Universidad de las Ciencias Informáticas

Facultad 3

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Componente para la creación de reportes del marco de trabajo Bosón.

Autor:

Yania Lisbet Bastida Cantero

Tutor(es):

Ing. Julio C. Ocaña Bermúdez

Ing. Aniel Pérez Orozco

La Habana, Junio, 2016

"Año 58 de la Revolución"

Lo importante en ciencia no es tanto obtener nuevos hechos como descubrir nuevas formas de pensar en ellos.

William Lawrence Bragg

Declaración de autoría

Declaro que soy la única autora del trabajo “Componente para la creación de reportes para el marco de trabajo Bosón” y autorizo a la Universidad de las Ciencias Informáticas hacer uso de la misma en su beneficio, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Autor: Yania Lisbet Bastida Cantero

_____

Tutor: Ing. Julio C. Ocaña Bermúdez

_____

Tutor: Ing. Aniel Pérez Orozco

_____

Co-Tutor: Ing. Addiel Águila Espinaco

Datos de contacto

Autor: Yania Lisbet Bastida Cantero

Dirección: Trinidad, Sancti Spíritus, Cuba.

Teléfono (celular):53592281

Correo Electrónico: ylbastida@estudiantes.uci.cu

Tutor: Ing. Julio C. Ocaña Bermúdez

Categoría Científica: Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI.

Correo Electrónico: jcocana@uci.cu

Dpto. Departamento de Desarrollo de Componente.

Centro: Centro de Informatización de Entidades.

Tutor: Ing. Aniel Pérez Orozco

Categoría Científica: Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI).

Correo Electrónico: aporozco@uci.cu

Centro: Centro de Informatización de Entidades.

Co-Tutor: Ing. Addiel Águila Espinaco

Categoría Científica: Ingeniero en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI), graduado en el año #.

Correo Electrónico: aespinaco@uci.cu

Centro: Cultural Wilfredo Lam

Dedicatoria

Quiero dedicarles esta tesis a Dios y a los principales promotores de mi vida. A mi madre la guía de los 3 integrantes de la familia a ti por ser la persona que siempre ha confiado en mí, a mi papá por ser la persona que siempre está a nuestro lado y mi Tata, el hermano que siempre me cuida y me apoya. Este título va para ustedes

Agradecimientos

Agradezco a mi madre que se ha pasado su vida entera estudiando sin estar en la escuela gracias por tu dedicación, por siempre estar hay en los momentos buenos y malos de mi vida y darme siempre ese ánimo de que yo sí puedo.

A mi padre que aunque él no lo crea es para mí el papá mejor del mundo a ti gracias por estar a mi lado en los momentos difíciles, A mi hermanito que es lo más grande que me ha dado mi mama y que espero ser un ejemplo a seguir para él.

A Moisés mi hermano mayor que lo considero parte de mi familia a ti gracias por apoyarme y darme ánimo cuando lo necesite, a mis tías Lidia, Elena que siempre estuvieron ahí para mi mamá apoyándola mientras yo no estaba gracias a ustedes por ayudarme en estos últimos momentos.

A mi novio Osmar que ha sido la personas más consecuentes en este tiempo a ti gracias por ayudarme y siempre estar ahí cuando más lo necesitaba. A mis amistades Verónica, Leyriel y Arianna que siempre recuerden que los estimo mucho por lo que son, A mi tutor Julio por ayudarme y estar siempre a mi lado apoyándome a ti gracias. A mis tutores Aniel y Espinaco por dedicarme su tiempo.

A mi grupo de baile MegaDance de la FRCA por los momentos buenos que pasamos juntos, A mis amistades de Trinidad por siempre estar pendientes de mi entre ellos Dariana, Yarisneidis, Adrian, Dana, Ernesto. A las personas que conocí estés año y que se volvieron mis fanáticos a ustedes gracias, a mis amistades del proyecto y a la UCI por darme la oportunidad de convertirme en una profesional, a todos ustedes mis más sinceros agradecimientos.

Resumen

El presente trabajo se centra en el desarrollo de un componente que sirva como apoyo al marco de trabajo Bosón en lo referente a generar reportes en diferentes formatos: pdf, xls y csv, lo cual evita la pérdida de tiempo del equipo de desarrollo a la hora de crear un reporte. Previo a la implementación se realizó una búsqueda y análisis de herramientas informáticas que generan reportes en la actualidad, propiciando estas infinidades de ventajas para el desarrollo del componente Reporte. Con el objetivo de guiar la investigación se empleó la Metodología de desarrollo para la actividad productiva de la UCI (AUP-UCI). Para lograr un producto con calidad, se decidió desarrollar un componente utilizando el framework Symfony 2.7.0 siendo este compatible con las mayorías de las plataformas, bibliotecas e infraestructura que existen. Se realizó un estudio de las herramientas y tecnologías propuestas por el Departamento de Desarrollo de Componentes del Centro de Informatización de Entidades que fueran compatibles con el componente a desarrollar. Como resultado del trabajo se obtuvo un componente que permite generar reportes de manera rápida, obteniéndose la información deseada por el usuario.

Palabras claves: csv, diseñar plantillas, generar reportes, pdf, xls.

Índice General

Índice de Tablas

Índice de Figuras

Introducción

El desarrollo actual de las tecnologías de la información y las comunicaciones (TIC) y la rapidez con que fluye la información a nivel mundial, ha propiciado que esta última sea considerada como uno de los principales activos de cualquier organización. Su valor, depende en gran medida, en la forma en que ayuda a los individuos de la organización para que tomen decisiones que los conduzcan a lograr los objetivos y metas propuestas.

En la actualidad existen disímiles instituciones que gestionan grandes volúmenes de datos correspondientes a sus negocios. En la mayoría de los casos estas organizaciones requieren de una alternativa eficiente que les proporcione información útil en tiempo real, por esta razón es recomendable devolver los datos en forma de reportes.

Un reporte es aquel documento que se utilizará cuando se desee mostrar una determinada información. Son de gran utilidad para conservar y mostrar los análisis y resultados de una organización.

La creación de reportes de forma dinámica a partir de la información almacenada permitirá una reducción de tiempo y esfuerzo en el análisis de la información, por lo que se considera un importante requisito que les permite a los usuarios acceder de forma simple y rápida a los datos con una mejor visualización de los resultados obtenidos.

Cuba no está exenta de las transformaciones substanciales que originan los avances en las TIC. En aras de lograr la independencia tecnológica se crea un centro de altos estudios llamado Universidad de las Ciencias Informáticas (UCI), que tiene entre sus objetivos la formación de profesionales que contribuyan a la producción de aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo. La Universidad de Ciencias Informáticas (UCI), cuenta con varios centros de producción de software, entre los que se encuentra el Centro de Informatización de Entidades (CEIGE) que tiene como objetivo el desarrollo de sistemas de gestión para el entorno empresarial mediante el uso de tecnologías web.

En dicha área radica el departamento de desarrollo de componentes donde se construye el marco de trabajo Bosón, que tiene como objetivo agilizar el proceso de desarrollo de software, brindando una base tecnológica en el desarrollo de aplicaciones web para la gestión de entidades y el apoyo a la toma de decisiones. La mayoría de las veces estos

sistemas deben generar información sobre las tablas de las bases de datos en forma de reportes, por lo que constituye una tarea común en el desarrollo de aplicaciones.

Actualmente la creación de reportes en el marco de trabajo Bosón es un proceso que trae consigo pérdida de tiempo, pues el equipo de desarrollo tiene que buscar mecanismos para generar un reporte en un formato definido, construir la consulta a través de códigos, diseñar la plantilla mediante código twig, y luego integrar todos los elementos antes descritos siendo este un proceso incómodo a la hora de obtener una información de forma rápida.

Lo antes expuesto constituye la problemática que origina esta investigación, por lo que se define el siguiente **problema a resolver** ¿Cómo disminuir el tiempo en la creación de reportes en los sistemas que usen el marco de trabajo Bosón?

Se determina como **objeto de estudio** el proceso de creación de reportes en las aplicaciones web, enmarcado en el **campo de acción** el diseño e implementación de plantillas dentro del proceso de creación de reportes para aplicaciones web.

Se define como **objetivo general de la investigación:** desarrollar un componente que permita disminuir el tiempo en la creación de reportes sobre el marco de trabajo Bosón.

Por lo tanto se plantea la siguiente **idea a defender:** si se desarrolla un generador de reportes para el marco de trabajo Bosón se minimizará el tiempo en la creación de un reporte.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos:**

1 Identificar los elementos teóricos necesarios para el proceso de creación de un reporte.
2 Desarrollar la propuesta de solución siguiendo la metodología de desarrollo escogida.
3 Implementar la solución a partir de los resultados de la etapa de análisis y diseño.
4 Validar la solución desarrollada a partir de la ejecución de pruebas de software.

Para guiar el desarrollo de la propuesta de solución, se plantearon las siguientes **tareas de la investigación**:

- Confección del marco teórico de la investigación a partir de la búsqueda y revisión bibliográfica de las herramientas que permiten generar reportes en formato pdf, xls, csv.

- Análisis de la estructura y funcionamiento del marco de trabajo Bosón.
- Captura, análisis y descripción de los requisitos funcionales y no funcionales para el desarrollo del componente para el marco de trabajo Bosón.
- Elaboración del modelo de datos del componente para la creación de reportes del marco de trabajo Bosón.
- Implementación del componente para la creación de reportes del marco de trabajo Bosón.
- Realización de las pruebas de aceptación y unitarias.
- Validación de la solución mediante un pre-experimento.

Métodos teóricos:

Histórico-lógico: con el objetivo de actualizar los antecedentes y el estado actual del desarrollo de generadores de reportes. Se utiliza para estudiar las formas de solución a problemas similares sobre los generadores de reportes en el mundo y en la UCI.

Analítico-Sintético: Se estudian las diferentes documentaciones y bibliografías especializadas referentes a los generadores de reportes para arribar a conclusiones que ayuden a comprender el componente a desarrollar.

Métodos empíricos:

Método de observación: se utiliza este método para poder percibir directamente los hechos de la realidad objetiva y observar las actividades realizadas para determinar cómo ocurre realmente el proceso de creación de un reporte.

El Trabajo de Diploma se encuentra estructurado en cinco secciones: resumen, introducción, desarrollo, conclusiones y referencias bibliográficas. El desarrollo está compuesto por tres capítulos donde se abarca todo el proceso investigativo realizado.

Capítulo 1: **Fundamentación teórica**

En este capítulo se abordan los elementos teóricos que sirven de base a la investigación del problema planteado. Se describen los conceptos fundamentales asociados al dominio del problema. Además se realiza un estudio y análisis de varias herramientas que generan reportes a nivel internacional y nacional. Se define el entorno tecnológico para el desarrollo de la propuesta de solución y se realiza un análisis de la bibliografía utilizada.

Capítulo 2: Descripción de la propuesta de Solución

En este capítulo se realiza la descripción de la propuesta de solución, la captura, análisis y descripción de los requisitos funcionales y no funcionales. Se describe la arquitectura del componente, haciendo énfasis en los patrones de diseño y en el patrón arquitectónico modelo-vista-controlador. Se presenta el modelo de datos que hace referencia al producto a desarrollar junto con los diagramas de clases de diseño con estereotipos web.

Capítulo 3: Implementación y Validación

En este capítulo se describe la implementación de la solución. Se valida la solución a través de las pruebas unitarias y de aceptación. Además, se valida la investigación propuesta a través de un pre-experimento.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se realiza un estudio de los generadores de reportes existentes, para lo cual se hace necesario conocer algunos conceptos y características relacionados con el tema. Además se describen las herramientas, tecnologías, metodología y lenguajes a utilizar con el fin de diseñar plantillas, crear consultas y generar reportes, para así tener una visión real del estado del arte de la misma.

1.2 Reportes

Según la Real Academia de la Lengua Española (RAE) **reportes** es un informe o noticia.

Informe: según la Real Academia de la Lengua Española (RAE) es la descripción, oral o escrita, de las características y circunstancias de un suceso o asunto.

Después de un estudio realizado referente a las definiciones de reportes se concluye, que para la investigación un reporte no es más que un informe que organiza y exhibe la información almacenada en una base de datos.

