ha solicitado el servicio de generar reporte	
Postcondiciones	La configuración completada	
Flujo de eventos		
Flujo básico configurar reporte		
Actor		Sistema
1	Solicita los elementos disponibles para agregar al reporte	
		Permite crear elementos tipo gráfica, tabla, imagen y párrafo con los siguientes parámetros: **Tabla:** Listado de columnas, datos de la tabla, nombre de la tabla **Imagen:** Nombre de la imagen y la dirección donde se encuentra la imagen **Párrafo:** Título del párrafo y el texto **Gráfica de barra:** Título de la gráfica, datos correspondientes a las abscisas y ordenadas y el listado de los datos a graficar

		Gráfica de línea: Título de la gráfica, datos de abscisas y ordenadas y el listado de los datos a graficar **Gráfica de pastel:** Título de la gráfica y el listado de los datos a graficar
2	Crea los elementos a incluir en el reporte a partir de la información que posee y los parámetros que brinda el sistema	
3	Lista los elementos en el orden deseado	
4	Solicita el estilo del documento	
5		Permite seleccionar el estilo que tendrá el reporte.
6	Selecciona el estilo del reporte	
7	Agrega la lista de los elementos y el estilo a la configuración	
		Termina el caso de uso
Flujos alternos		
2.1 En el caso de que no corresponda la información del actor con los parámetros que brinda el sistema para crear los elementos del reporte		
Actor		Sistema

No puede crear los elementos que no corresponda la información con los parámetros que brinda el sistema	

Caso de uso Generar reporte en formato xls:

Tabla 5: *Descripción del caso de uso generar reporte en formato xls*

Objetivo	Exportar un reporte en formato xls	
Actores	Sistema	
Resumen	El caso de uso se inicia cuando el actor solicita el servicio de exportar reporte a xls	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El actor posee información para generar un reporte	
Postcondiciones	Se crea un documento con la información proporcionada por el actor en formato xls	
Flujo de eventos		
Flujo básico generar reporte en formato xls		
Actor		Sistema
1	Solicita el servicio de generar reporte en formato xls	
2		Realiza un llamado al caso de uso configurar reporte donde se realiza la configuración necesaria para generar el reporte
3		Toma los datos de la configuración realizada
4		Crea un documento con el logo del sistema y aplica el estilo seleccionado en la configuración

5		Añade al documento los elementos listados (imagen , tabla, párrafo, gráfica) en el orden que aparezcan en la configuración
6		Devuelve el documento
7		Termina el caso de uso
Flujos alternos		
3.1 En el caso que no se realice la configuración del servicio		
Actor		Sistema
		No procede a generar el reporte

2.6 Diseño del componente

El diagrama del modelo de dominio visto anteriormente muestra cómo se relacionan los objetos principales del componente para construir un reporte en el formato deseado. Sin embargo, esta estructura es una visión del mundo real, no sería óptima a la hora de implementar la aplicación ya que para cada formato habría que crear los parámetros correspondientes de forma independiente y realizar prácticamente la misma implementación. Para solventar esto se creó el siguiente diseño del componente como muestra la Figura 7.

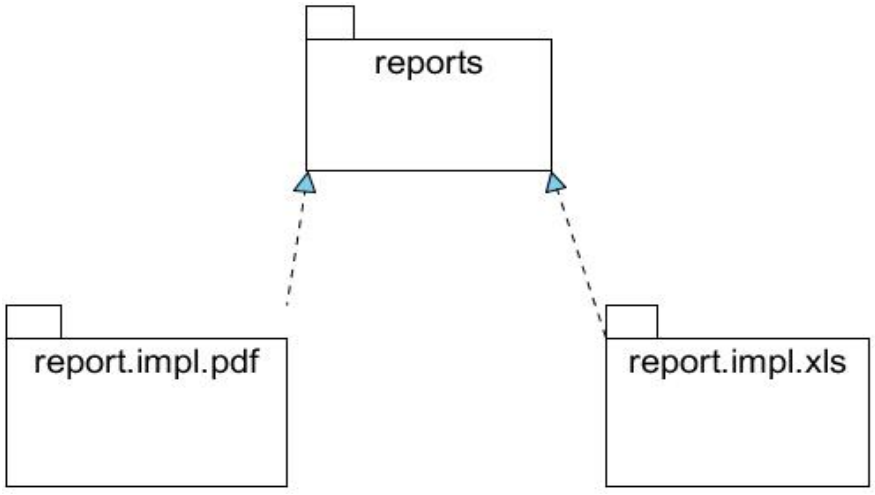


Figura 7: *Diseño del componente*

Descripción:

La solución está estructurada en tres componentes o bundles (reports, report.impl.pdf, report.impl.xls) que dependen entre sí para su funcionamiento. El componente **reports** contiene las clases principales e interfaces encargadas de estructurar el funcionamiento del sistema en general, así como las clases correspondiente a los parámetros que conforman los reportes. En los componentes de **report.impl.pdf** y **report.impl.xls** es donde se implementan los reportes para formato pdf y xls respectivamente, contienen la implementación de las clases planteadas en el componente Reports indistintamente para cada formato. El objetivo de esta estructura es separar el diseño de la API y los datos para generar los reportes de la implementación, haciendo así que el componente sea extensible para otras implementaciones que se deseen realizar sobre la misma API.

2.7 Diagrama de Paquetes

Un diagrama de paquete muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Incluyen conceptos fuertemente relacionados, esto sirve de ayuda para mejorar la comprensión y para abordar trabajos de análisis en paralelo (19).
El diagrama de paquete está constituido por dos tipos de elementos:

- **Paquetes:** permiten dividir un modelo en partes manejables mediante la agrupación de elementos que pueden ser casos de uso, clases o componentes. Pueden anidar otros paquetes dentro de sí.
- **Dependencias:** Indican que un elemento de un paquete requiere a otro de otro paquete distinto

La Figura 8 muestra el diagrama de paquetes del componente:

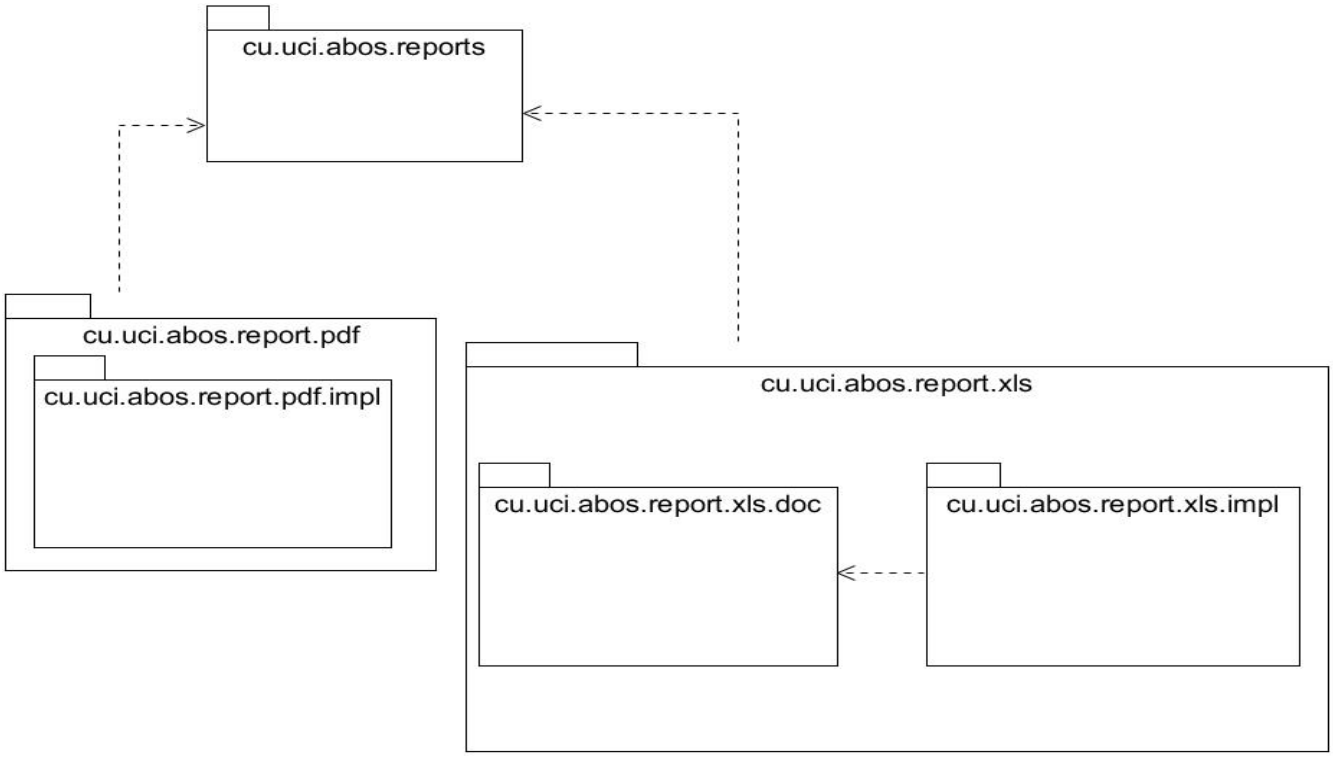


Figura 8: D*iagrama de paquetes*

Descripción:

El componente se divide en tres paquetes que representan los componentes o bundles del diseño antes mostrado. El paquete de cu.uci.abos.reports es el paquete principal del cual dependen los paquetes cu.uci.abos.report.pdf y cu.uci.abos.report.xls para implementar el conjunto de clases que conforman los componentes. El paquete cu.uci.abos.report.pdf está constituido por el paquete cu.uci.abos.report.pdf.impl, el mismo contiene las clases que implementan este componente. El paquete de cu.uci.abos.report.xls está compuesto por dos paquetes, el paquete cu.uci.abos.report.xls.doc que contiene las clases necesarias para definir los estilos de los reportes y el paquete cu.uci.abos.report.xls.impl donde se implementan las clases que conforman este componente.

2.8 Diagramas de clases

El diagrama de clases describe gráficamente las especificaciones de las clases de software en una aplicación. Definen de forma correcta las relaciones de dependencia, generalización y asociación de clases que constituyen el sistema. A diferencia del modelo de dominio, un diagrama de este tipo contiene las definiciones de las entidades de software en vez de conceptos del mundo real (15).