Los datos dentro de un reporte no pueden ser manipulados o modificados directamente, sino que tienen que ser transformados en alguna otra parte del sistema para que se reflejen los cambios una vez que el reporte sea generado nuevamente. Un reporte es generado dinámicamente, cada vez que se llama o se invoca desde el sistema, el reporte actualiza la información de los datos más recientes.

Generación de reportes: Es la obtención de reportes en un formato personalizado de tal manera que le sea útil al personal al cual va dirigido, en este proceso interviene la acción de los generadores de reportes.

Generadores de reportes: Son programas para crear informes en una amplia variedad de formatos. Extraen datos de los archivos o de las bases de datos y generan reportes de acuerdo al formato definido por el usuario, para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios.

1.3 Funcionamiento de los generadores de reportes

Los generadores de reportes se han creado con el objetivo de complementar los sistemas de información. Utilizan una especie de transparencia para el usuario por medio del cual

este realiza consultas a la base de datos y obtiene información de ella a través de reportes (Hernadez, 2003). Los generadores de reportes están compuestos principalmente por dos elementos básicos, un diseñador o editor de informes y la base de datos donde se almacenan los datos.

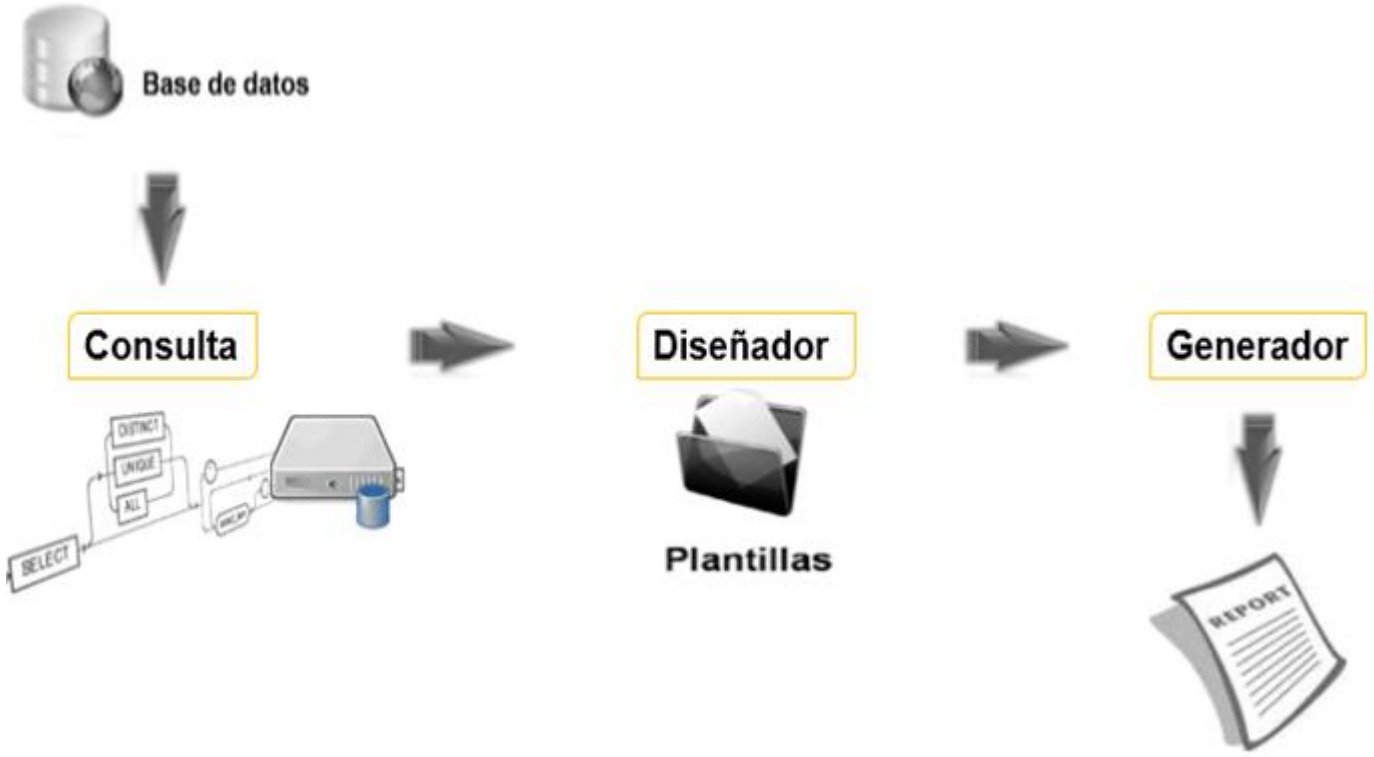

Figura 1. Esquema de un generador de reportes. **Fuente:** Elaboración del autor.

Componentes de un generador de reportes:

Reporte: estructura que contiene la información obtenida después de realizada la consulta a un conjunto de datos.

Consulta: conjunto de criterios y condiciones realizados a los campos de las entidades provenientes de una fuente de datos, permitiendo especificar diversos tipos de operaciones con los mismos. Se compone por la selección de los campos y las condicionales que se especifiquen en el diseño del reporte.

Diseñador de plantillas: sección de la aplicación donde se confecciona el reporte, incluyéndole título, descripciones, entre otros aspectos que requiera el desarrollador que se muestre en el informe.

Generador de reporte: sección de la aplicación en la cual una vez diseñado el reporte se pasa a generar el mismo, con el objetivo de obtener resultados en el formato definido.

El diseñador visual de reportes es un programa que ayuda a los usuarios y desarrolladores a la creación de reportes. A través de una interfaz simple de usar, se provee las funciones más importantes para crear plantillas en poco tiempo. Una vez concluida la etapa de diseño y captura de los datos a la base de datos a través de las consultas, comienza la etapa de generación del reporte, donde se toman los datos capturados y se mezclan con la plantilla diseñada, para luego visualizarlos en el formato definido por el usuario.

1.4 Etapas de la generación de reportes

El proceso de generación de reportes consiste en configurar la estructura de los informes en forma de plantilla, la cual contendrá la apariencia antes de incluir los datos de las consultas y así el reporte se obtiene con la presentación deseada por el desarrollador. A continuación, se presentan las etapas de creación de reportes.

- **Construcción de la consulta**
- **Construcción de la plantilla**

  Nombre: en esta sección estará el nombre del reporte.

  Descripción: está sección es opcional para el usuario.

  Cuerpo: esta sección es la encargada de mostrar los elementos que contiene el reporte.

- **Presentación del reporte:**

  Portada.

  Datos obtenidos.

1.5 Generadores de reportes

Existe una amplia gama de generadores de reportes que ofrecen varias experiencias globales que hacen que sean una referencia relevante para la concepción de un generador de reportes, es por eso que se hace necesario realizar un análisis de algunas de las soluciones que poseen características semejantes a las planteadas en la investigación.

Crystal Reports

Crystal Reports es la herramienta de elaboración de informes estándar para Visual Studio .NET de la empresa alemana SAP[1]**.** Utilizan un diseñador de Crystal Reports integrado. El diseñador de plantillas de Crystal permite diseñar y modificar los informes del entorno de programación integrado de Visual Studio .NET. Dicha herramienta puede programarse directamente desde Visual Studio .NET, es utilizado en el sistema operativo Microsoft Windows, dispone de licencias por usuario designado para el diseño de plantillas, independientemente de la edición de Crystal Reports adquirida, se distribuye bajo los términos de la EULA[2], licencia por la cual el uso de un producto solo está permitido para un único usuario (SOLUTIONS, 2013).

JasperReports Server

JasperReports Server es un servidor de informes independientes e integrables. Proporciona informes y análisis que se pueden incrustar en una aplicación web o móvil, así como operan un centro de información central para la empresa mediante la entrega de información en tiempo real. JasperReports Server exporta a diferentes formatos como son: HTML, PDF, EXCELS, CSV, ODT, RFT. Utiliza el lenguaje de programación Java y webservices para aplicaciones operacionales personalizadas; simplifica y mejora el rendimiento de la generación de informes para la entrega a los usuarios y trabaja bajo la licencia privativa (CORPORATION, 2013).

FastReport.Net

FastReport es una herramienta de generación de informes para diseñadores de software .Net que permite al usuario crear informes .Net independientes de la aplicación. FastReport.Net 2013.3 puede usarse con Microsoft Visual Studio 2005, 2008, Microsoft Visual Studio 2010-2012 y Delphi Prism, así como en el programa Visual Studio Express Edition y en dispositivos móviles. Posee características para generar informes para Delphi, .Net, C++ Builder y RAD Studio. Los reportes son exportados a los siguientes de formatos: HTML, XML, PDF, CSV, XLS entre otros (Report, 2016).

1 -Sistemas, Aplicaciones y Productos en Procesamiento de Datos.
2 -End User Licensing Agreement (Acuerdo de licencia de usuario final).

PHP Report Maker v6.0.1

PHP Report Maker permite crear informes web dinámicos de php desde las bases de datos MySQL, PostgreSQL, Microsoft Access, Microsoft SQL Server y Oracle. Con PHP Report Maker, los usuarios pueden exportar informes a formatos tales como HTML, PDF, Word o Excel. PHP Report Maker está diseñado para alta flexibilidad, permite numerosas opciones para generar los informes que mejor se adapten a las necesidades del cliente. Los códigos generados son limpios y fáciles de personalizar. Es multiplataforma, siendo esta una ventaja para los usuarios ya que se puede utilizar en varios sistemas operativos. Se encuentra bajo la licencia privativa (Raycraft, 2015).

Generador dinámico de reportes v.2

Generador dinámico de reportes es una aplicación desarrollada por el Centro de tecnologías de gestión de datos (DATEC) de la UCI. Trabaja sobre el marco de trabajo Symfony v1.4 el cual utiliza lenguaje php. Es multiplataforma. Soporta imágenes, gráficas, así como varios orígenes de datos. Genera reportes en los formatos: PDF, XML, HTML, CSV, XLS, RTF, TXT. El generador dinámico de reportes utiliza el lenguaje php para el desarrollo de la vista y para generar los reportes utiliza un motor de reportes JasperReports el cual está basado en lenguaje de programación Java (Claudia Garcia Suárez del Villar, 2013).

GeReport: Sistema de Gestión de Reportes Dinámicos

GeReport es una herramienta destinada al diseño, generación y configuración de los reportes relacionados con los datos históricos almacenados en una fuente de datos, generando reportes como imagen y en los formatos HTML, PDF y Excel. Para la interacción con las aplicaciones externas, el sistema implementa un servicio que expone los metadatos de los reportes para poder utilizarlos sin restricciones de lenguajes y plataformas. El lado del cliente se implementó utilizando código JavaScript, para lo cual, se usó Dojo 1.8 como marco de trabajo. El lado del servidor se desarrolló mediante código escrito en lenguaje php, para lo cual se utilizó CodeIgniter como marco de trabajo. Es una plataforma que puede ser usada sin pagos de licencia, se ejecuta bajo un entorno web, y es multiplataforma. (Rafael Felipe Bomate Gavio, 2014).

1.5.1 Análisis de los generadores de reportes.

Crystal Reports es una herramienta de elaboración de informes estándar para el lenguaje de programación Visual Studio lo que es incompatible con el producto a realizar, ya que el componente a desarrollar utiliza el lenguaje de programación php, no es multiplataforma por lo que solo es utilizada en el sistema operativo Windows, es un software privativo y de uso restringido.

La integración con JasperReports Server representa un alto costo funcional, ya que el lenguaje utilizado es Java, además de que dicha herramienta incumple con las pautas de diseño establecidas para Bosón en el aspecto arquitectónico.

FastReport.Net presenta el inconveniente de que está basado en los lenguajes de programación Delphi, C++ y en el entorno de desarrollo integrado Visual Studio.

PHP Report Maker v6.0.1 presenta el inconveniente que se encuentra bajo la licencia privativa, siendo este un obstáculo para utilizarlo.

Diseñador del generador dinámico de reportes v2.0 tiene como principal inconveniente que para generar un reporte utiliza el motor de generación de reportes JasperReports que el mismo está basado en Java y Bosón no utiliza motor de reporte, la versión de Symfony utilizada es la v1.4 siendo obsoleto para el componente que se está desarrollando.

La integración con el GeReport: Sistema de Gestión de Reportes Dinámicos representaría un alto costo en tiempo igual o mayor que desarrollar un nuevo sistema, pues serían muchas las modificaciones para lograr integrarlo y no se podría reutilizar toda la implementación ya que el sistema esta implementado en diferentes marcos de trabajo.

Teniendo en cuenta los inconvenientes presentados por las herramientas antes mencionadas, surge la presente investigación para desarrollar un componente multiplataforma que posea flexibilidad, permitiendo al usuario final el diseño de plantillas para generar reportes personalizados desde la web. Para el desarrollo del componente se utilizan las características y elementos primordiales de gran fortaleza que brindan los generadores de reportes, más utilizados en el mundo y en la UCI como guía para estructurar el componente de generador de reportes a desarrollar. Todo esto permite que al tratarse de una tecnología propia se faciliten los siguientes aspectos: asesoramiento continuo, personalización e integración del mismo con el marco trabajo Bosón.

1.6 Metodología de desarrollo

La metodología de desarrollo propuesta por la Universidad para el proceso de desarrollo del software es la Metodología de Desarrollo para la Actividad Productiva de la Universidad de las Ciencias Informáticas (AUP-UCI) que es una modificación, realizada por la UCI, del Proceso Unificado Ágil (AUP por sus siglas en inglés). En la metodología AUP-UCI se realiza una adaptación para proyectos correspondientes a la metodología AUP quedando la misma en 3 fases (Sánchez, 2014).

- **Inicio:** se corresponde con la primera fase de la metodología AUP. En esta fase se llevan a cabo las actividades relacionadas con la planeación del proyecto. Se realiza un estudio de la organización cliente para obtener la información necesaria para futuras estimaciones.
- **Ejecución:** se corresponde con las fases: Elaboración, Construcción y Transición de la metodología AUP. En esta fase se llevan a cabo las actividades necesarias para desarrollar el software: obtención de requisitos, definición de la arquitectura, el diseño del sistema, la implementación y liberación del producto.
- **Cierre:** nueva fase creada para la modificación de AUP. En esta fase se analizan los resultados del proyecto y su ejecución, además de que se realizan las actividades de cierre de proyecto.

La presente investigación estará enfocada a la fase de **Ejecución** ya que en la misma se efectúan las actividades requeridas para desarrollar el software, se realiza todo el modelado de la solución para una mejor comprensión del componente, se describen los requisitos, la arquitectura, se implementa y se entrega la solución al cliente.

La metodología define 4 escenarios para el modelado del negocio y el sistema en los proyectos (Sánchez, 2014) :

- **Escenario 1:** se encarga de modelar varias iteraciones entre los trabajadores del negocio/actores del sistema, donde la atención se centra en cómo el usuario va a utilizar el sistema.
- **Escenario 2:** el objetivo de este escenario se basa en la gestión y presentación de la información.
- **Escenario 3:** el escenario se centra en la representación de la cantidad de niveles de detalles y en las relaciones entre los procesos identificados.

Escenario 4: el escenario se basa en los proyectos que como resultado obtengan un negocio bien definido .El cliente estará siempre acompañando al equipo de desarrollo, y los proyectos no sean muy extensos. (Sánchez, 2014).

Por tener una definición bien clara del negocio y el componente a desarrollar no es muy extenso, la variante a escoger para el desarrollo es el **Escenario 4** ya que el producto de trabajo a entregar son las Historias de Usuario (HU) y estas no poseen mucha información.

1.7 Tecnologías y lenguajes empleados en la propuesta de solución

Para desarrollar la solución propuesta se emplean las herramientas y tecnologías definidas por el Departamento de Desarrollo de Componentes en la Vista de entorno de desarrollo tecnológico.

A continuación se describen las características que brindan las herramientas y tecnologías definidas.

1.7.1 Lenguaje de modelado UML 2.0

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés Unified Modeling Language) es un estándar de modelado, para la visualización, especificación, construcción y documentación de los artefactos de sistemas en los que el software juega un papel importante. Permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados (Rumbaugh, 2007).

Este lenguaje además, proporciona un conjunto de elementos de modelado, anotaciones, relaciones y normas que pueden aplicarse a una actividad de desarrollo de software. Sin embargo, UML también puede ser empleado para modelar otros dominios, tales como el modelado de sistemas y el modelado de negocio (Leffingwell, 2003).

Para facilitar el uso de este lenguaje de modelado se utiliza la herramienta Visual Paradigm la cual permite aprovechar las facilidades de uso y la amplia gama de utilidades que brinda. A continuación una descripción de esta herramienta.

1.7.2 Visual Paradigm 8.0 Enterprise Edition

Es una de las herramientas UML CASE del mercado, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones.

Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto. Además, apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (Leffingwell, 2003).

1.7.3 Marco de trabajo

Symfony v2.7.0

Es un popular marco de trabajo para desarrollar aplicaciones php. Está compuesto por varios componentes independientes creados por el proyecto Symfony. Ha sido ideado para aprovechar las características de PHP 5.3 y versiones superiores siendo uno de los marcos de trabajo de php con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no se acoplan correctamente en un determinado proyecto (Eguiluz, 2013).

AngularJS v1.3

AngularJS es un framework basado en JavaScript. Este framework se entiende como una estructura software, en este caso basada en el lenguaje JavaScript. Este permitirá desarrollar una aplicación con funcionalidades que ya están implementadas en AngularJS como extensión de HTML. AngularJS permite la creación de aplicaciones fundamentadas en el patrón modelo-vista-controlador, además de actualizar el HTML de las páginas de forma dinámica haciendo más fluida la interacción del usuario (Salinero, 2014).

Bosón v1.0

Bosón consiste en una arquitectura de referencia para los proyectos de la UCI que desarrollen en PHP. Este proporcionará todas las herramientas y las mejores prácticas para ayudar al desarrollo de productos. Con el desarrollo de este proyecto se ahorra tiempo y recursos materiales en la construcción de soluciones informáticas, ya que provee un conjunto de componentes tecnológicos que son indispensables para la mayoría de los proyectos. El hecho de utilizar como marco base a Symfony 2 garantiza la presencia de una comunidad mundial que brinda soporte y actualizaciones a diario. A continuación se presentan algunos de sus componentes:

Caché: permite utilizar varios mecanismos de almacenamientos para salvar temporalmente cualquier estructura de datos.

Seguridad: redefine la seguridad de Symfony2 para gestionar la autenticación y la autorización. Utiliza el estándar SAML 2.0 para la comunicación, el patrón SSO para la autenticación y la gestión de usuarios basada en roles para la autorización.

IUX: incluye todos los elementos necesarios para desarrollar con los componentes visuales de la Interfaz Única creada para los productos de la UCI.

Excepciones: permite definir las excepciones del sistema en un fichero de configuración y lanzarlas sin necesidad de crear clases. También admite la internacionalización de las excepciones así como un *log* de ocurrencia de las mismas.

Estructura y Composición (EC): permite definir de forma dinámica las estructuras y nomencladores que se necesiten en la aplicación a desarrollar.

Trazas: permite almacenar en una base de datos las trazas de excepciones, acciones y accesos a datos. También permite registrar el rendimiento (tiempo de ejecución y memoria) de las trazas de acción. (Torres, 2014)

1.7.4 Mapeador de objetos: ORM Doctrine 2

Doctrine 2 es un mapeador de objeto relacional (ORM) para PHP 5.3.0 que proporciona persistencia transparente de objetos php. La principal tarea de los objetos relacionales es la traducción transparente entre objetos php y las filas relacionales de la base de datos. Una de las características claves de Doctrine es la opción de escribir las consultas de base de datos en un dialecto SQL propio orientado a objetos llamado Lenguaje de Consulta Doctrine (DQL por Doctrine Query Language). Además DQL difiere ligeramente de SQL en que abstrae considerablemente la asignación entre las filas de la base de datos y objetos, permitiendo a los desarrolladores escribir consultas de una manera sencilla y flexible (Pacheco, 2011).

1.7.5 Servidor web. Apache 2.4.7

Apache es compatible con una multitud de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita de código abierto. Es un servidor altamente configurable de diseño modular siendo muy fácil ampliar sus capacidades. Trabaja con un gran número de lenguajes como Perl y PHP. Permite personalizar las respuestas ante los posibles errores que se puedan dar en el servidor. Tiene una alta configurabilidad en la creación y gestión de registros (Ciberaula, 2010).

1.7.6 Entorno de Desarrollo Integrado: PhpStorm 9.1

PhpStorm es un Entorno de Desarrollo Integrado (IDE) de php. El IDE proporciona autocompletamiento de código de manera inteligente, resaltando la sintaxis, configuración extendida del formato de código, comprobación de errores, entre otros. Es compatible con mezclas de idiomas y refactorización de código. No necesita configuración para la depuración y presenta un editor de HTML, CSS y JavaScript, además de php (JetBrains, 2016).

1.7.7 Lenguaje de programación: PHP 5

PHP 5 (acrónimo recursivo de PHP: Hipertexto Preprocessor) es un lenguaje abierto de propósito general, ampliamente utilizado para el desarrollo web y puede ser embebido en páginas HTML. Lo que distingue a PHP es que el código se ejecuta en el servidor, lo que genera el HTML que se enviara al cliente. El cliente recibirá los resultados de ejecutar esa secuencia de comandos, pero no sabrá cuál será el código subyacente[3]. Incluso puede configurar el servidor web para que procese todos los archivos HTML con PHP, y luego no hay manera de que los usuarios puedan saber lo que se ejecutó en el servidor (PHPnet, 2016).

1.7.8 JavaScript

Es un lenguaje de programación interpretado. Se define como orientado a objetos. Maneja objetos dentro de la página web y sobre ese objeto se pueden definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades y modificación de archivos. (Valdés, 2007).

1.7.9 Hojas de Estilo en Cascada: CSS 3

CSS es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. CSS se crea para separar el contenido de forma, que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas. (Álvarez, 2008).

CSS3 es el último estándar de CSS, siendo completamente compatible con versiones anteriores de CSS. CSS3 se ha dividido en "módulos" que contienen las especificaciones

[3] Que subyace por debajo de algo

de versiones anteriores además de las nuevas. Algunos de los módulos más importantes son:

- Selectores
- Modelo de caja
- Fondos y bordes
- Valores de imagen y contenido sustituido
- Efectos de texto
- Transformaciones 2D y 3D
- Animaciones
- Disposición de columnas múltiples

La mayor parte de las nuevas propiedades de CSS3 se implementan en los navegadores modernos. (w3schools, 2016)

1.7.10 Lenguaje de Marcado de Hipertexto: HTML 5

HTML, siglas de Hyper Text Markup Language (lenguaje de marcado de hipertexto), en su versión 5, es usado para describir la estructura y el contenido en forma de texto.; describiendo hasta un cierto punto, la apariencia de un documento. Además, se ha convertido en el formato más fácil para la creación de páginas web debido a su sencillez y no hay que compilar el código para ver si funciona. Se puede ver en forma inmediata el resultado del trabajo; siendo usado además para complementar el texto con objetos tales como: imágenes, tablas entre otros (Eguiluz, 2010).

1.8 Especificación de requisitos

La especificación de requisitos del software es una descripción completa del comportamiento del sistema a desarrollar. Incluye la descripción de todas las interacciones que se prevén que los usuarios tendrán con el software (Pytel, 2011).

1.9 Requerimientos del software

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones. Estos reflejan las necesidades de los clientes que ayuden a

resolver determinado problema. Los requerimientos de software se clasifican en: funcionales y no funcionales (Sommerville, 2005).

1.10 Patrones de diseño

Un patrón de diseño nombra, motiva y explica de forma sistemática un diseño general que afronta un problema de diseño recurrente en los sistemas orientados a objetos. Son la base para la búsqueda de soluciones a problemas en el desarrollo del diseño del software (Hall, 2005).