2.8.1 Diagrama de clases del componente reports

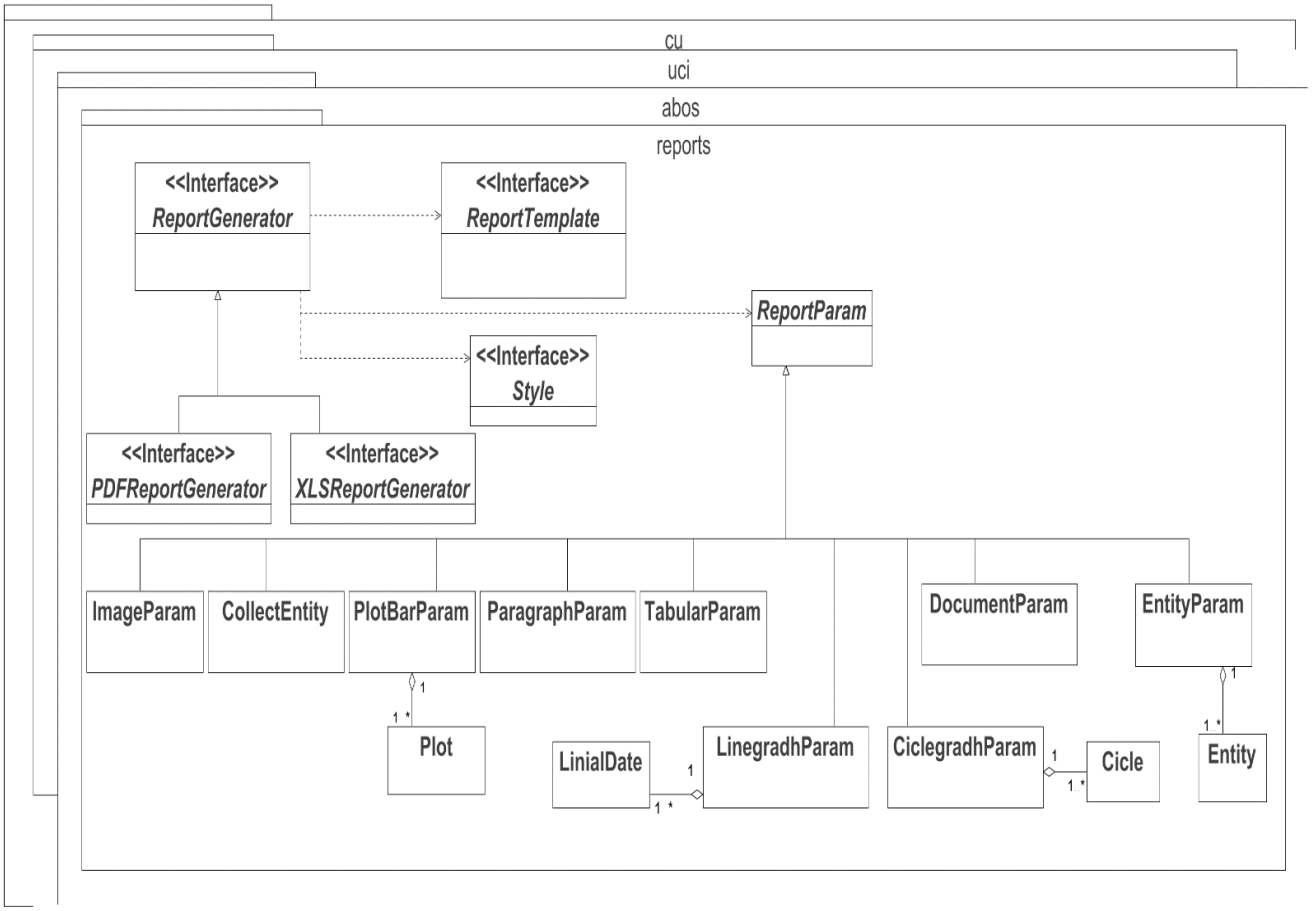


Figura 9: *Diagrama de clases del componente reports*

2.8.2 Diagrama de clases del componente report.pdf

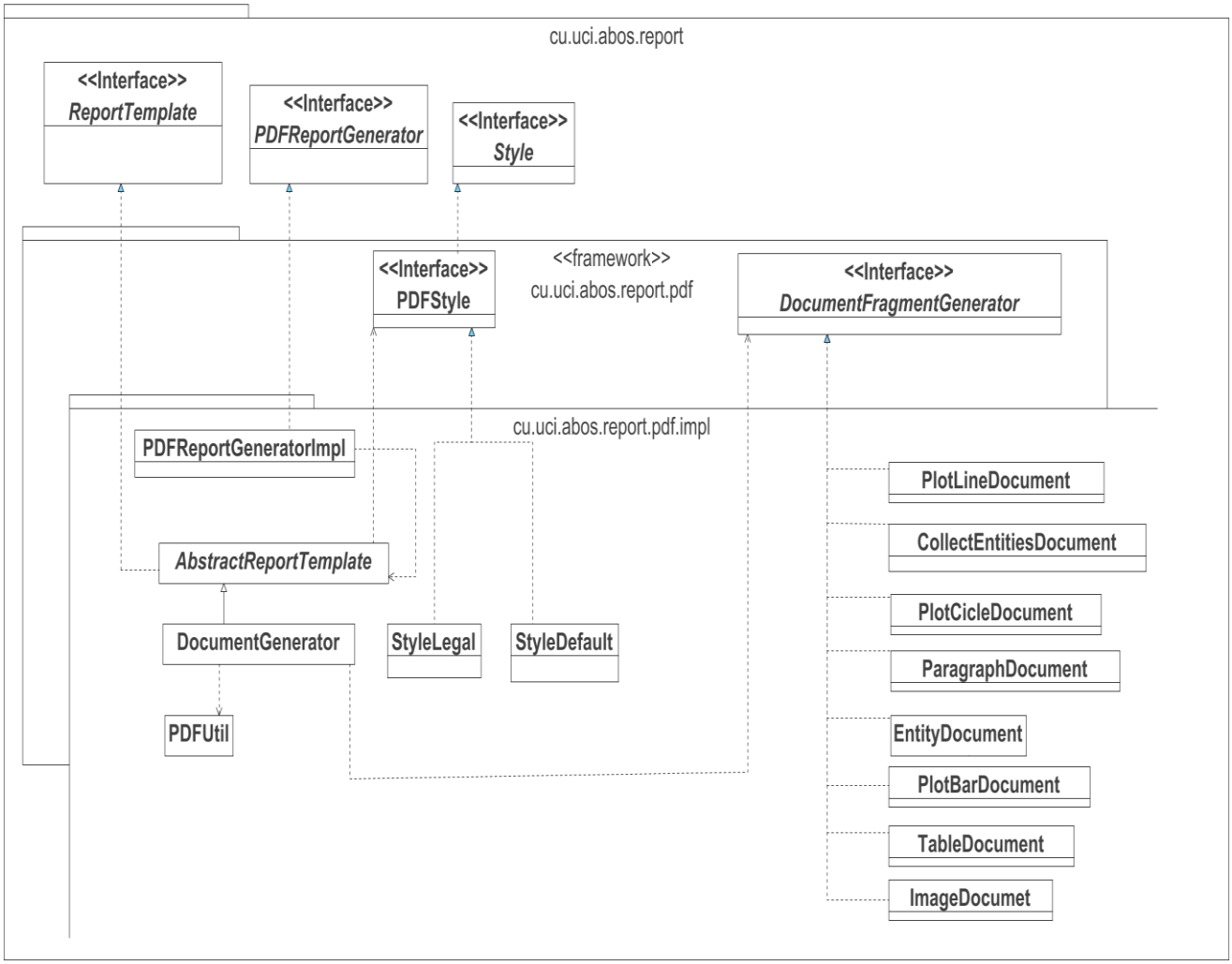


Figura 10: *Diagrama de clases del componente report.pdf*

2.8.3 Diagrama de clases del componente report.xls

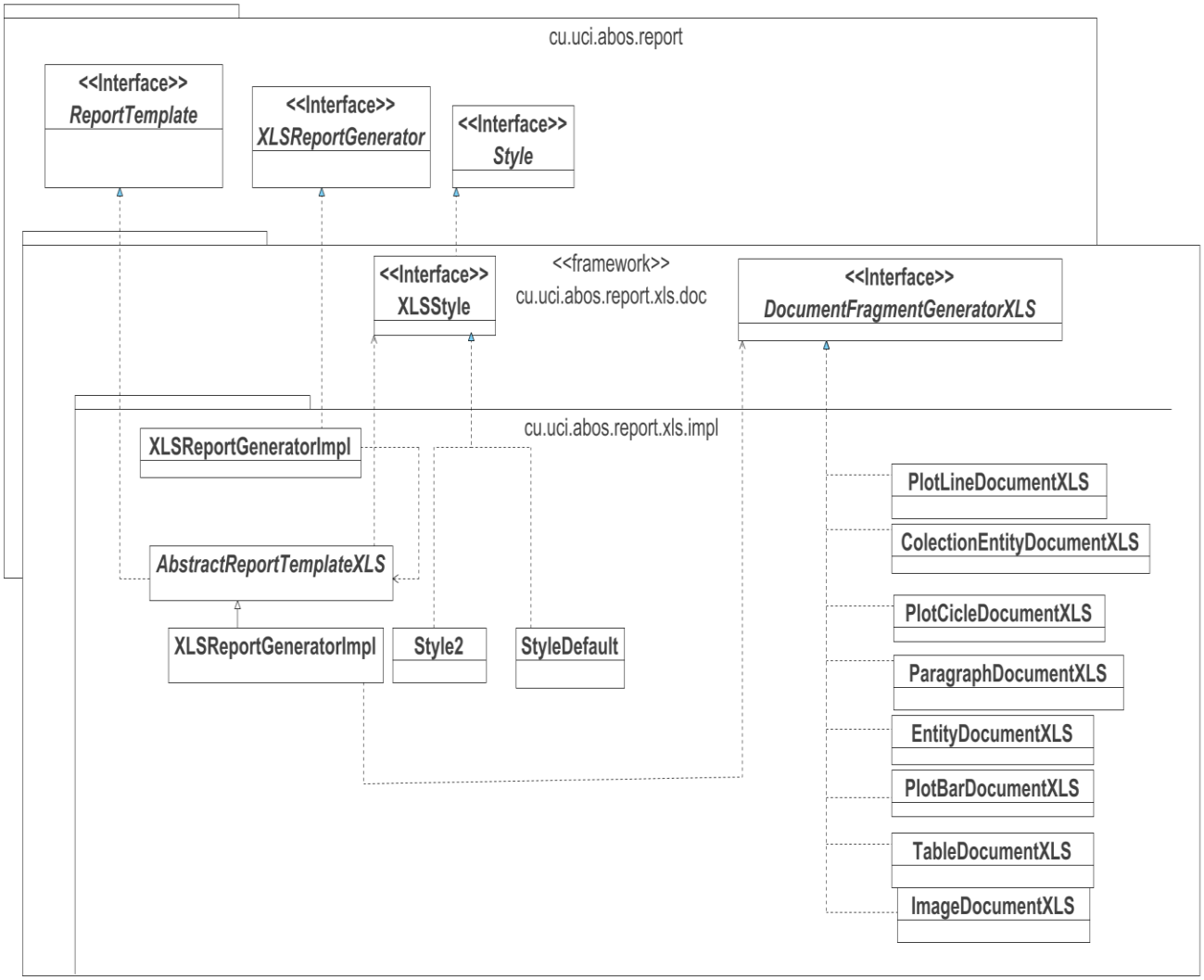


Figura 11: *Diagrama de clases del componente report.xls*

2.9 Diagramas de secuencia del componente

Un diagrama de secuencia es un dibujo que muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas. Los diagramas de secuencia muestran qué objetos se comunica con qué otros objetos y qué mensajes disparan estas comunicaciones (19).

Las siguientes imágenes muestran los diagramas de secuencia por cada caso de uso del sistema:

2.9.1 Diagrama de secuencia del caso de uso generar reportes en formato pdf

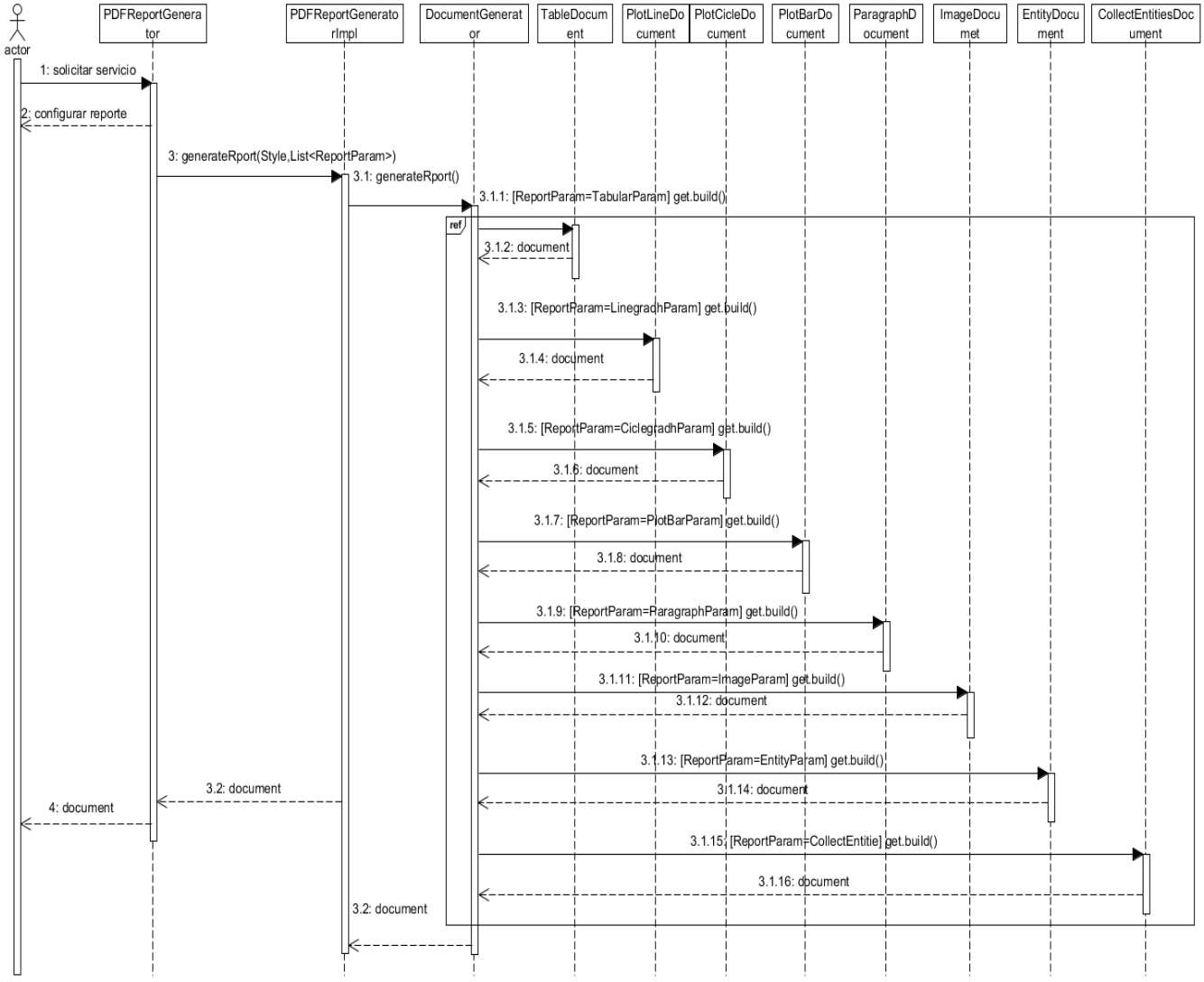


Figura 12: *Diagrama de secuencia generar reporte en formato pdf*

2.9.2 Diagrama de secuencia del caso de uso generar reporte en formato xls

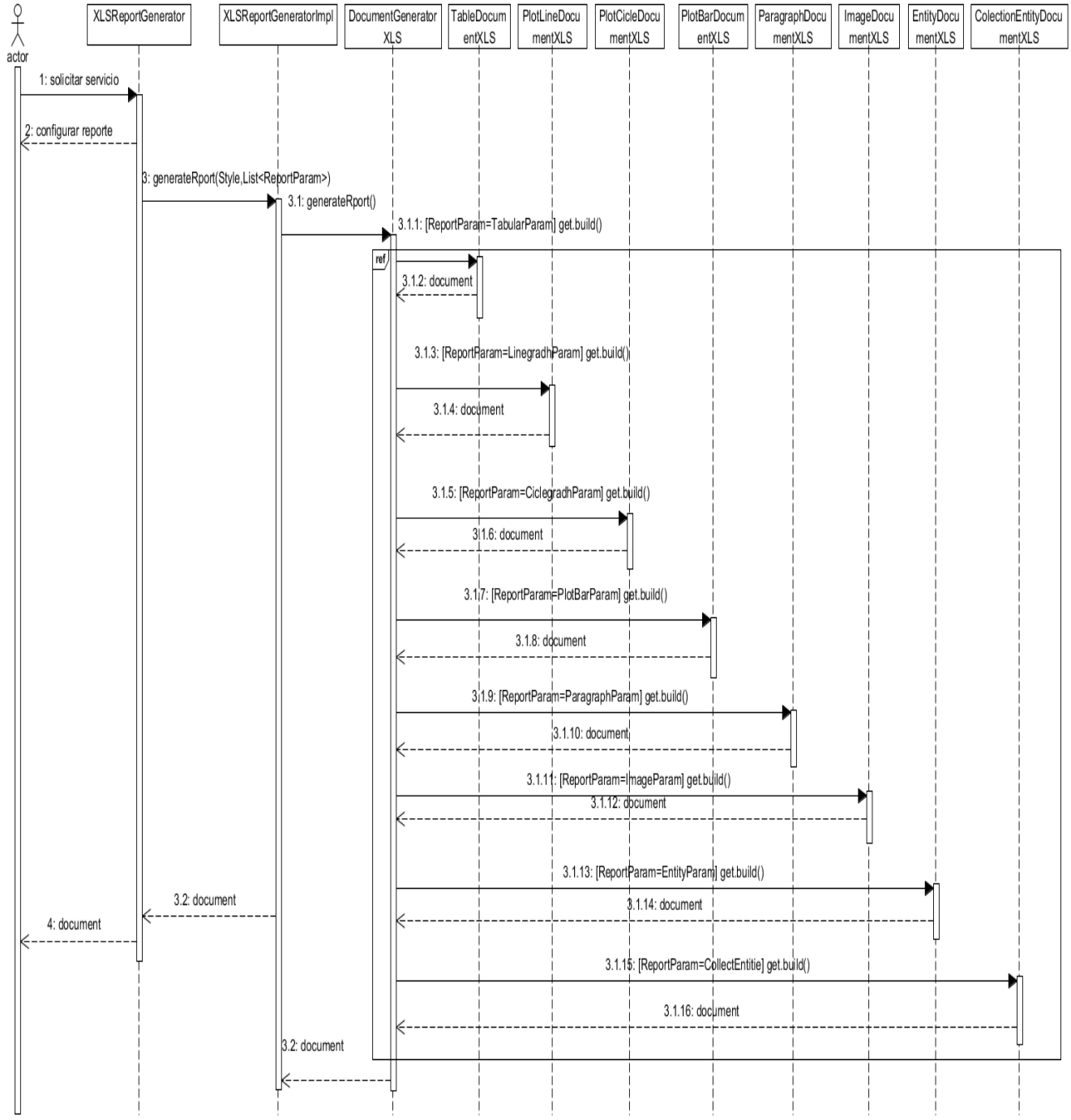


Figura 13: *Diagrama de secuencia generar reporte en formato xls*

2.9.3 Diagrama de secuencia del caso de uso configurar reporte

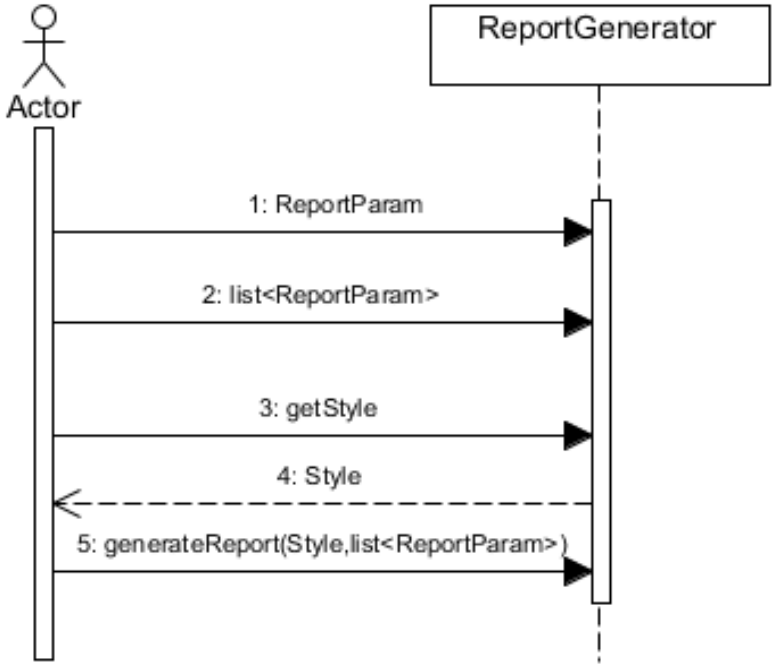


Figura 14: *Diagrama de secuencia configurar reporte*

2.10 Patrones de diseños utilizados

Un patrón de diseño define un esquema de refinamiento de los subsistemas o componentes dentro de un sistema, o las relaciones entre estos. Es una solución recurrente a un problema común observado o descubierto durante el estudio o construcción de numerosas aplicaciones. Su principal objetivo es incrementar la calidad del software en términos de reusabilidad, mantenimiento y extensibilidad (19).

2.10.1 Patrones GRASP

Los patrones GRASP (General Responsibility Asignment Software Patterns, “Patrones de Software para la Asignación General de Responsabilidad”) son patrones generales de software para asignación de responsabilidades.

- **Experto:** Este patrón consiste en asignar una responsabilidad al experto en información, es decir, a la clase que cuenta con la información necesaria para cumplir con la responsabilidad. En el componente un ejemplo de la utilización de este patrón es la clase ParagraphDocumentXLS en el bundle report.xls quien se encarga de agregar un párrafo a un documento.

- **Patrón Creador:** El patrón guía la asignación de responsabilidades relacionada con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Su propósito principal es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Un ejemplo de su utilización en el presente componente es la clase DcumentGenerator en el bundle report.pdf quien crea instancias de las clases especializadas en agregar los elementos al reporte
- **Patrón Bajo Acoplamiento:** Pretende asignar una responsabilidad para mantener el bajo acoplamiento, es decir, el diseño de clases más independientes, que no se relacionen con muchas otras, que reducen el impacto de los cambios, que son más reutilizables y acrecienten la oportunidad de una mayor productividad. Un ejemplo donde se ve el bajo acoplamiento es en el bundle de report.pdf la clase TableDocument.
- **Patrón alta cohesión:** Su objetivo es asignar una responsabilidad, de modo que la cohesión siga siendo alta. Las clases con alta cohesión se caracterizan por tener responsabilidades estrechamente relacionadas y no realizar un trabajo enorme. Una clase con alta cohesión es útil porque es fácil darle mantenimiento, entenderlas y reutilizarlas. El patrón de alta cohesión se refleja en los bundles de report.xls y report.pdf en aquellas clases que construyen un reporte, donde se divide la responsabilidad de crear la plantilla y de insertar los elementos que tendrá el reporte, ejemplo: en el bundle de report.pdf la clase DocumentGenerator crea la plantilla y las clases derivadas de DocumentFranmentGenerator son las que agregan a la plantilla los elementos del reporte.
- **Controlador:** Un Controlador es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema. El Controlador define el método para la operación del sistema. El controlador recibe la solicitud del servicio desde la capa de presentación y coordina su realización, normalmente delegando a otros objetos. Un ejemplo de la utilización de este patrón en el componente son las clases de PDFReportGeneratorImpl y XLSReportGeneratorImpl en los bundles de report.pdf y report.xls.

2.11 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componente. De igual forma describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponible en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (20).

2.11.1 Diagrama de componentes

Un componente representa un modelo físico de los elementos de un modelo que pueden ser relacionados en un diagrama de componentes. Un diagrama muestra varios componentes de un sistema, describiendo sus elementos físicos y sus dependencias. Algunos de los estereotipos estándar de componentes son los siguientes (18):

- **Ejecutable:** Un programa que puede ser ejecutado en un nodo
- **Archivo:** Un fichero que contiene código fuente o datos
- **Librería:** Una librería estática o dinámica
- **Tabla:** Una tabla de base de datos
- **Documento:** Un documento que contenga datos

A continuación, se describe el diagrama de componente para el sistema.

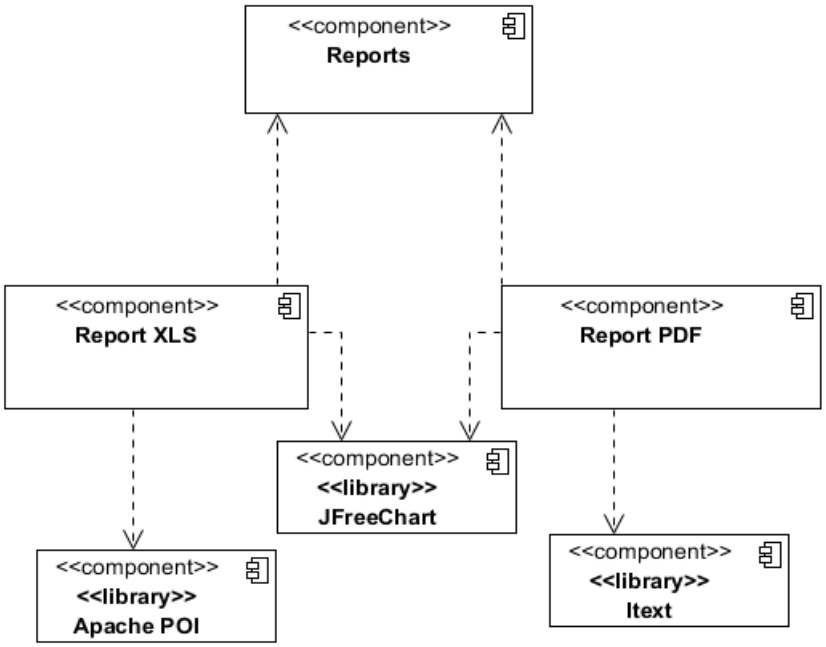


Figura 15: *Diagrama de componentes*

La Figura 15 muestra una estructuración de los componentes del sistema y sus relaciones. A continuación, se describen de manera general cada uno de los componentes.

- **Reports:** El componente Reports está conformado por el conjunto de clases que definen los parámetros, estilos y estructura que conforman un reporte. Es el componente principal del sistema.
- **Report PDF:** Este componente es el encargado de implementar un reporte en formato pdf. Para ello depende del componente Reports quien contiene los parámetros necesarios para formar el reporte. Depende de la librería Itext para crear y modificar el documento del reporte en formato pdf y de la librería JFreeChart para crear y agregar las gráficas al documento.
- **Report XLS:** El componente Reports XLS es el encargado de implementar un reporte en formato xls. Para ello al igual que el componente Report PDF depende de Reports quien contiene los parámetros necesarios para formar un reporte. Depende también de la librería Apache POI para crear y modificar el documento en formato xls y de la librería JFreeChart para crear y agregar las gráficas al documento.
- **Itext:** Esta es la librería que contiene el conjunto de herramientas necesarias para el trabajo con documentos en formato pdf. Es pieza clave en el sistema para el desarrollo de los reportes en dicho formato.
- **Apache POI:** Esta librería permite a los programadores crear, modificar y visualizar archivos de Microsoft Office. Se utiliza en el sistema para crear y modificar los reportes en formato xls.
- **JFreeChart:** Proporciona las vías para crear una gran variedad de gráficas con calidad profesional. Se utiliza en el sistema para crear las gráficas que se integran a los reportes tanto en formato pdf como en xls.

2.12 Estándares de codificación

Con el objetivo de mejorarla calidad y entendimiento del código del componente generación de reporte para la arquitectura ABOS se utilizaron los siguientes estándares de codificación en la implementación:

- Nombres en inglés para las clases y métodos.
- Los nombres de clases comienzan con la primera letra en mayúscula y el resto con minúscula, en caso de ser una palabra compuesta se emplea la notación PascalCasing. Ejemplo: ReportParam, ReportGenerator.
- Espacios de separación dentro del código para que sea entendible.
- Los nombres de los atributos de las clases e instancias con minúscula, en caso de ser un nombre compuesto el inicio de cada palabra compuesta con mayúscula. Ejemplo: reportParam.