1.11 Métricas para la evaluación de Diseños Orientado a Objeto

Las métricas, dentro del contexto de la ingeniería del software, son un grupo de medidas efectuadas sobre programas, documentación, procesos de desarrollo o al mantenimiento, que permite una comparación previa con valores que proporcionan una indicación cuantitativa de extensión, cantidad, dimensión, capacidad y tamaño de algunos atributos de un proceso o producto (Pressman, 2006).

Para medir el diseño existen numerosas métricas pero entre las más referenciadas se pueden encontrar la familia de métricas de diseño orientado a objetos propuestas por Chidamber y Kemerer, y las orientadas a clases de Lorenz y Kidd.

Chidamber y Kemerer (Chidamber, y otros, 1994) establecen 6 métricas basadas en la programación orientada a objetos.

- **Métodos ponderados por clase:** su objetivo es medir la complejidad de una clase.
- **Profundidad del árbol de herencia:** mide la distancia desde una clase a la raíz del árbol de herencia
- **Número de hijos:** mide el número de subclases inmediatas a una clase en el árbol de herencia, y mide la anchura de una jerarquía de clases.
- **Acoplamiento entre objetos:** mide el acoplamiento de la dependencia de una clase con varias clases del sistema. Existe dependencia cuando la clase utiliza métodos o atributos de otras clases.
- **Respuesta de una clase:** obtiene el tamaño del conjunto de respuesta de una clase.
- **Falta de cohesión en los métodos:** es el número de grupos de métodos de una clase que no acceden a atributos comunes de la misma.

Lorenz y Kidd dividen las métricas basadas en clases en tres grupos (Lorenz, 1994):

- **Métricas de tamaño de la clase:** se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad.
- **Métricas de herencia:** se centra en la forma en que se reutilizan las operaciones en la jerarquía de clases.
- **Métricas de las características internas de las clases:** examinan la cohesión y asuntos relacionados con el código.

Se realizó un estudio de las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC) propuestas por Lorenz y Kidd, teniendo en cuenta que se validará el diseño solo a través de las métricas de tamaño de las clases.

Tamaño Operacional de Clases

La métrica TOC es una métrica especializada en el diseño orientado a objeto, midiendo características de clases además de las correspondientes a comunicación y colaboración. Está dada por el número de métodos y atributos asignados a una clase para evaluar los siguientes atributos de calidad: responsabilidad, complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que, para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización (Lorenz, 1994). El tamaño operacional de una clase se puede determinar empleando medidas para saber el número total de operaciones que contiene.

Tabla 1.Atributos que evalúa TOC. **Fuente:** Elaboración de autor

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Para estos atributos de calidad están definidos los siguientes criterios y categorías de evaluación:

Tabla 2.Rango de valores para los criterios de la métrica TOC. **Fuente:** Elaboración de autor

Atributo	Categorías	Criterios
Responsabilidad	Baja	<= Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio
Complejidad de implementación	Baja	<= Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio
Reutilización	Baja	> 2 * Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	<= Promedio

Relaciones entre clases (RC)

La métrica RC permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas (Lorenz, 1994). A continuación se presentan los atributos de calidad que mide esta métrica.

Tabla 3. Atributos de calidad que mide RC. **Fuente:** Elaboración de autor

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 4.Criterios y categorías para evaluar la métrica RC. **Fuente:** Elaboración de autor

Atributo	Categorías	Criterios
Acoplamiento	Ninguna	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<= Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio
Reutilización	Baja	> 2 * Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	<= Promedio
Cantidad de pruebas	Baja	<= Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	> 2 * Promedio

1.12 Conclusiones parciales

- El análisis de los principales conceptos asociados al dominio del problema facilitó una comprensión de la investigación.
- El establecimiento del estado del arte permitió analizar algunas de las tendencias actuales de los generadores de reportes tanto nacionales como internacionales, que sirvieron de base para estructurar el componente a desarrollar.
- El estudio y análisis de la metodología AUP-UCI, permitió guiar el proceso de desarrollo de software.
- Se realizó un análisis detallado de las herramientas y tecnologías propuestas por el Departamento de Desarrollo de Componente para lograr una mejor comprensión de cada una de ellas.

Capítulo 2: Descripción de la propuesta de solución

2.1 Introducción

En el presente capítulo se realiza una descripción de la propuesta de solución a través de los requisitos funcionales y no funcionales, así como las particularidades del diseño. Se muestran aspectos esenciales de la arquitectura del sistema y los patrones de diseño empleados. Además; se hace referencia a los productos de trabajo de la fase Ejecución: Modelado del negocio, Requisitos, Análisis y diseño.

2.2 Propuesta de solución

Una vez concluido el estudio de algunos generadores de reportes y el análisis del ambiente de desarrollo propuesto por el Departamento de Desarrollo de Componente, se decide desarrollar un componente para el marco de trabajo Bosón que permita la creación de reportes de forma rápida. Los pasos que debe seguir el desarrollador para la creación de un reporte se muestran en el Manual de Usuario Reportes ver (Anexo 1).

2.3 Técnicas de captura de requisitos

Existen varias técnicas que posibilitan la captura de los requisitos de software Para el desarrollo de la solución propuestas las utilizadas fueron la **entrevista** y la **tormenta de idea**.

La **entrevista** permitió obtener conocimiento de las necesidades del cliente para la correcta elaboración del producto. Mientras que el uso de la técnica **tormenta de idea**, permitió acumular nuevas ideas e información proporcionada por varias personas involucradas.

Las personas entrevistadas fueron las siguientes: analista y desarrolladores involucrados del departamento de Desarrollo de Componente del centro CEIGE.

2.4 Requerimientos del software.

2.4.1 Requisitos funcionales

Según la IEEE 610.32 los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Dependen de las necesidades de los clientes, constituyendo un paso fundamental para la satisfacción de dichos clientes y proporcionando una mejor comprensión para el equipo de desarrollo (IEEE, 1998).

Como resultado de la aplicación de las técnicas de captura de requisitos empleadas se obtuvieron un total de 21 requisitos funcionales. A continuación, se representan los requisitos funcionales del sistema en la Tabla 1. Los mismos serán identificados con las siglas RF_más_el_número_del_requisito (Ej. RF_1.). Estos requisitos permiten definir las funcionalidades del componente Reporte enfocándose en las necesidades y aspiraciones de los desarrolladores.

Tabla 5.Descripción de los Requisitos Funcionales. **Fuente:** Elaboración del autor.

Generar Reportes		
Identificador	**Nombre**	**Descripción**
RF_1	Adicionar Reporte	Este requisito se encarga de que el sistema adicione un reporte.
RF_2	Listar entidades y atributos de BD	Este requisito se encarga de que el sistema muestre un listado de todas las entidades y atributos que se encuentran almacenados en la Base de Datos.
RF_3	Adicionar atributos a la consulta	Este requisito se encarga de que el sistema adicione atributos.
RF_4	Listar atributos de la consulta	Este requisito se encarga de que el sistema muestre un listado de todos los atributos.
RF_5	Modificar atributos a la consulta	Este requisito se encarga de que el sistema modifique los parámetros de los atributos.
RF_6	Eliminar atributos a la consulta	Este requisito permite que el sistema elimine un atributo.

RF_7	Crear Condición[4]	Este requisito se encarga de que el sistema cree las condiciones.
RF_8	Modificar Condición	Este requisito se encarga de que el sistema modifique una condición.
RF_9	Eliminar Condición	Este requisito permite que el sistema elimine condiciones.
RF_10	Construir Consulta DQL[5]	Este requisito permite que el sistema construya automáticamente la consulta DQL.
RF_11	Adicionar Filtro	Este requisito se encarga de que el sistema adicione un filtro.
RF_12	Modificar Filtro	Este requisito se encarga de que el sistema modifique un filtro.
RF_13	Buscar Filtro	Este requisito se encarga de que el sistema busque un filtro.
RF_14	Eliminar Filtro	Este requisito se encarga de que el sistema elimine filtros.
RF_15	Construir Plantilla	Este requisito se encarga de que el desarrollador edite la plantilla.

[4] Son los filtros que se le aplican a los atributos, que sería el equivalente a los WHERE () en base de datos. Restricciones que desee que la información cumpla.

[5] Lenguaje de Consulta de Doctrine (DQL -Doctrine Query Language) .DQL o bien puede ser escrito como una cadena o generarse usando un poderoso objeto.

RF_16	Mostrar Vista Previa de Plantilla	Este requisito se encarga de que el sistema muestre una vista previa del reporte en el formato HTML.
RF_17	Listar Reporte	Este requisito se encarga de que el sistema muestre un listado de los reportes.
RF_18	Modificar Reporte	Este requisito se encarga de que el sistema modifique un reporte.
RF_19	Eliminar Reporte	Este requisito se encarga de que el sistema elimine reportes.
RF_20	Buscar Reporte	Este requisito se encarga de que el sistema realice una búsqueda de un reporte.
RF_21	Ejecutar Reporte	Este requisito se encarga de que el sistema muestre el reporte en el formato escogido por el desarrollador.

Historias de usuarios

Las historias de usuarios (HU), según la metodología de software empleada en la presente investigación, se define como una de las formas que se utilizan para encapsular requisitos. Esta forma se evidencia en el cuarto escenario condicionado por el Modelado del Negocio (Sánchez, 2014). En los anexos se mostrarán las HU Adicionar Filtro, Listar atributos de la consulta y Eliminar Reporte. Los restantes se encuentran en el expediente de proyecto. A continuación se muestra un ejemplo de una Historia de Usuario correspondiente al RF_8 Modificar Condición.

Tabla 6.Descripción HU_8:Modificar_Condición. **Fuente:** Elaboración del autor.

Número: 8	**Nombre del requisito:** Modificar condición.
Programador: Yania Lisbet Bastida Cantero	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 28 horas
Riesgo en Desarrollo: Poca experiencia de los estudiantes en las tecnologías de desarrollo del proyecto.	**Tiempo Real:** N/A
Descripción: Este requisito se encarga de modificar la condición. El usuario selecciona una de las condiciones listadas y modifica los atributos siguientes: - Campos - Filtros	
Observaciones: N/A **Prototipo de interfaz:**	

Figura 1: Modificar condición

Validación de los requisitos funcionales.

Con el objetivo de verificar si los requisitos del software obtenidos definen el sistema que el cliente desea, se llevó a cabo un proceso de validación de los mismos, para el cual se emplearon las siguientes técnicas:

- **Revisiones de los requisitos**: la revisión de las especificaciones de los requerimientos del proceso fue realizada con la analista principal del proyecto. Con la utilización de esta técnica se valida que no existan errores en el contenido, malas interpretaciones o información incompleta, dando como resultado que fueran aprobados los requisitos descritos. Se realizaron dos iteraciones obteniéndose como resultado que en la primera iteración se capturaron 6 requisitos funcionales y en la segunda iteración se incrementaron 14 requisitos funcionales quedando estos descritos y validados en la segunda iteración.
- **Construcción de prototipos**: permite al usuario tener una visión de la estructura de la interfaz del sistema junto a sus funcionalidades; además de verificar que fueron esos los requisitos propuestos. El usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

2.4.2 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Están vinculados generalmente a requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer se puede determinar con facilidad cómo ha de comportarse y qué cualidades debe tener (IEEE, 1998).

Para el desarrollo del componente se definieron un total de 14 RNF los que fueron clasificados en usabilidad, disponibilidad, software, hardware. Estos elementos no funcionales, constituyen los requisitos de la aplicación a implementar y fueron validados mediante la definición en conjunto con el cliente. La tabla de RNF se muestra a continuación.

Tabla 7.Descripción de los Requisitos No Funcionales. **Fuente:** Elaboración de autor.

Identificador	Clasificación	Requisito
RNF_1	Usabilidad	• Todos los mensajes de error del sistema deberán incluir una descripción textual del error.