- Los nombres de clases, atributo, funciones y objetos acorde con los propósitos de los mismos.

2.13 Diagrama de despliegue

Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal es su diseño. Este modelo representa una correspondencia entre la arquitectura del software y la arquitectura del sistema (18).

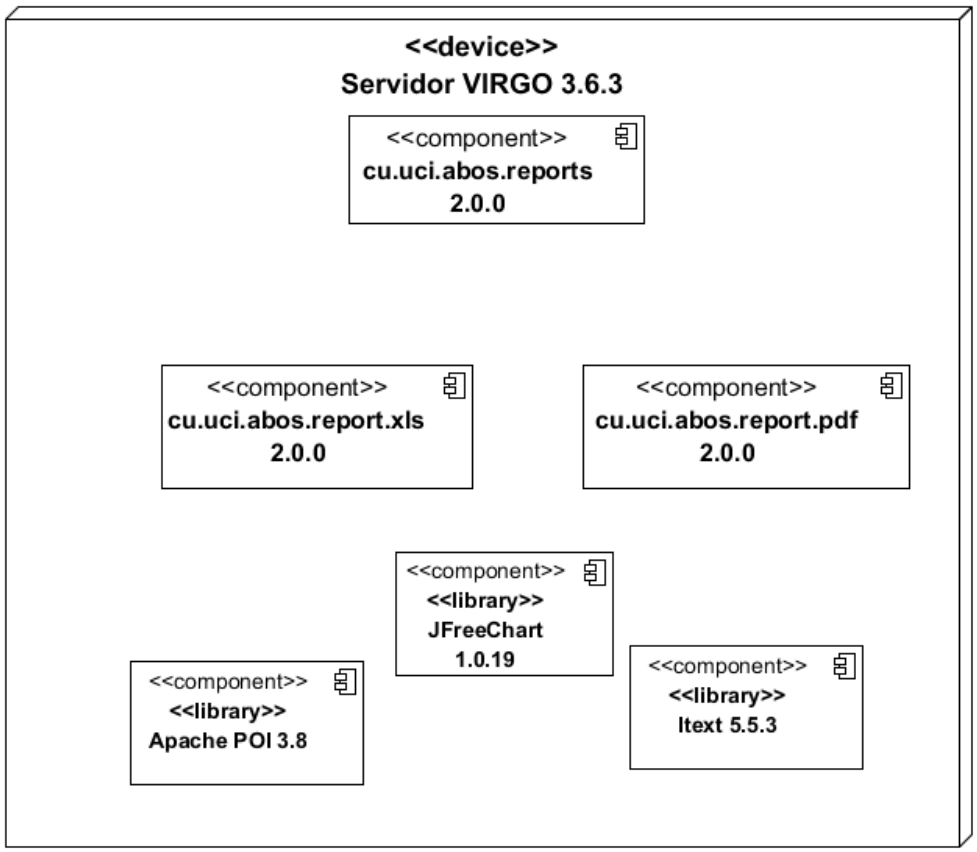


Figura 16: *Diagrama de despliegue*

Descripción:

El componente se despliega en el servidor Virgo en su versión 3.6.3 donde se insertan los tres bundles para la generación de reportes: cu.uci.abos.reports 2.0.0, cu.uci.abos.report.pdf 2.0.0 y cu.uci.abos.xls 2.0.0. Las librerías de JFreeChart Itext y Apache POI se despliegan también en el servidor con el objetivo de cumplir con las dependencias de los bundles para realizar los reportes.

Conclusiones parciales

En el presente capítulo se presentó el diseño e implementación del componente Generación de Reporte mediante los principales artefactos que genera la metodología de desarrollo RUP en el proceso de desarrollo del software. Se realizaron los diagramas de clases correspondiente a cada componente por separado, así como los diagramas de secuencia, paquetes, componentes y de despliegue necesarios para presentar el diseño y la implementación realizada durante el desarrollo del componente. Con el objetivo de mejorar la calidad del software en términos de reusabilidad, mantenimiento y extensibilidad se expusieron los patrones de diseños y estándares de codificación utilizados.

CAPITILO 3: PRUEBAS AL COMPONENTE GENERACIÓN DE REPORTES

Introducción

El presente capítulo completa el proceso de desarrollo del componente en cuestión a partir de la realización de las pruebas necesarias para la verificar la solución. En el mismo se definen las pruebas de software como elemento esencial para garantizar la calidad del componente y la revisión del cumplimiento de los requisitos.

3.1 Pruebas de software

El desarrollo de software implica una serie de actividades de producción donde las posibilidades que aparezcan las fallas humanas son comunes. Los errores pueden darse desde el primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea, así como en los posteriores pasos del diseño y desarrollo. Las pruebas de software son un elemento fundamental para contrarrestar los errores cometidos y garantizar la calidad del mismo, esta constituye una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados. Los resultados son observados y registrados, y la evaluación es hecha de algún aspecto del sistema o componente (14).

3.2 Configuración del entorno de prueba

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probarlo se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso. Por ello se desarrolló un bundle de pruebas con el objetivo de consumir los servicios que generan los componentes. El mismo contiene los datos necesarios para probar los distintos escenarios posibles para todas las funcionalidades del sistema.

3.3 Pruebas de caja negra

Para desarrollar las pruebas al sistema se decidió utilizar el método de caja negra, que se refiere a las pruebas llevadas a cabo sobre la interfaz del software. Es decir, a través de ellas se pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y que la integridad de la información externa se mantiene (14).

Las pruebas de caja negras intentan encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes
- Errores de interfaz
- Errores en estructura de datos o en accesos a base de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Para confeccionar los casos de pruebas de caja negra existen distintas técnicas. Entre las que se encuentran:

- **Partición de equivalencia**: Es una técnica de pruebas de caja negra que divide el campo de entrada de un programa en clases de datos de los que pueden derivar casos de prueba. Esta se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número total de casos de pruebas que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada.
- **Análisis de Valores Límite:** El Análisis de Valores Límites (AVL) es una técnica de diseño de casos de pruebas que complementa la participación equivalente. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de pruebas también para el caso de salida.
- **Métodos basados en grafos:** Empieza creando un grafo de objeto importantes y sus relaciones, y después diseñado una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir los errores.
- **Guiada por casos de pruebas:** verifican las especificaciones funcionales y no consideran la estructura interna del programa. Se realiza sin el conocimiento interno del producto. No validan funciones ocultas por tanto los errores asociados a ellas no serán encontrados.

3.4 Pruebas realizadas

Con el objetivo de comprobar el correcto funcionamiento de las funcionalidades del presente sistema, el software fue integrado a la arquitectura ABOS como bundle y sometido a una serie de pruebas utilizando como medio el bundle de pruebas mencionado anteriormente. Con estas pruebas se pudo corroborar que el sistema responde positivamente a todas las funcionalidades que se desee realizar sobre el mismo. A continuación, se muestran los resultados obtenidos al aplicar las pruebas de caja negras al componente utilizando la técnica de particiones de equivalencia.

3.4.1 Diseño de casos de prueba

CU Exportar reporte en formato pdf

- **Descripción generar:** Este caso de uso le permite al actor generar y exportar un reporte en formato pdf.
- **Condiciones de ejecución:** El actor posee información para generar un reporte
- **Secciones:** Diseño del caso de prueba

Tabla 6: *Diseño de caso de prueba para el caso de uso exportar reporte en formato pdf*

Escenario	Descripción	Parámetros	Estilo	Configurar reporte	Respuesta del sistema	Flujo central
EC 1.1 Generar reporte en formato pdf	Se exporta el reporte construido por el actor en formato pdf.	V Tabla	V Predeterminado	V Realizada	El sistema permite generar un reporte en formato pdf. Le proporciona las vías para realizar la configuración a partir del estilo y los parámetros a incluir en el reporte. Ver CU Configurar reporte	1. El actor solicita al sistema el servicio de generar reporte en formato pdf. 2. El sistema envía la configuración a realizar para generar el reporte. 3. El actor realiza la configuración siguiente: Ver CU Configurar reporte 3.1 Crea los parámetros que tendrá el reporte (imagen, tabla, párrafo, gráfica). 3.2 Lista los parámetros en el orden que desee ser representados en el reporte. 3.3 Selecciona el estilo del reporte. 3.4 Termina la configuración añadiendo la lista de parámetros y el

						estilo del reporte. 4. El sistema genera el reporte en formato pdf a partir de la configuración realizada.
EC 1.2 Campos vacíos.	Existen campos vacíos y no se puede generar un reporte	I Vacío V Gráfica	V Predeter minado V Vacío	V No realizada V Realizada	El sistema no procede a la ejecución de los pasos posteriores para generar el reporte mientras hallan campos vacíos.	1. El actor solicita al sistema el servicio de generar reporte en formato pdf. 2. El sistema envía la configuración a realizar para generar el reporte. 3. El actor realiza la configuración siguiente: Ver CU Configurar reporte 3.1Crea los parámetros que tendrá el reporte (imagen, tabla, párrafo, gráfica). 3.2 Lista los parámetros en el orden que desee ser representados en el reporte. 3.3 Selecciona el estilo del reporte. 3.4 Termina la configuración añadiendo la lista de parámetros y el estilo del reporte. 4. El sistema genera el reporte en formato pdf a partir de la configuración realizada.

CU Exportar reporte en formato xls

- **Descripción generar:** Este caso de uso le permite al actor generar y exportar un reporte en formato xls.
- **Condiciones de ejecución:** El actor posee información para generar un reporte
- **Secciones:** Diseño del caso de prueba

Tabla 7: *Diseño de caso de prueba para el caso de uso exportar reporte en formato xls*

Escenario	Descripción	Parámetros	Estilo	Configurar reporte	Respuesta del sistema	Flujo central
EC 1.1 Generar reporte en formato xls	Se exporta el reporte construido por el actor en formato xls	V Tabla	V Predeterminado	V Realizada	El sistema permite generar un reporte en formato xls. Le proporciona las vías para realizar la configuración a partir del estilo y los parámetros a incluir en el reporte. Ver CU Configurar reporte	1. El actor solicita al sistema el servicio de generar reporte en formato xls. 2. El sistema envía la configuración a realizar para generar el reporte. 3. El actor realiza la configuración siguiente: Ver CU Configurar reporte 3.1Crea los parámetros que tendrá el reporte (imagen, tabla, párrafo, gráfica). 3.2 Lista los parámetros en el orden que desee ser representados en el reporte. 3.3 Selecciona el estilo del reporte. 3.4 Termina la configuración añadiendo la lista de parámetros y el estilo del reporte. 4. El sistema genera el reporte en formato xls a partir de la configuración realizada.
EC 1.2 Campos vacíos.	Existen campos vacíos y no se puede generar un reporte	I Vacío V Gráfica	V Predeterminado V Vacío	V No realizada V Realizada	El sistema no procede a la ejecución de los pasos posteriores para generar el reporte mientras hallan campos vacíos.	1. El actor solicita al sistema el servicio de generar reporte en formato xls. 2. El sistema envía la configuración a realizar para generar el reporte. 3. El actor realiza la configuración siguiente: Ver CU Configurar reporte. 3.1Crea los parámetros que tendrá el reporte (imagen, tabla, párrafo, gráfica). 3.2 Lista los parámetros en el orden que desee ser representados en el reporte. 3.3 Selecciona el estilo del reporte. 3.4 Termina la

						configuración añadiendo la lista de parámetros y el estilo del reporte. 4. El sistema genera el reporte en formato xls a partir de la configuración realizada.

Descripción de las variables:

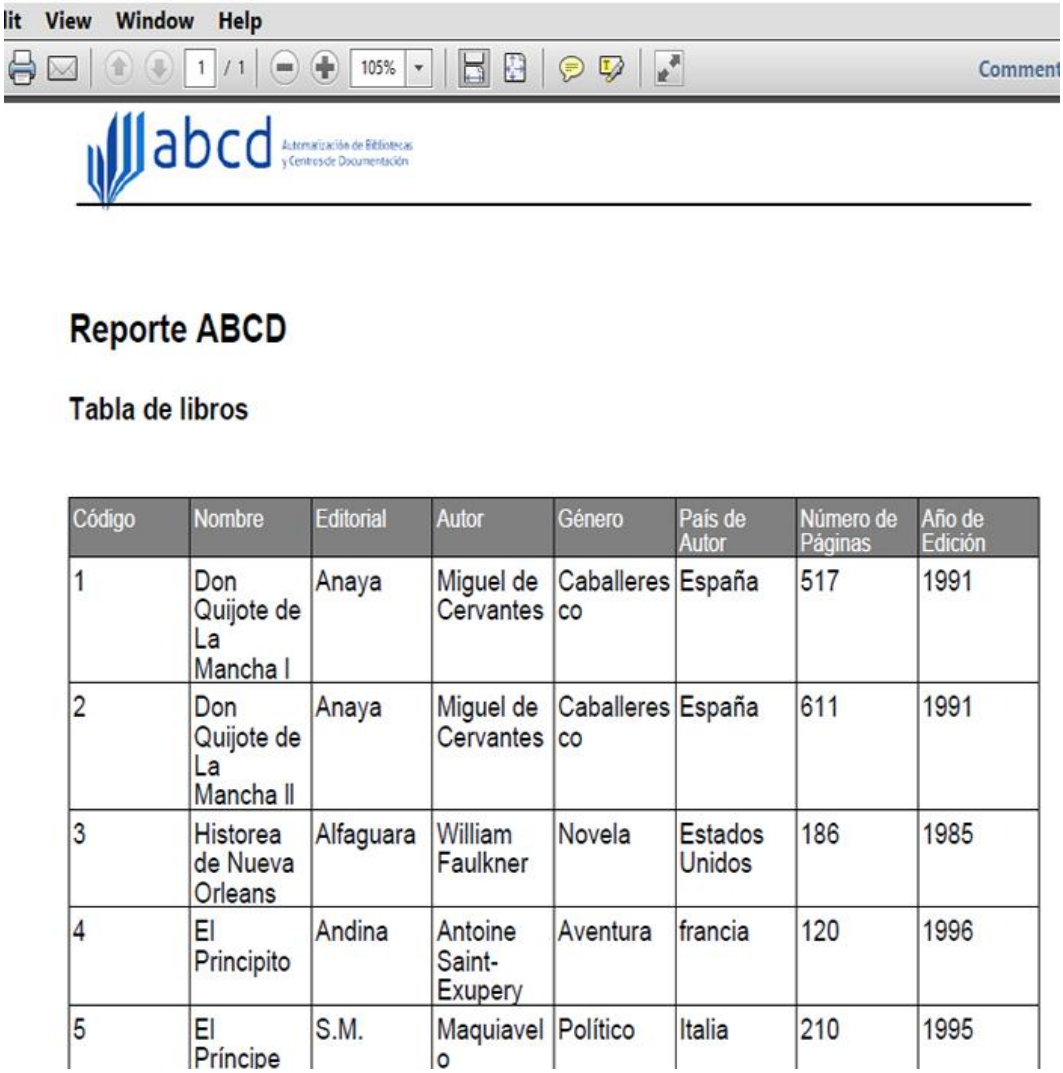
Tabla 8: *Descripción de las variables de los casos de prueba*

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Parámetros	lista de elementos	No	Lista de los elementos que conformarán un reporte (tabla, imagen, párrafo, gráfica)
2	Estilo	objeto de selección	No	Estilo con el cual se conformará el documento
3	Configuración	Acción	No	Acción de configuración de reporte. Ver CU Configurar reporte.

3.4.2 Resultados de las pruebas

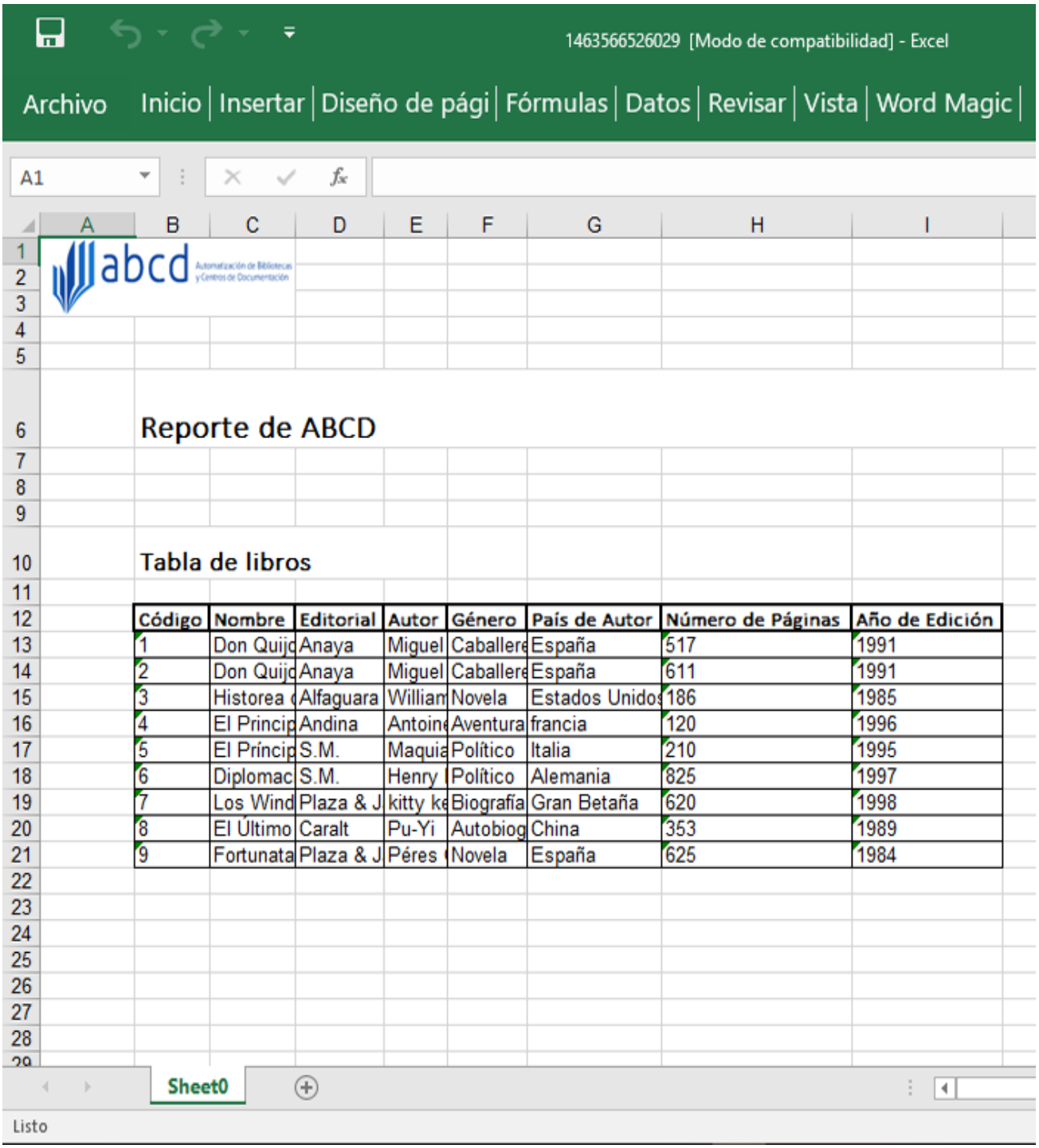
Durante la realización de las pruebas al componente se realizaron varios casos de pruebas para cada funcionabilidad con el objetivo de validar el correcto funcionamiento del mismo. Como resultado final se logró una aplicación que cumple con los requisitos funcionales presentados. Para corroborar esto se presenta a continuación un conjunto de imágenes correspondiente a los resultados obtenidos en algunos de los casos de pruebas realizados:

- **Reportes generados con una tabla para los formatos pdf y xls respectivamente**



Código	Nombre	Editorial	Autor	Género	País de Autor	Número de Páginas	Año de Edición
1	Don Quijote de La Mancha I	Anaya	Miguel de Cervantes	Caballeresco	España	517	1991
2	Don Quijote de La Mancha II	Anaya	Miguel de Cervantes	Caballeresco	España	611	1991
3	Historea de Nueva Orleans	Alfaguara	William Faulkner	Novela	Estados Unidos	186	1985
4	El Principito	Andina	Antoine Saint-Exupery	Aventura	francia	120	1996
5	El Príncipe	S.M.	Maquiavelo	Político	Italia	210	1995

Figura 17: *Reporte representando una tabla relacionada con datos de libros en formato pdf*



Código	Nombre	Editorial	Autor	Género	País de Autor	Número de Páginas	Año de Edición
1	Don Quijo	Anaya	Miguel	Caballere	España	517	1991
2	Don Quijo	Anaya	Miguel	Caballere	España	611	1991
3	Historea	Alfaguara	William	Novela	Estados Unidos	186	1985
4	El Princip	Andina	Antoine	Aventura	francia	120	1996
5	El Príncip	S.M.	Maquia	Político	Italia	210	1995
6	Diplomac	S.M.	Henry	Político	Alemania	825	1997
7	Los Wind	Plaza & J	kitty ke	Biografía	Gran Betaña	620	1998
8	El Último	Caralt	Pu-Yi	Autobiog	China	353	1989
9	Fortunata	Plaza & J	Péres	Novela	España	625	1984

Figura 18: *Reporte representando una tabla relacionada con datos de libros en formato xls*

- **Reportes generados con una gráfica de barras para los formatos pdf y xls respectivamente**

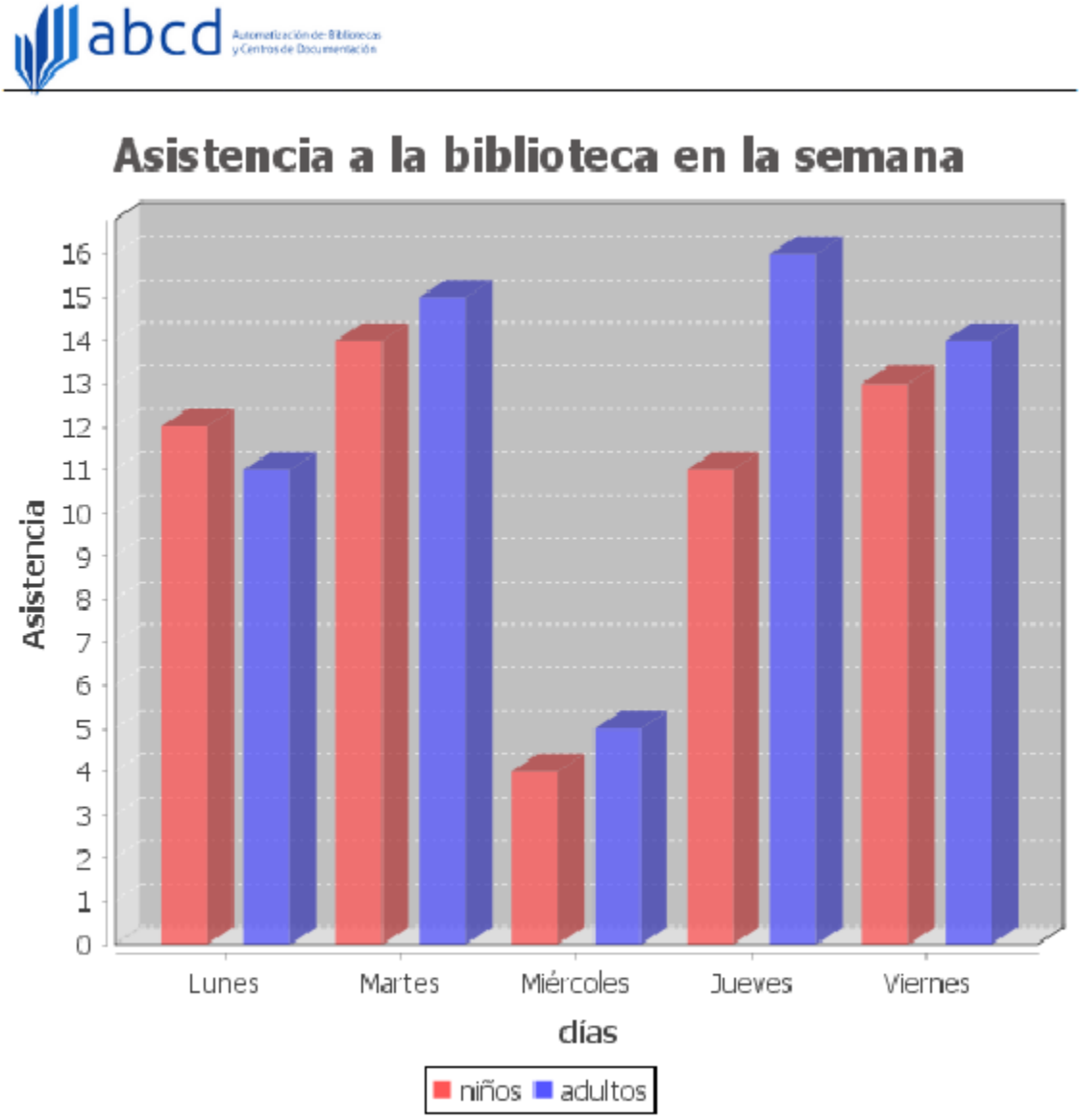


Figura 19: *Reporte con una gráfica de barras representando los datos de la asistencia a la biblioteca en una* semana en formato pdf