		• Los campos de cada interfaz tendrán títulos asociados a su función. • Los errores cometidos por los usuarios les serán notificados. • El sistema mostrará las opciones desactivadas siempre que no se hayan cumplido las condiciones previas para su activación. • La confirmación de la entrada de datos deberá poder hacerse mediante el uso de mouse.
RNF_2	Disponibilidad	• El componente para la creación de reportes estará disponible de forma permanente y funcionará sin necesidad de intervención del administrador.
RNF_3	Software	Para el cliente: • Navegador Mozilla Firefox 40.0 o superior. Para el servidor: • Sistema operativo Linux en cualquiera de sus distribuciones. • Un servidor Apache 2.4 o superior con módulo PHP 5.4 disponible o superior.
RNF_4	Hardware	Para el servidor los requerimientos mínimos deben ser: • Procesador Core-i3 a 2.2 GHz de velocidad de procesamiento con 4Gb de memoria RAM. • 40Gb de espacio libre en disco duro. • Tarjeta de red. Para el cliente los requerimientos mínimos deben: • Procesador Pentium III a 1.0 GHz con 1 Gb de memoria RAM. • Tarjeta de red.

2.5 Patrón Arquitectónico Modelo-Vista-Controlador

El desarrollo del componente Reporte está definido a partir del marco de trabajo Symfony2, el cual está basado en el patrón arquitectónico **Modelo-Vista-Controlador (MVC).** Este patrón favorece a las aplicaciones que manejan gran cantidad de datos, ya que estas

necesitan una separación de conceptos para facilitar la programación en diferentes capas de manera paralela e independiente.

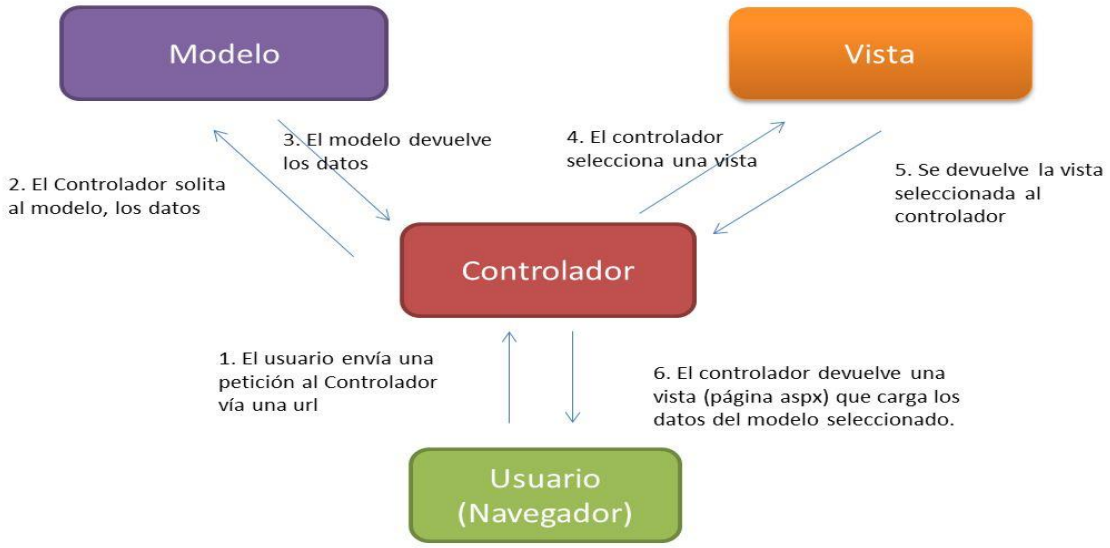

Figura 2. Funcionamiento de patrón Modelo-Vista-Controlado. **Fuente:** asp.net mvc 2| Mi blog técnico

MVC define la separación de estas tres capas:

- **Modelo**: elementos (objetos) que contienen los datos y definen la lógica para manipular dichos datos (Álvares, 2014).
- **Vista**: hace referencia a los elementos que representan algo visible en la interfaz de usuario, por ejemplo, un panel o botones (Álvares, 2014).
- **Controlador**: actúa como un mediador entre los objetos del modelo y la vista. Un objeto Controlador comunica datos de ida y vuelta entre los objetos del modelo y de la vista; realiza todas las tareas específicas de la aplicación, tales como la entrada del usuario o la carga de procesamiento de datos de configuración (Álvares, 2014).

2.6 Patrones de diseño

2.6.1 Patrones de Diseño de Asignación de Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (Grosso, 2011).

Los patrones utilizados en el desarrollo del componente se describen a continuación:

Experto: asigna una responsabilidad al experto en información; es la clase que posee información necesaria para realizar algún evento referente a la misma. Un ejemplo del uso de este patrón se evidencia en la clase ***ReporteController***, pues la misma contiene toda la información referente a los reportes, permitiendo crearlos y gestionarlos.

Creador: es el responsable de la creación o instanciación de nuevos objetos o clases. Un ejemplo del uso de este patrón se evidencia en la clase ***FiltroController*** cuando se crea un objeto del tipo **Filtro.**

Controlador: es el intermediario entre una determinada interfaz y el algoritmo que la implementa. Un ejemplo del uso de este patrón se puede encontrar en la clase ***ReporteController,*** la cual maneja la información almacenada de los reportes.

Bajo Acoplamiento: permite que las clases existentes posean responsabilidades de tal forma que estas no dependan en gran medida de otras, este patrón se evidencias en todas las clase del componente Reporte entre ellas se encuentra: ***ReporteController, FiltroController, QueryController.***

Alta Cohesión: Es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. Indica la relación que existe entre los elementos de un mismo módulo.

2.7 Diagrama de clases de diseño con estereotipos web

El propósito del diagrama de clase con estereotipos web es describir gráficamente las especificaciones de las clases del software que se utilizan en el sistema, así como las relaciones que existen entre estas, representando las interfaces de la aplicación. Para el desarrollo del componente, se elaboraron un total de tres diagramas de clases del diseño con estereotipo web, estos serán mostrados en los Anexos 5,6 y 7.

2.8 Modelo de datos

El modelo de datos es un conjunto de conceptos que sirven para describir la estructura semántica y las relaciones que existen entre las entidades de una base de datos. Su representación es algo importante para el funcionamiento del sistema, ya que muestra los datos que serán contenidos en el mismo (Sommerville, 2005).

A continuación se muestra la estructura de la base de datos del Componente para la creación de reportes del marco de trabajo Bosón, la cual permite realizar la representación

lógica de los datos procesados por el sistema. Este modelo muestra las entidades de datos y los atributos asociados a cada una de ellas.

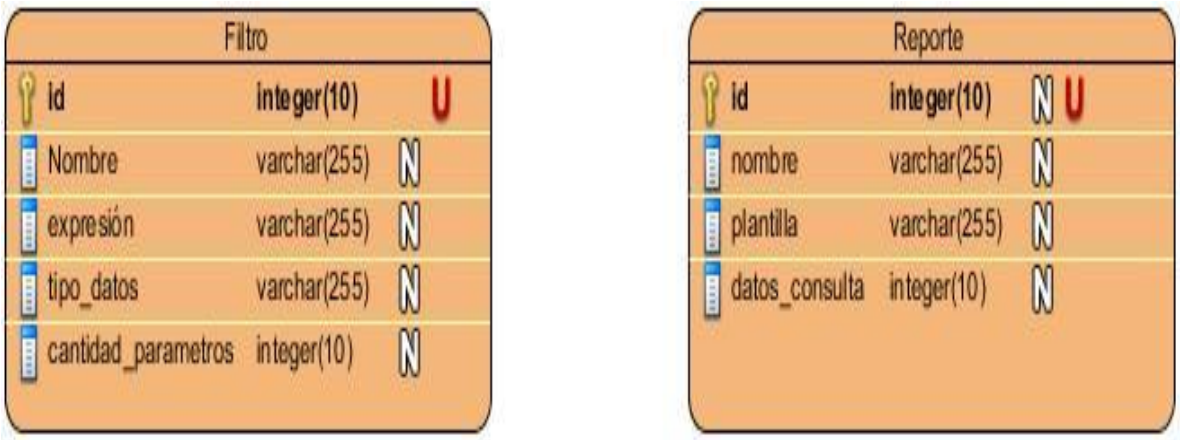

Figura 3. Modelo de Datos. **Fuente**: Elaboración del autor

El modelo de datos está compuesto por dos tablas, las cuales no tienen relación porque las mismas gestionan diferentes informaciones. La tabla **Filtro** está compuesta por los atributos **id**: siendo este el identificador del filtro, **Nombre**: atributo que almacena el nombre del filtro, **cantidad_parametros** este atributo presenta la cantidad de parámetros a comparar, **expresión** guarda la expresión correspondiente al filtro y **tipo_datos** guarda un JSON[6] con los tipos de datos a los que se le puede aplicar el filtro.

La tabla Reporte contiene los atributos **id:** valor por el cual se identifica un reporte, **nombre:** atributo que almacena el nombre del reporte, el atributo **plantilla** guarda los datos de la plantilla que se corresponde con texto HTML y el atributo **datos_consulta** guarda los datos de la consulta en formato JSON.

2.9 Conclusiones parciales

- En el presente capítulo se realizó la descripción del componente mediante el uso de la metodología seleccionada en la etapa de Ejecución.

[6]JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

- Como parte de la realización de la captura de requisitos, se aplicaron técnicas de ingeniería de requisitos descritas en el presente capítulo, definiéndose un total de 21 RF y 14 RNF, los que fueron descritos y agrupados.
- Se realizó un estudio sobre todas las HU definidas como punto de partida para el diseño y realización de los prototipos del sistema, lo que se concretó con la elaboración de las Plantillas de Historias de Usuarios, que aglutinan su descripción, junto al prototipo y los requisitos a los que responde cada una.
- Se realizó un análisis sobre el uso de los patrones arquitectónicos utilizados durante el desarrollo de la aplicación haciendo énfasis en el patrón Modelo-Vista-Controlador.
- Se realizó un análisis sobre la arquitectura a la cual responde el componente, resaltando el uso de los diferentes patrones de diseño dentro de los que se encuentran los GRASP permitiendo que la programación estuviese bien estructurada.
- Con la realización de los diagramas de clases del diseño con estereotipos web se obtuvo una visión más exacta del sistema en términos de implementación, siendo de gran ayuda para el equipo de desarrollo.
- Para describir la estructura semántica y las relaciones que existen entre las entidades de una base de datos se elaboró el Modelo de Datos del componente siendo validados los requisitos funcionales aplicándole un conjunto de técnicas.

Capítulo 3: Implementación y Validación

3.1 Introducción

En el presente capítulo se abordan varias actividades que son imprescindibles para que el software finalice con la calidad requerida, entre las que se encuentran la fase de implementación, validación y las pruebas. Además, se muestran los resultados de las métricas y técnicas empleadas para validar el diseño del componente, se especifican las pruebas que serán realizadas al software y se describen las irregularidades encontradas en las mismas; de ello se generan los principales productos de trabajo definidos por la metodología, correspondientes a la fase de ejecución, en sus disciplinas Implementación y Pruebas internas.

3.2 Implementación

La implementación constituye una de las fases importantes del desarrollo de software. En ella se toman como punto de partida los resultados obtenidos en el diseño. Su importancia reside en que se obtiene como consecuencia un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de software.

3.2.1 Diagrama de componentes

Un diagrama de componentes representa la separación de un sistema de software en componentes físicos (por ejemplo, archivos, módulos, paquetes, base de datos etc.) y muestra la organización y las dependencias existentes entre los componentes (Arizaca, 2009).

En el diagrama de componentes de la Figura 4, se evidencian las relaciones presentes en el componente Reporte. Dicho componente está compuesto por tres paquetes fundamentales: Modelo- Vista- Controlador.