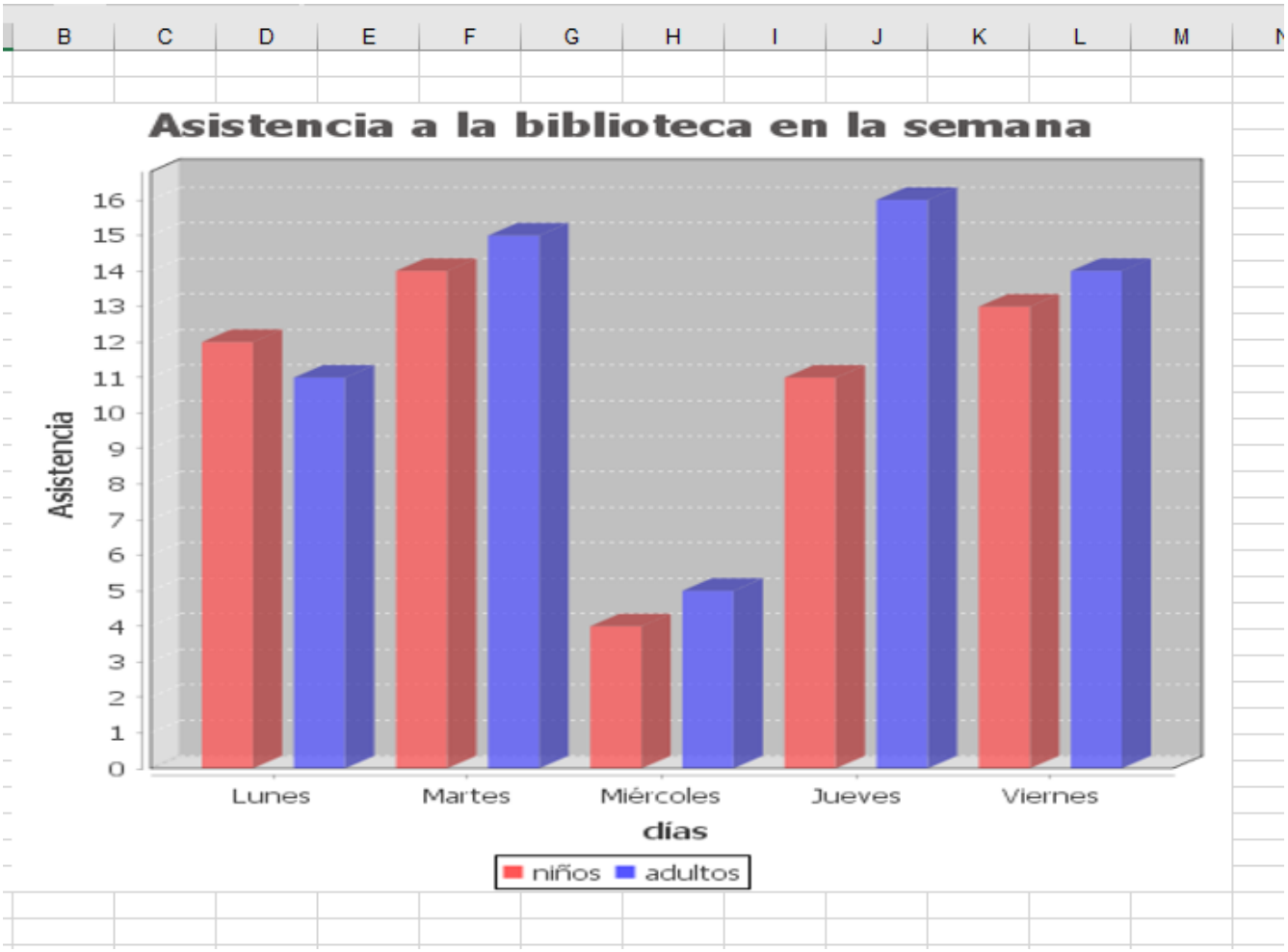
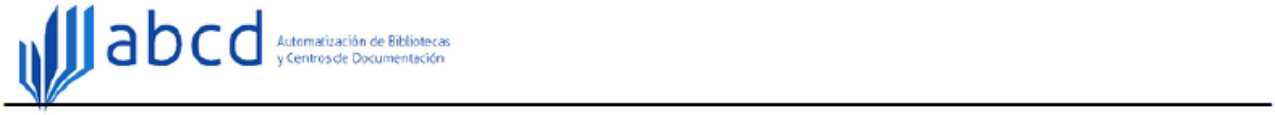


Figura 20: *Reporte con una gráfica de barras representando los datos de la asistencia a la biblioteca en una semana en formato xls*

- **Reportes generados con un párrafo como contenido en los formatos pdf y xls respectivamente**



Reporte ABCD

Reportes

El componente generación de reportes para el Sistema de Automatización de Bibliotecas y Centros de Documentación es un proyecto sustentado sobre las necesidades de la Arquitectura Base Orientada a Servicios de exportar información existente en el sistema en formatos pdf y xls. El objetivo principal de esta investigación es la creación de un componente que se integre en dicha arquitectura y que permita exportar los reportes en los formatos propuestos.

Figura 21: *Reporte con un párrafo como contenido relacionado con la automatización de las bibliotecas en formato pdf*

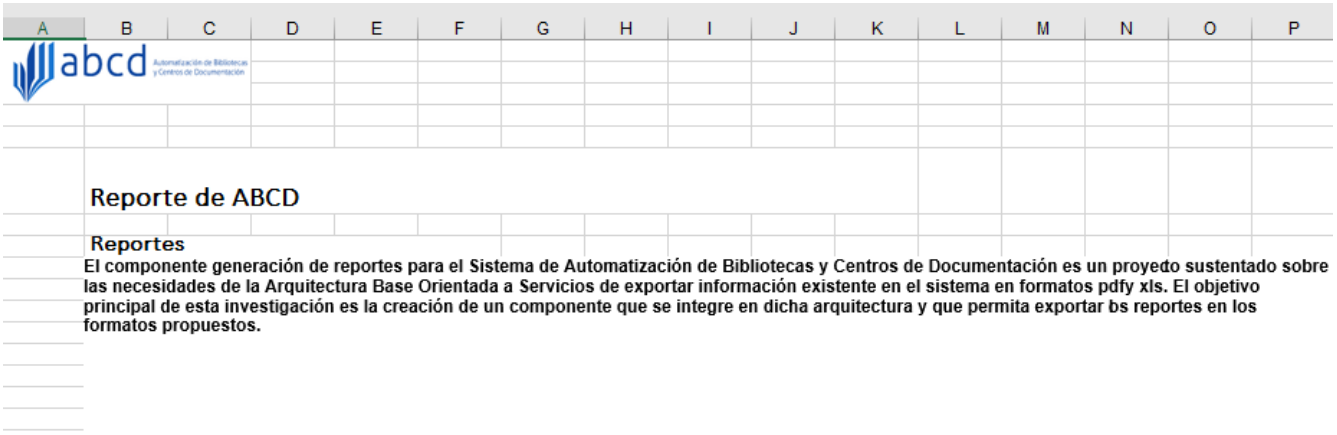
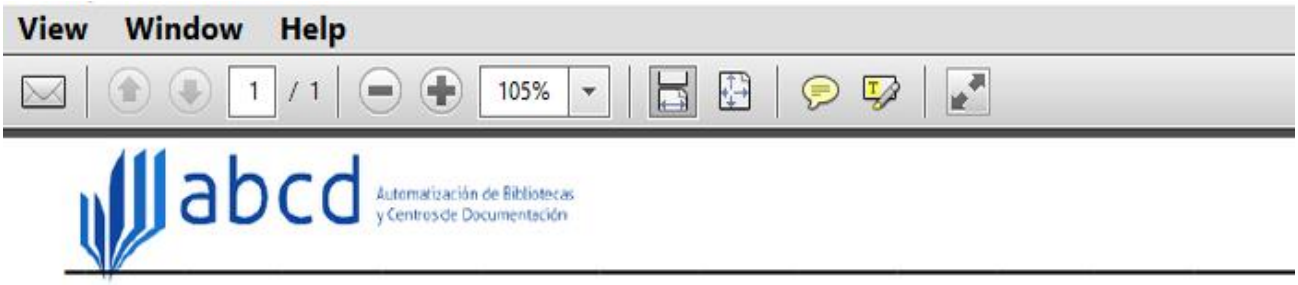


Figura 22: *Reporte con un párrafo como contenido relacionado con la automatización de las bibliotecas en formato xls*

- **Reportes generados con una imagen como contenido en los formatos pdf y xls respectivamente**



Reporte ABCD

Mi biblioteca

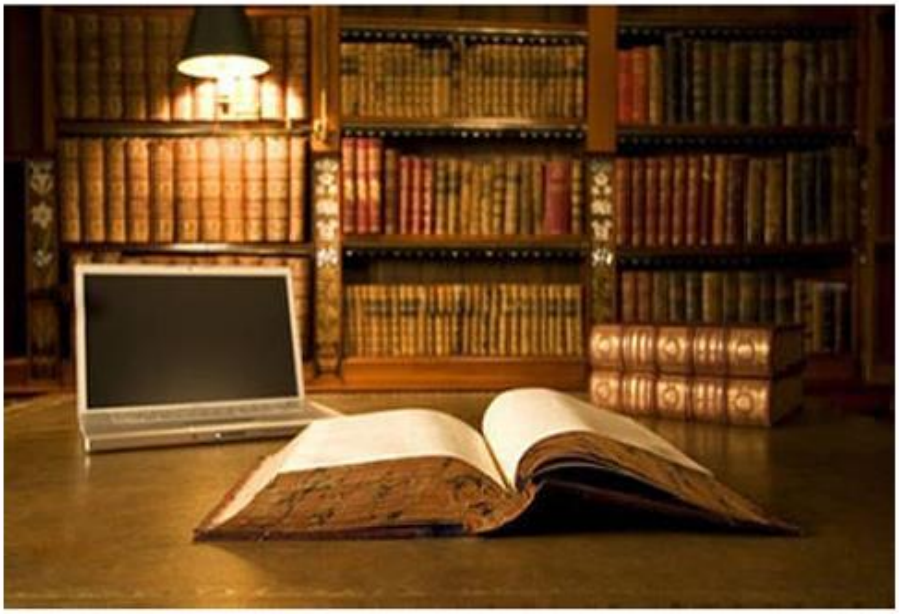


Figura 23: *Reporte con una imagen como contenido en formato pdf*

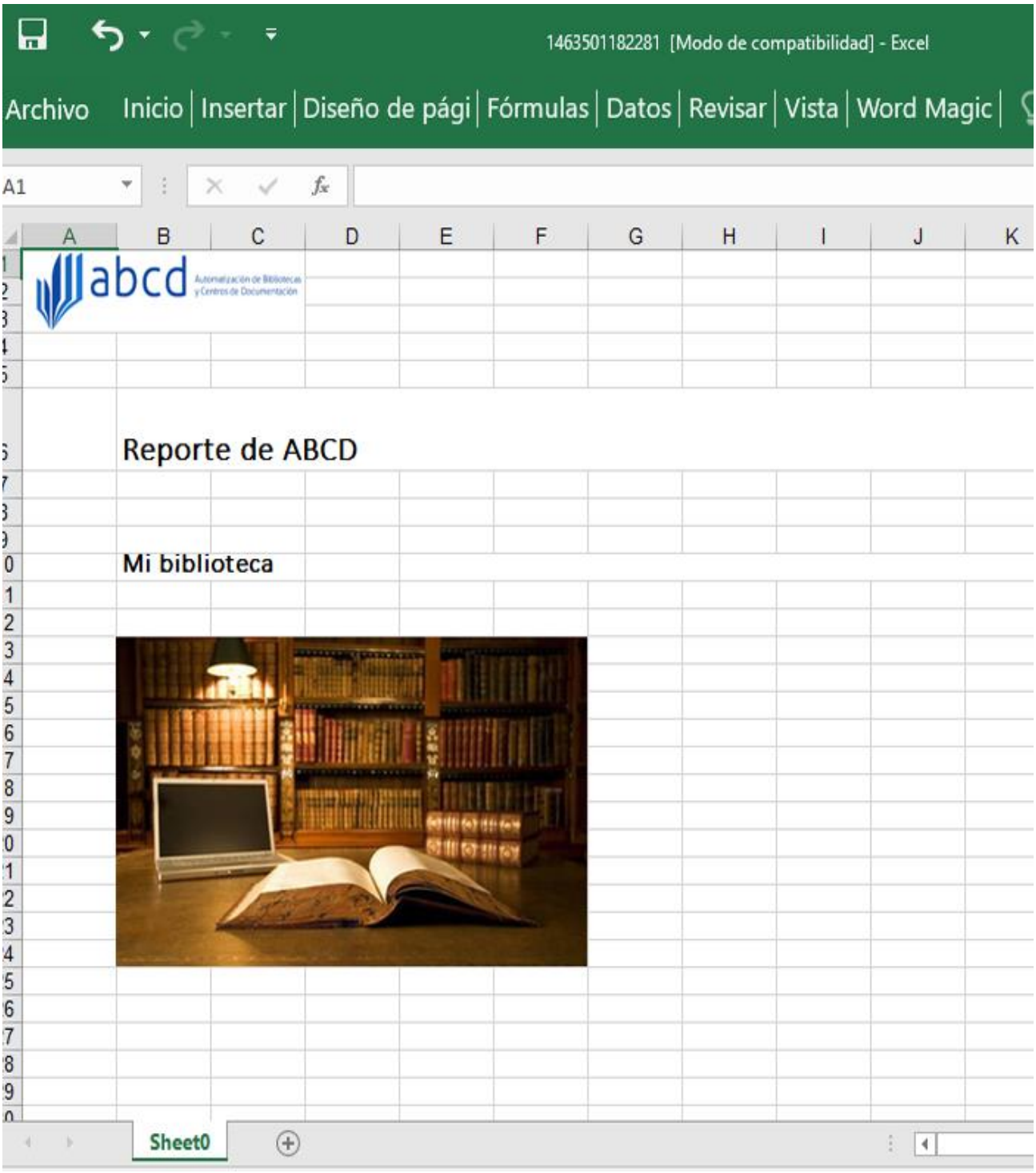


Figura 24*: Reporte con una imagen como contenido en formato xls*

- **Reportes generados con una gráfica de pastel como contenido en los formatos pdf y xls respectivamente**