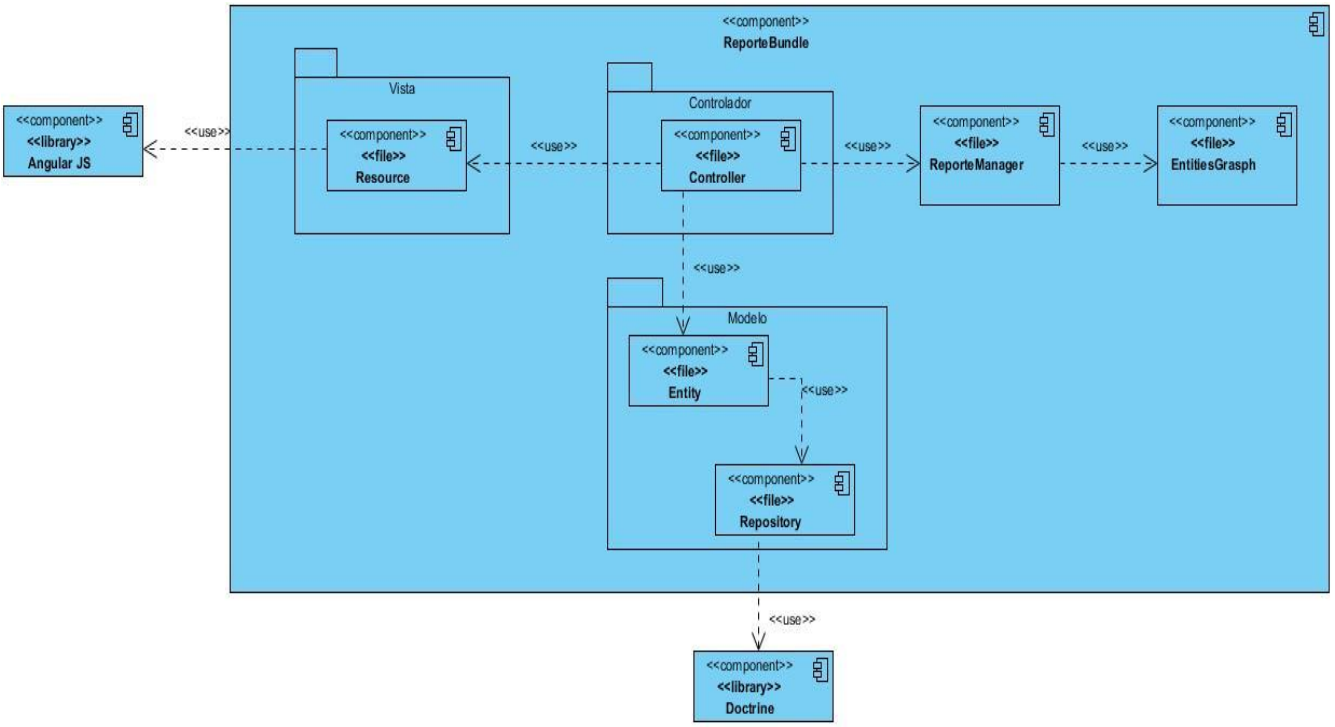

Figura 4. Diagrama de Componentes. **Fuente**: Elaboración del autor.

El componente Reporte contiene el paquete de Controladores, responsable de dar respuesta a las peticiones realizadas por los usuarios, posee una relación con los componentes Vista y Modelo.

El diagrama de componentes está compuesto por un componente principal **Reporte** que representa el componente en general. El paquete **Vista** que utiliza el componente **Resource** el cual agrupa un total de 12 vista las cuales representan las interfaces con las que interactúa el usuario, las mismas utilizan la biblioteca AngularJS. El paquete **Modelo** agrupa los componentes relacionados con las clases de acceso a datos como es el componente **Entity,** que tienes un total de 2 Entidades como son **Reporte** y **Filtro,** utilizan el componente **Repository** y este a su vez utiliza la biblioteca Doctrine para el mapeo de los datos. El componente **Modelo** está relacionado con el controlador para dar respuesta a la vista.

El paquete **Controlador** contiene un total de 5 controladores como son **ReporteController, FiltroController, AdminController, ConfigController, QueryController**. Los mismos están agrupados en el componente Controller relacionado con la lógica del negocio agrupando todos los controladores de la herramienta desarrollada. Su objetivo es obtener

las peticiones del usuario y dar respuesta al mismo mediante su interacción con el modelo. Utiliza un componente **ReporteManager** que se encarga de todo lo referente a las consultas, la vista previa y a la ejecución del reporte. Dicho componente hace uso del componente **EntitiesGraph** encargado de las relaciones entre los atributos (Ejemplo la unión de los atributos). De igual forma, estos componentes pueden ser agrupados y organizados atendiendo a la arquitectura del sistema para lograr un mejor entendimiento de sus funciones, distribución e interacciones entre sí.

3.2.2 Estándar de codificación

Con el objetivo de lograr una estandarización en la programación del componente se decide aplicar el estilo de escritura CamelCase. Las variantes a utilizar son lowerCamelCase y UpperCamelCase. Este facilitará la lectura, comprensión y mantenimiento del código.

A continuación se describe a grandes rasgos las convenciones de nomenclatura.

General:

- Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.
- En todo momento se utilizarán nombres que sean claros, concretos y libres de ambigüedades. Ejemplo: " tipoDatos " y no solamente "tp".
- El nombre de todas las variables y métodos comenzarán con letra minúscula y si está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula. Ejemplo: "createAction (Request $request)".

Identación:

- El contenido siempre se identará con *tabs*, nunca utilizando espacios en blanco.

Clases:

- Los nombres de las clases comenzarán con mayúsculas; si están compuesto por varias palabras, todas las palabras intermedias que lo componen comenzarán con mayúscula. Ejemplo: “ReportController”.
- Los nombres de las clases están escritos en inglés.
- Se utilizará palabras completas, evitando acrónimos y abreviaturas, a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML.

Nombre de variables:

- No se utilizarán nombres de variables que puedan ser ambiguos.
- No se dará nombres sin sentido a variables temporales. Por ejemplo: temp, i, tmp.
- Los nombres de las variables empezarán con minúscula; si está compuesto por varias palabras todas las palabras internas que lo componen comienzan con mayúscula. Ejemplo: “cantidadDParametros”.

3.3 Validación

Con los productos de trabajos obtenidos en la disciplina análisis y diseño se comenzará la fase de implementación, donde, a partir de los resultados obtenidos anteriormente, se implementará el sistema en términos de componentes, ficheros, código fuente y scripts ejecutables. Con el objetivo de certificar esta disciplina se realizó la validación de los resultados obtenidos con el uso de métricas.

3.3.1 Validación del diseño de la aplicación web

Para comprobar la calidad y fiabilidad del diseño del Componente para la creación de reportes del marco de trabajo Bosón fueron aplicadas las métricas proporcionadas por Lorenz y Kidd; ya que se desea validar el diseño basado en clases, haciendo énfasis en las métricas de tamaño de las clases descriptas en el Capítulo 1; entre ellas las TOC (Tamaño Operacional entre Clases) y RC (Relacione entre Clases). Las mismas fueron escogidas ya que no existe herencia en las clases implementadas. A continuación se muestra un resumen de los resultados de la aplicación de ambas métricas.

Tamaño Operacional de Clases (TOC)

Se muestra el resultado de la aplicación de la métrica TOC en las 15 clases que juegan un papel fundamental en el proceso del componente Reporte.

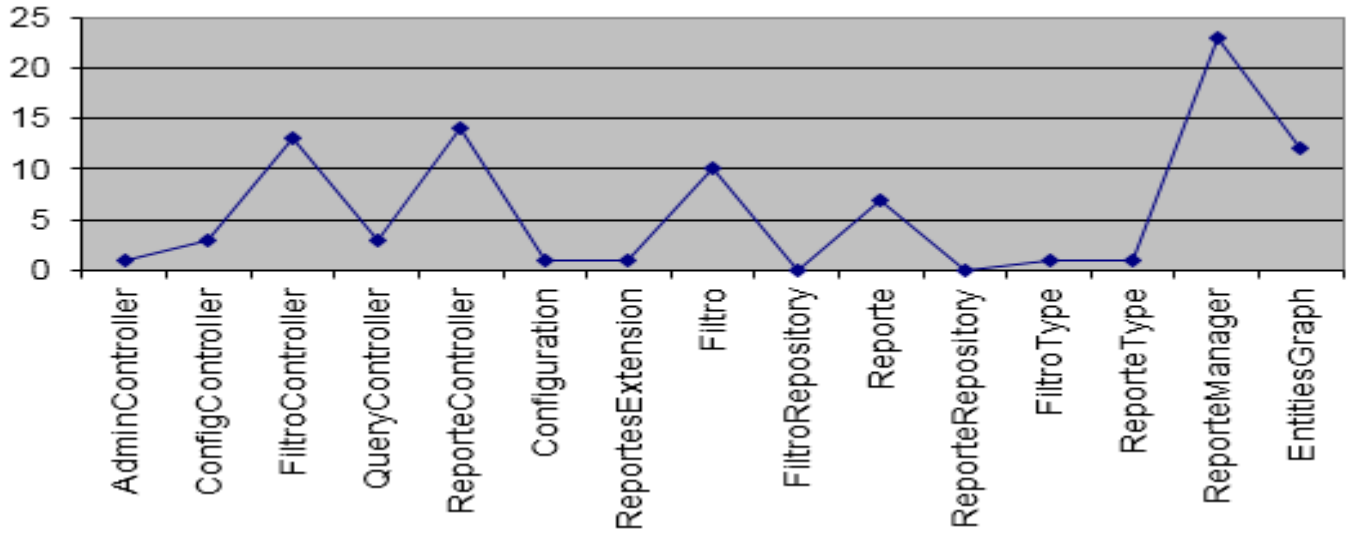

Figura 5. Cantidad de procedimientos por clases. **Fuente**: Elaboración del autor.

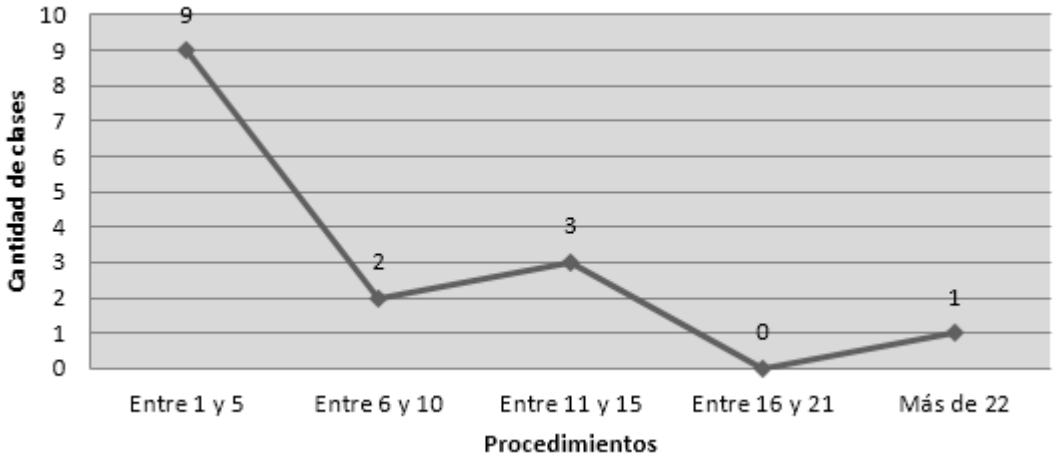

Figura 6. Resumen de cantidad de procedimientos por clases. **Fuente**: Elaboración del autor.

Tabla 8.Valores de las variables de calidad: Responsabilidad, Complejidad y Reutilización. **Fuente:** Elaboración de autor

Criterios	**Responsabilidad**		**Complejidad**		**Reutilización**	
Nivel	Cantidad de Clases	Promedio	Cantidad de Clases	Promedio	Cantidad de Clases	Promedio
Baja	9	60	9	60	3	20

Media	3	20	3	20	3	20
Alta	3	20	3	20	9	60

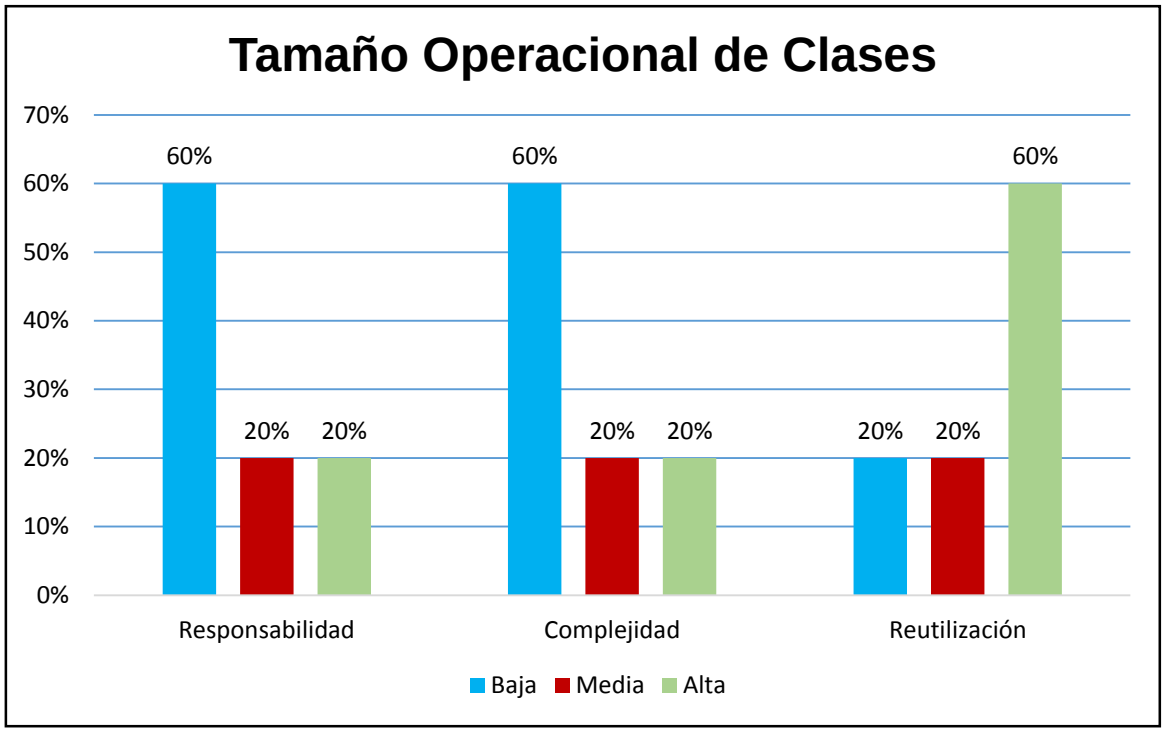

Figura 7. Resultado de las variables: Responsabilidad, Complejidad y Reutilización. **Fuente**: Elaboración del autor.

Luego de aplicada la métrica TOC con umbral de 6 procedimientos (90 procedimientos en total /15clases existentes) se observa que las clases del diseño de la aplicación web no se encuentran sobrecargadas en cuanto a responsabilidades y el nivel de complejidad de las mismas no es muy alto, lo que favorece en gran medida la reutilización de estas.

Relaciones entre Clases (RC)

La métrica RC permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas. Luego de aplicar la métrica RC, se arrojaron los siguientes resultados:

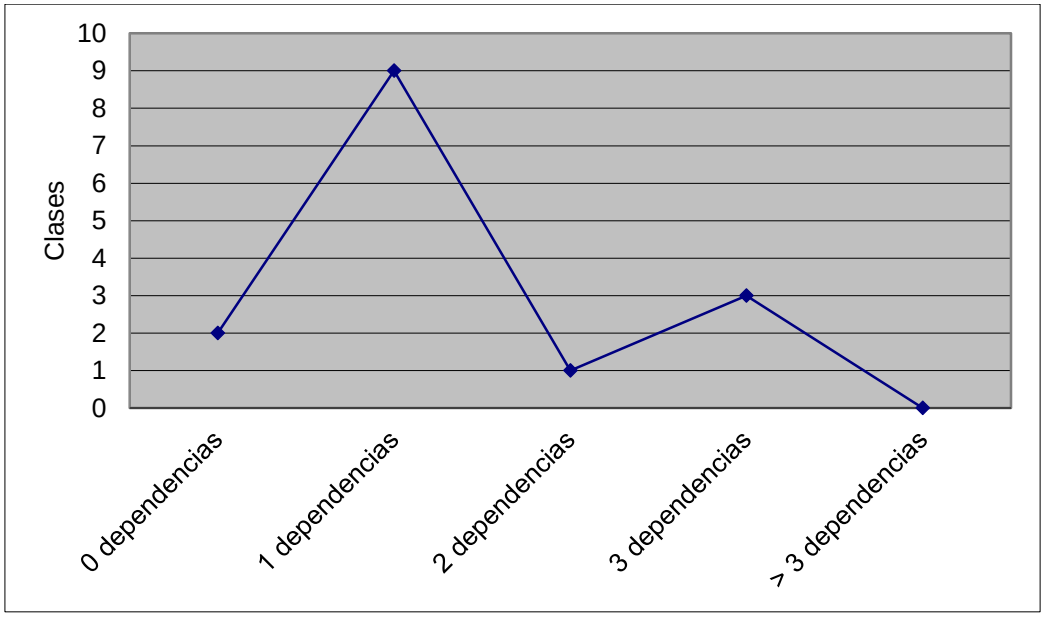

Figura 8. Resumen de Relaciones de uso. **Fuente:** Elaboración del autor.

Tabla 9.Valores de las variables: Acoplamiento, Complejidad, Reutilización y Cantidad de Pruebas.

Fuente: Elaboración de autor

Criterios	**Acoplamiento**		**Complejidad Mant**		**Reutilización**		**Cant. Pruebas**	
Nivel	Cantidad de Clases	Promedio%	Cantidad de Clases	Promedio%	Cantidad de Clases	Promedio%	Cantidad de Clases	Promedio
Ninguno	2	13.3	-	-	-	-	-	-
Bajo	9	60	11	73.33	3	20	11	73.33
Medio	1	6.6	1	6.66	1	6.66	1	6.66
Alto	3	20	3	20	11	73.33	3	20

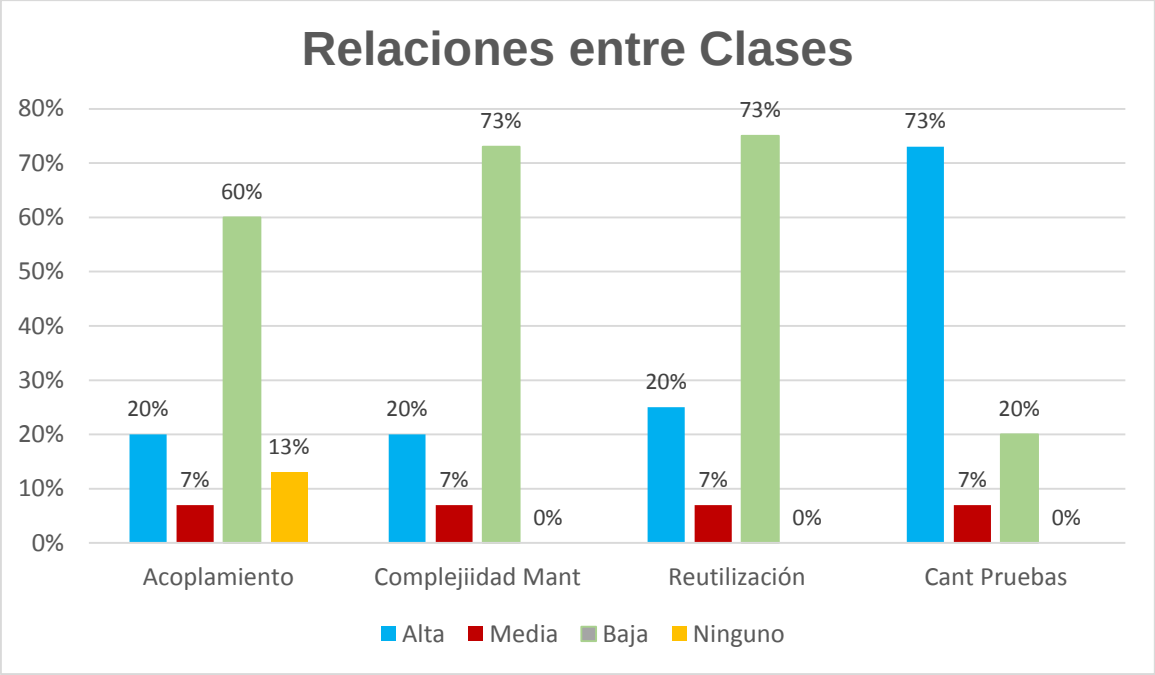

Figura 9. Resultado de las variables Acoplamiento, Complejidad, Reutilización y Cantidad de Pruebas**.**
Fuente: Elaboración del autor.

Una vez aplicada la métrica RC y teniendo en cuenta el umbral definido para validar el diseño, se obtiene como resultado que las clases promueven el bajo acoplamiento; la complejidad de mantenimiento y la cantidad de pruebas no son altas, y en consecuencia, el grado de reutilización es alto.

Los resultados obtenidos de la aplicación de las métricas TOC y RC demuestran que el diseño del componente de Reportes no es complejo, que las clases presentan bajo acoplamiento y un alto grado de reutilización.

3.3.2 Validación de la Investigación

Investigación experimental

Es un proceso sistemático y una aproximación científica a la investigación en la cual el investigador manipula una o más variables; controla y mide cualquier cambio en otras variables. Cuenta con un grupo de control, los sujetos son asignados al azar entre los grupos y el investigador sólo pone a prueba un efecto a la vez. Asimismo, es importante saber qué variable(s) se desean probar y medir (Explorable, 2013).

Existen 3 tipos de diseño experimental (Lucia Negreira, 2010).

- **Cuasiexperimentos** son particularmente útil para estudiar problemas en los cuales no se puede tener control absoluto de las situaciones, pero se pretende tener el mayor control posible, aun cuando se estén usando grupos ya formados. Es decir, el cuasiexperimento se utiliza cuando no es posible realizar la selección aleatoria de los sujetos participantes en dichos estudios. Por ello, una característica de los cuasiexperimentos es el incluir "grupos intactos", es decir, grupos ya constituidos.
- **Experimento puro** es aquel en el que se manipula una o varias variables independientes para observar sus cambios en las variables dependientes en una situación de control (Carolina, 2010).
- **Pre-Experimento** consiste en analizar una sola variable, por lo que existe la manipulación de la variable independiente. Los pre-experimentos se clasifican en:
  - Estudio de caso de una sola medición ,
  - Diseño de preprueba-postprueba con una sola medición.
  - Comparación de grupo estático (Tareas, 2011).

A continuación se describen las variables presentes en la investigación.

Variable Dependiente: el tiempo de creación de un reporte.

Variable Independiente: desarrollo de un generador de reportes para el marco de trabajo Bosón.

Para validar dicha investigación se aplicó el diseño experimental Pre-experimento, utilizando el método de la preprueba-postprueba con un solo grupo, aplicando la técnica de medición. El diseño de dicho pre-experimento tiene la siguiente estructura:

G: Grupo de sujeto.

El Grupo de sujeto escogido fue una muestra de 5 personas de un grupo de 10 que pertenecen al Departamento Desarrollo de Componente.

X: Tratamiento

El tratamiento a aplicar es la técnica de medición (Cálculo del Tiempo Promedio).

O1: pre –prueba

Para realizar el proceso de generar un reporte de los estudiantes de la Facultad 3 mayores de 20 años en formato PDF con un encabezado que diga “Estudiantes de la Facultad 3

mayores de 20 años". Para la realización de este proceso, el desarrollador contará con el entorno que se describe a continuación:

- marco de trabajo Symfony v2.7 configurado.
- 2 Entidades de Doctrine las cuales son: Facultad (id, numero, nombreDecano), Estudiante (id, nombre, apellidos, ci, edad).
- Un bundle para exportar a formato PDF.
- Entorno Integrado de Desarrollo PhpStorm v9.1

O2: post-prueba

Para realizar el proceso de generar un reporte de los estudiantes de la Facultad 3 mayores de 20 años en formato PDF con un encabezado que diga "Estudiantes de la Facultad 3 mayores de 20 años". Para la realización de este proceso, el desarrollador contara con el entorno que se describe a continuación:

- Componente Reportes.
- Manual de usuario del componente Reporte.
- Entorno Integrado de Desarrollo PhpStorm v9.1

Con el uso de la técnica se realizaron un total de cinco mediciones a varios expertos en diferentes momentos sin utilizar el componente y utilizando el componente. La estimación del tiempo se realizó en minutos. Los resultados fueron los siguientes:

Tabla 10.Medición del tiempo: Creación de Reporte (antes del componente). **Fuente:** Elaboración de autor

Personas	**Tt (Tiempo Total)**
Persona A	59 minutos
Persona B	47 minutos
Persona C	45 minutos
Persona D	40 minutos
Persona E	32 minutos
Tiempo Promedio	44.6 minutos

Tabla 11.Medición del tiempo: Creación de Reporte (Con el Componente). **Fuente:** Elaboración de autor

Personas	Tt (Tiempo Total)
Persona A	7 minutos
Persona B	5 minutos
Persona C	4 minutos
Persona D	3 minutos
Persona E	2 minutos
Tiempo Promedio	4.2 minutos

La evaluación de las mediciones anteriores permite demostrar que existe una reducción aproximada del tiempo en la ejecución del proceso de 40.4 minutos. El componente muestra una gran rapidez en el proceso de creación de un reporte por lo que se logró minimizar el tiempo, obteniéndose reportes de manera rápida.