Figura 25: *Reporte con una gráfica de pastel con datos de las visitas a la biblioteca por meses en formato pdf*

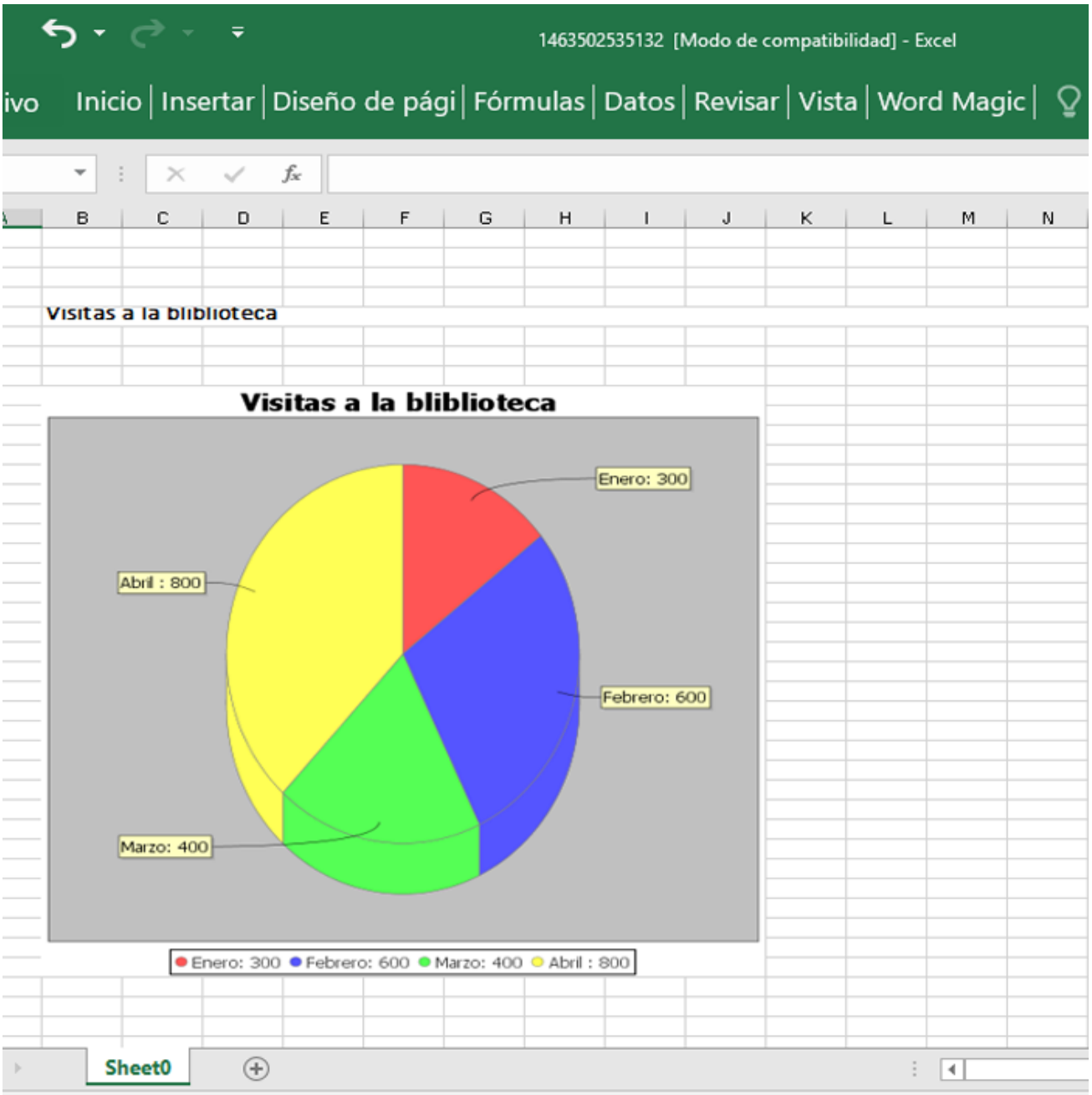


Figura 26: *Reporte con una gráfica de pastel con datos de las visitas a la biblioteca por meses en formato xls*

- **Reportes generados con una gráfica de línea como contenido en los formatos pdf y xls respectivamente**

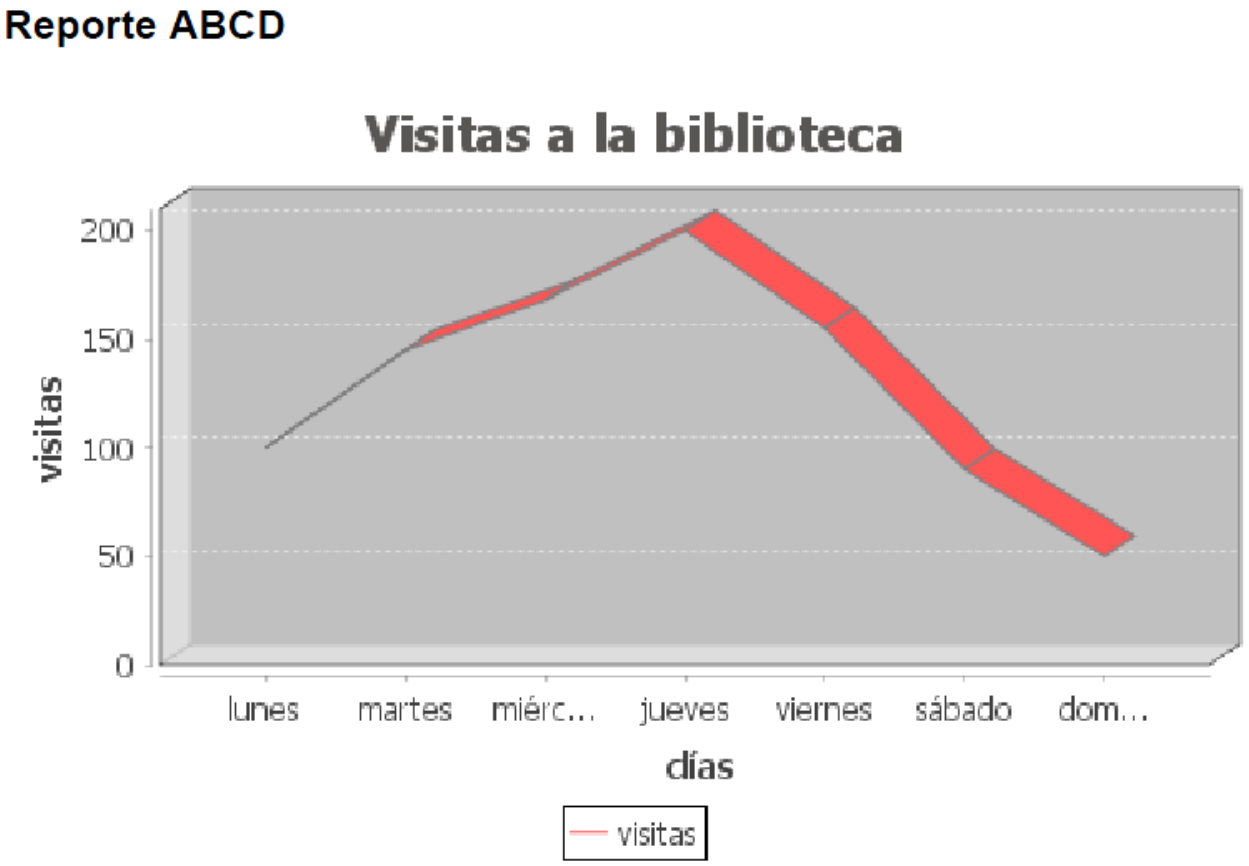


Figura 27: *Reporte con una gráfica de línea con datos de las visitas a la biblioteca en una semana en formato pdf*

Figura 28: *Reporte con una gráfica de línea con datos de las visitas a la biblioteca en una semana en formato xls*

3.5 No Conformidades encontradas en los casos de prueba

En el trascurso de las pruebas realizadas fueron encontradas algunas no conformidades que posteriormente fueron solucionadas. A continuación, se muestra las gráficas con la relación de las no conformidades encontradas por cada caso de pruebas realizado:

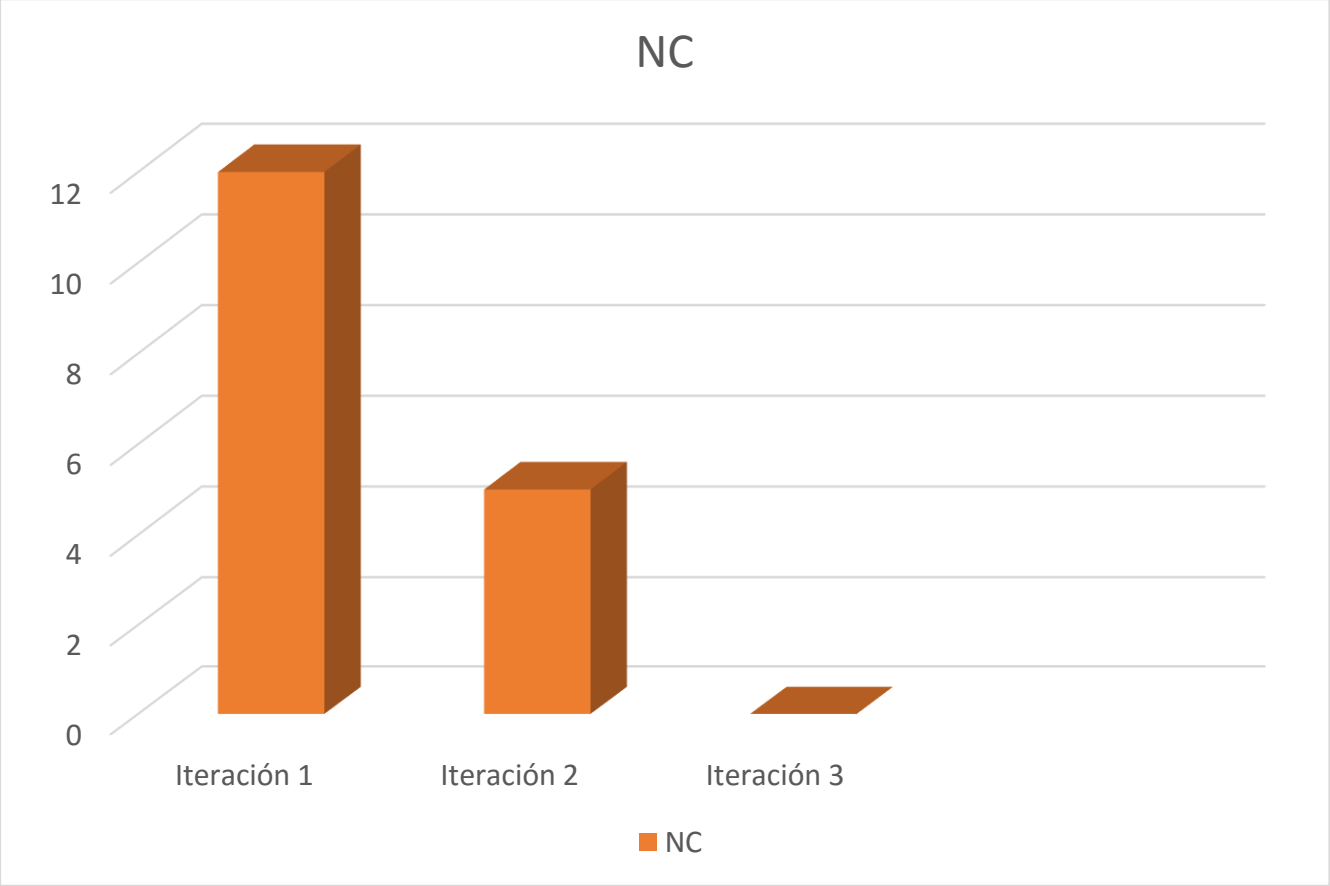


Figura 29: *No conformidades del caso de prueba exportar reporte en formato pdf*

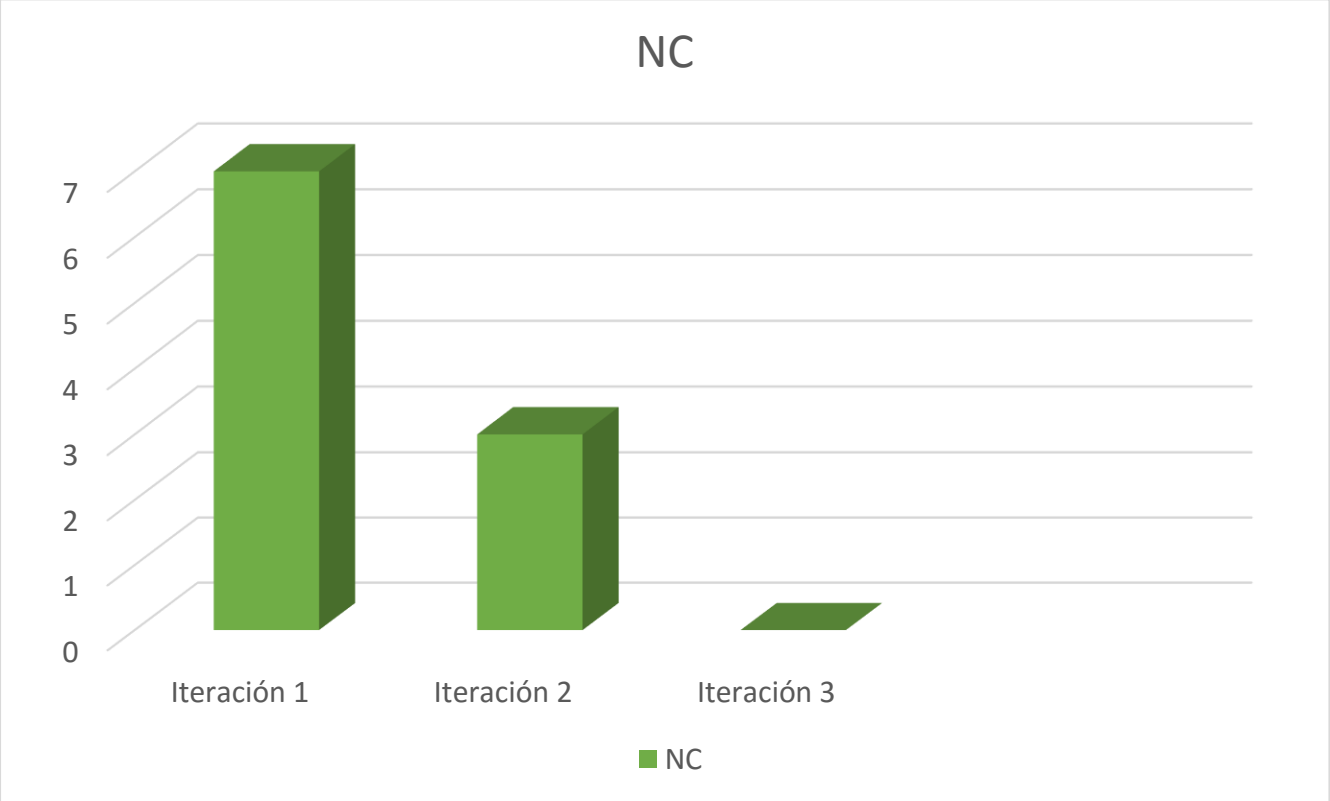


Figura 30: *No conformidades del caso de prueba exportar reporte en formato xls*

3.6 Conclusiones parciales

Con el desarrollo de este capítulo se implementaron todos los casos de pruebas necesarios para validar el correcto funcionamiento del componente, así como el cumplimiento de sus funcionalidades. Para verificar esto se utilizó la implementación de las pruebas de caja negra obteniéndose un resultado satisfactorio en cada una de las pruebas realizadas. Como resultado final se obtuvo un componente completamente funcional el cual cumple con todos los requerimientos definidos por el cliente.

CONCLUSIONES

El desarrollo del presente trabajo estuvo basado en un conjunto de etapas que fueron realizadas para dar cumplimiento a los objetivos específicos planteados en aras de alcanzar las metas propuestas para su realización. Por lo que la culminación del componente Generación de Reporte permite arribar a las siguientes conclusiones:

- Con la valoración de las herramientas de generación de reporte que utiliza java se seleccionaron las librerías necesarias para el desarrollo del componente. Se seleccionó la librería Itext para el desarrollo de los reportes en formato pdf, Apache POI para la generación de reportes en formato xls y JFreeChart para incluir en los reportes las gráficas estadísticas.
- En el transcurso del desarrollo del componente se realizó el diseño e implementación de las funcionalidades identificadas. La solución obtenida es un componente capaz de generar reportes en los formatos propuestos pdf y xls. Los mismos permiten mostrar la información en forma de tablas, párrafos, imágenes y gráficas de comparación como son las gráficas de barras, de pastel y de línea, cumpliéndose de esta forma el objetivo propuesto.
- Las pruebas realizadas al componente permitieron corregir las fallas internas que no se habían encontrado durante el proceso de desarrollo, proporcionando una mayor calidad al sistema y viabilidad al cliente. Con el resultado final de las pruebas se concluyó que el componente Generación de Reporte estaba listo para ser integrado en la arquitectura ABOS como bundle, brindando los servicios de generación de reporte en formato pdf y xls respectivamente.

RECOMENDACIONES

A partir de las experiencias adquiridas durante el desarrollo del presente trabajo se proponen las siguientes recomendaciones:

- Continuar con el desarrollo del componente generación de reportes mediante la incorporación de otros formatos a exportar los reportes como son los formatos doc y xml.
- Crear y modificar los estilos de los reportes dentro del sistema de acuerdo a la entidad donde se despliegue el componente.
- Incorporar nuevos juegos de datos al componente para representar la información.

Referencias

1. **EclipseLink. [En línea] [Citado el: 7 de 2 de 2016.] http://www.eclipse.org/eclipselink/.**

2. Remote Application Platform (RAP). [En línea] [Citado el: 15 de 1 de 2016.] http://www.eclipse.org/rap/.

3. Java y Tú. [En línea] [Citado el: 14 de 1 de 2016.] https://www.java.com/es/.

4. Spring Dynamic Modules Reference Guide. [En línea] [Citado el: 15 de 2 de 2016.] http://docs.spring.io/osgi/docs/current/reference/html/.

5. Alexandre de Castro, Alves. *OSGi in Depth.* Shelter Island, NY : Manning, 2012. 978-1-935182-17-7.

6. Cogoluègnes, Arnaud. *Spring dynamic modules in action.* s.l. : Manning, 2010. 978-1-935182-30-6.

7. Heffelfinger, David R. *JasperReports for Java developers.* s.l. : Packt Publ, 2006.

8. iReport Designer | Jaspersoft Community. [En línea] [Citado el: 17 de 2 de 2016.] http://community.jaspersoft.com/project/ireport-designer.

9. Lowagie, Brouno. *Itext in action.* s.l. : manning, 2011.

10. JFreeChart. [En línea] [Citado el: 20 de 11 de 2015.] http://www.jfree.org/jfreechart/.

11. Apache POI - the Java API for Microsoft Documents. [En línea] [Citado el: 10 de 2 de 2016.] https://poi.apache.org/.

12. BIRT Home. [En línea] [Citado el: 5 de 2 de 2016.] http://www.eclipse.org/birt/.

13. S.Pressman, Roger. *INGIENERÍA DEL SOFTWARE. UN ENFOQUE PRÁTICO Séptima edición.* México : McGrawHil, 2010. 978-607-15-0314-5.

14. Larman, Craig. *UML y patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : Prentice Hall, 1999. ISBN 970-17-0261-1.

15. Software Design Tools for Agile Teams, with UML, BPMN and More. [En línea] [Citado el: 15 de 1 de 2016.] https://www.visual-paradigm.com/.

16. Eclipse - The Eclipse Foundation open source community website. [En línea] [Citado el: 14 de 1 de 2016.] https://eclipse.org/.

REFERENCIAS BIBLIOGRÁFICAS

17. Ivar Jacobson, Grady Booch y James Rumbaugh. *El Proceso Unificado De Desarrollo de Software.* Madrid : Addison Wesley, 2000. ISBN 84-7829-036-2.

18. Larman, Craig. *UML y Patrones.* 2da Edición. México : Prentice Hall, 2002. 0-13-092569-1.

19. Pressman, Roger S. *Ingeniería de Software.* s.l. : Mc Graw Hill.

Bibliografía

1. **EclipseLink. [En línea] [Citado el: 7 de 2 de 2016.] http://www.eclipse.org/eclipselink/.**

2. Remote Application Platform (RAP). [En línea] [Citado el: 15 de 1 de 2016.] http://www.eclipse.org/rap/.

3. Java y Tú. [En línea] [Citado el: 14 de 1 de 2016.] https://www.java.com/es/.

4. Spring Dynamic Modules Reference Guide. [En línea] [Citado el: 15 de 2 de 2016.] http://docs.spring.io/osgi/docs/current/reference/html/.

5. Alexandre de Castro, Alves. *OSGi in Depth.* Shelter Island, NY : Manning, 2012. 978-1-935182-17-7.

6. Cogoluègnes, Arnaud. *Spring dynamic modules in action.* s.l. : Manning, 2010. 978-1-935182-30-6.

7. Heffelfinger, David R. *JasperReports for Java developers.* s.l. : Packt Publ, 2006.

8. iReport Designer | Jaspersoft Community. [En línea] [Citado el: 17 de 2 de 2016.] http://community.jaspersoft.com/project/ireport-designer.

9. Lowagie, Brouno. *Itext in action.* s.l. : manning, 2011.

10. JFreeChart. [En línea] [Citado el: 20 de 11 de 2015.] http://www.jfree.org/jfreechart/.

11. Apache POI - the Java API for Microsoft Documents. [En línea] [Citado el: 10 de 2 de 2016.] https://poi.apache.org/.

12. BIRT Home. [En línea] [Citado el: 5 de 2 de 2016.] http://www.eclipse.org/birt/.

13. S.Pressman, Roger. *INGIENERÍA DEL SOFTWARE. UN ENFOQUE PRÁTICO Séptima edición.* México : McGrawHil, 2010. 978-607-15-0314-5.

14. Larman, Craig. *UML y patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : Prentice Hall, 1999. ISBN 970-17-0261-1.

15. Software Design Tools for Agile Teams, with UML, BPMN and More. [En línea] [Citado el: 15 de 1 de 2016.] https://www.visual-paradigm.com/.

16. Eclipse - The Eclipse Foundation open source community website. [En línea] [Citado el: 14 de 1 de 2016.] https://eclipse.org/.

BIBLIOGRAFÍA

17. Ivar Jacobson, Grady Booch y James Rumbaugh. *El Proceso Unificado De Desarrollo de Software.* Madrid : Addison Wesley, 2000. ISBN 84-7829-036-2.

18. Larman, Craig. *UML y Patrones.* 2da Edición. México : Prentice Hall, 2002. 0-13-092569-1.

19. Pressman, Roger S. *Ingeniería de Software.* s.l. : Mc Graw Hill.

20. Guevara, Jorge Martinez Ladron de. *Fundamentos de la Programación en Java.* s.l. : Facultad de Informática Universidad Complutense de Madrid.España.

21. Visual Paradigm. [En línea] http://www.visual-paradigm.com..

22. Significados. [En línea] 2013-2016. http://www.significados.com/item/.

23. iText. [En línea] [Citado el: 20 de 1 de 2016.] http://itextpdf.com/.

24. Benefits of Using OSGi – OSGi™ Alliance. [En línea] [Citado el: 14 de enero de 2016.] https://www.osgi.org/developer/benefits-of-using-osgi/.

25. Definición de Reporte . [En línea] [Citado el: 15 de noviembre de 2015.] http://definicion.mx/reporte/.

26. JasperReports® Library | Jaspersoft Community. [En línea] [Citado el: 17 de 2 de 2016.] http://community.jaspersoft.com/project/jasperreports-library.

27. Jacobson Ivar, Booch Grady, Rumbaugh James. *El PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE.* Madrid : ADDISON WESLEY, 2000. 84-7829-036-2000.

GLOSARIO DE TÉRMINOS

ABCD: Automatización de Bibliotecas y Centros de documentación.

ABOS: Arquitectura Base Orientada a Servicios.

API: Una API (siglas de Application Programming Interface) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

Componente: Refiere al componente de software que es un elemento de un sistema de software que ofrece un conjunto de servicios o funcionalidades.

CRUD: Es el acrónimo de Crear, Leer, Actualizar y Borrar se usa para referirse a las funcionalidades básicas en base de datos o la capa de persistencias en un software.

Framework: Marco de trabajo que constituye una estructura conceptual y tecnológica compuesta por componentes de software que permiten el desarrollo rápido de software.

Java EE: Java Platform, Enterprise Edition (anteriormente conocido como Java 2 Platform, Enterprise Edition o **J2EE** hasta la versión 1.4; traducido informalmente como **Java Empresarial**) es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

JDBC: Acrónimo de Java Database conmectivity. API que permite la ejecución de operaciones sobre base de datos desde el lenguaje java.

JPA: Java Persistence API sus siglas en inglés, proporciona un modelo de persistencia para mapear bases de datos en java.

ORM: Object Relational Mapping sus siglas inglés, permite la transformación de tablas de la base de datos en entidades.

PDF: Del inglés Portable Document Format, formato de documento portátil, es un formato de almacenamiento de documentos.

UML: Unified Modeling Language sus siglas en inglés o lenguaje Unificado de Modelado trata de un estándar para crear esquemas, diagramas y documentación relativa al desarrollo de software.

XLS: Del inglés **(**Microsoft Excel file format), extensión para los archivos de hojas de cálculo utilizados en Microsoft Excel.