3.4 Pruebas de software

3.4.1 Pruebas unitarias

Una prueba unitaria es una unidad estructural de código que es aplicada solamente a pequeñas partes del código. Para comprobar que cada sentencia de código se ejecuta al menos una vez, se realizaron pruebas al código de las funcionalidades más complejas desde el punto de vista de la programación en cada uno de los métodos del componente.

Pruebas de funcionalidad

El principal objetivo de esta prueba es comprobar que las funcionalidades de la aplicación se realizan de forma correcta y responden a las necesidades del cliente, apoyándose en los casos de pruebas diseñados para cada funcionalidad, evidenciadas en el producto de trabajo Diseños de casos de Prueba, generado en la disciplina pruebas internas de la etapa de ejecución de la metodología AUP-UCI.

Método de caja negra

El método de caja negra se centra principalmente en los requisitos funcionales del software permitiendo obtener un conjunto de condiciones de entrada que ejerciten completamente

todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos (Oré B, 2009).

El método de la caja negra define técnicas; dentro de ellas se encuentra la **técnica de partición de equivalencia.** El objetivo de aplicar esta técnica al componente es para comprobar que las funcionalidades de la aplicación se realizan de forma correcta, utilizando para esta prueba los diseños de casos de prueba realizados.

Diseños de casos de prueba

El objetivo del diseño de casos de prueba es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para diseñar un caso de prueba, se selecciona una característica del sistema o componente que se está probando, luego se selecciona un conjunto de entradas que ejecutan dicha característica, obteniéndose las salidas esperadas o rangos de salida (Sommerville, 2005).Se elaboraron un total de 21 diseños de casos de pruebas los cuales se encuentran el expediente de proyecto.

Casos de pruebas diseñados

A continuación se muestran un ejemplo de un diseño de caso de prueba que ha sido realizado con el fin de observar los resultados aplicando el método de prueba de caja negra y utilizando la técnica de partición de equivalencia.

Descripción general

El requisito funcional referente a Adicionar Filtro se inicia cuando el desarrollador desea adicionar un filtro. Para ello debe seleccionar la opción de Filtro, presionar el botón adicionar filtro, y luego introducir los datos solicitados. El requisito funcional termina cuando finalmente el sistema muestra un mensaje de confirmación que se ha adicionado un filtro.

Condiciones de ejecución:

El usuario debe estar autenticado en el sistema.

El usuario debe seleccionar el componente Reporte.

El usuario debe seleccionar la opción de Filtro.

Tabla 12.Diseño de Caso de Prueba. **Fuente:** Elaboración de autor

Escenario	Descripción	Flujo Central
EC: 1.1 Adicionar Filtro Correctamente opción Acepta	Se adicionan filtros al sistema.	1. Se selecciona la opción Adicionar en la parte superior derecha de la interfaz. 2. Se levanta la interfaz Adicionar nombre. 3. Se inserta el nombre del filtro. 4. Presiona el botón Aceptar. 5. El sistema valida los datos. 6. Se levanta otra interfaz para introducir los datos del filtro. 7. Se configuran los datos. 8. Se presiona el botón Aceptar. 9. El sistema valida los datos. 10. Se muestra un mensaje indicando el éxito de la operación. 11. Se cierra el formulario.
EC: 1.2 Cancelar Operación de Adicionar Filtro.	El usuario desea cancelar la operación.	1. Se selecciona la opción Adicionar en la parte superior derecha de la interfaz. 2. Se levanta la interfaz Adicionar nombre. 3. Se presiona el botón Cancelar para abortar la operación. 4. Se cierra el formulario.
EC: 1.3 Aplicar Operación de Adicionar Filtro.	El usuario desea adicionar un filtro y la ventana de los datos se queda abierta.	1. Se selecciona la opción Adicionar en la parte superior derecha de la interfaz. 2. Se levanta la interfaz Adicionar nombre. 3. Se inserta el nombre del filtro. 4. Presiona el botón Aceptar. 5. El sistema valida los datos.

		6. Se levanta otra interfaz para introducir los datos del filtro. 7. Se configuran los datos. 8. Se presiona el botón Aplicar. 9. El sistema valida los datos. 10. Se muestra un mensaje indicando el éxito de la operación. 11. Se mantiene abierta la ventana para continuar adicionando reportes.

Tabla 13. Descripción de variables. **Fuente:** Elaboración de autor

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Nombre del Filtro	Texto	No	Caracteres alfanuméricos
2	Cantidad de Parámetros	Texto	No	Caracteres numéricos
3	Tipo de Datos	Combobox	No	Selección
4	Expresión	Texto	No	Puede estar en forma de texto.

En las pruebas de caja negra realizadas, a los diseños de casos de pruebas. En la primera iteración se detectaron 16 no conformidades a las cuales se les dio solución. En la segunda se detectaron 8 no conformidades las cuales fueron resueltas en su totalidad. Todas tuvieron solución en un tiempo máximo de 3 días, como se muestra en la Figura 10**.**

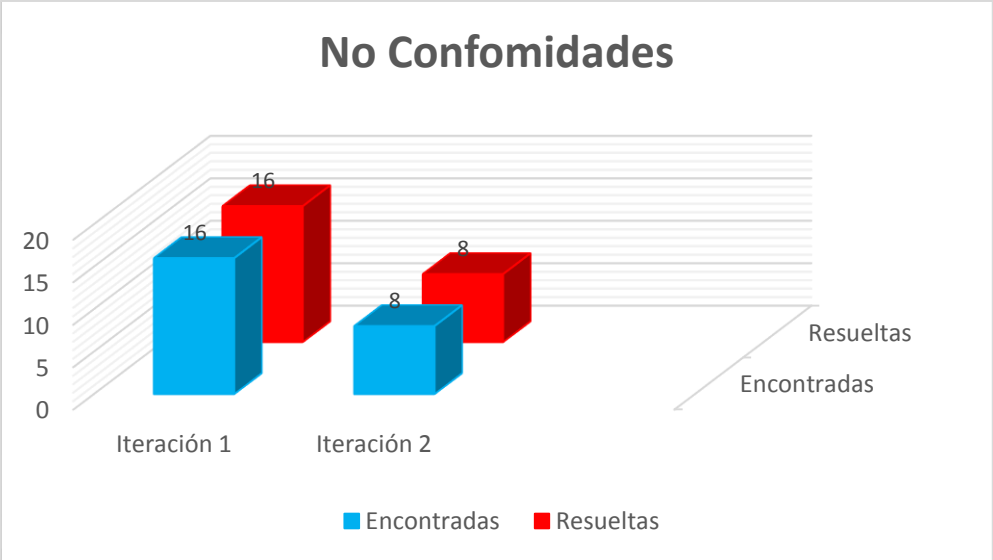

Figura 10. Relación de las No Conformidades de las prueba de caja negra. **Fuente:** Elaboración del autor.

Método de caja blanca

El método de caja blanca está dirigido a las funciones internas de un software, utilizándose la técnica del **camino básico** para la presente investigación. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Presman, 2007).

La técnica fue aplicada a los métodos con mayor complejidad funcional, ya que los restantes eran métodos lineales. El método utilizado como ejemplo fue el updateAction ($id) perteneciente a la clase ReporteController como base para realizar el procedimiento anteriormente descrito.

A continuación, se muestra este método en la Figura 11:

```
public function updateAction(Request $request, $id)
{
    $em = $this->get('doctrine.orm.entity_manager');

    $entity = $em->getRepository('ReportesBundle:Reporte')->find($id);
    $response = new Response();

    if (!$entity) {
        $response->setContent('Unable to find Reporte entity.');
        $response->setStatusCode(Response::HTTP_NOT_FOUND);
    } else {
        $editForm = $this->createEditForm($entity);
        $editForm->handleRequest($request);

        if ($editForm->isValid()) {
            $em->flush();

            $response->setContent('The Usuario was updated successfully.');
        } else {
            $errors = $this->getAllErrorsMessages($editForm);

            $response->setContent($this->serialize($errors));
            $response->setStatusCode(Response::HTTP_INTERNAL_SERVER_ERROR);
        }
    }

    return $response;
}
```

Figura 11. Método utilizado para la aplicación de la técnica camino básico**. Fuente:** Elaboración del autor.

Para aplicar la técnica del camino básico se realizaron una serie de pasos que se describen a continuación:

A partir del método presentado se construye el grafo de flujo de control; el cual está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

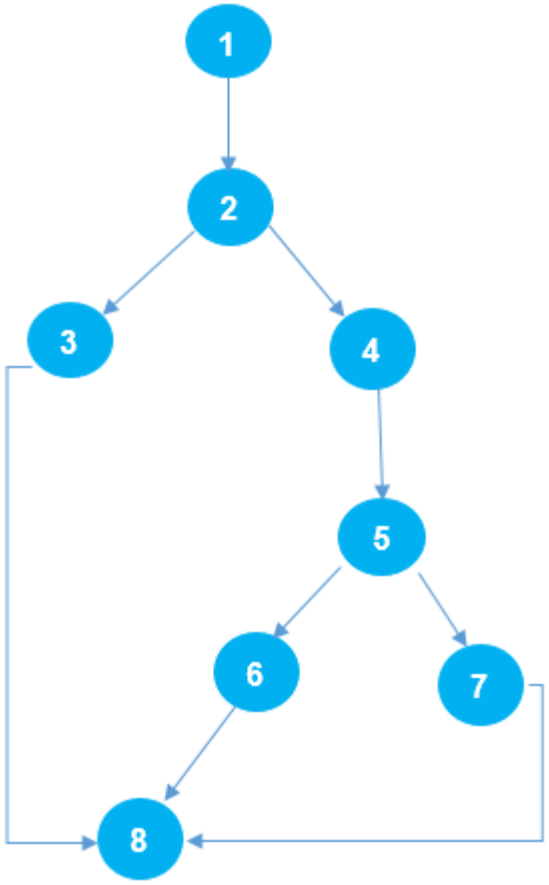

Figura 12. Grafo de flujo asociado al camino básico. **Fuente:** Elaboración del autor.

1-una vez construido el grafo de flujo se procede a calcular la complejidad ciclomática del grafo mediante las siguientes formas:

- El número de regiones corresponde a la complejidad ciclomática. V(G) = 3
- La complejidad ciclomática, V(G), de una gráfica de flujo, G, se define como V(G)= a-n+2 siendo a en número de aristas y n el número de nodos V(G) =9-8+2=3
- V (G) = c + 1, siendo c el número de nodos de condiciónV(G) =2+1=3

2-Se determina un conjunto básico de caminos independientes, el valor de V (G) proporciona el número de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los 3 caminos siguientes:

Camino básico # 1: 1-2-3-8.

Camino básico # 2: 1-2-4-5-6-8

Camino básico # 2: 1-2-4-5-7-8

3-Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico, cada camino independiente es un caso de prueba a realizar. En este caso se obtuvieron 3 caminos básicos, por tanto, se hace necesario la confección de igual número de casos de prueba, para aplicar las pruebas a este método (Presman, 2007).

Tabla 14.Casos de prueba asociados a la técnica de camino básico. **Fuente:** Elaboración de autor

Número de camino	Descripción	Entrada	Resultados esperados
1	No se encontró el identificador	Identificador, Respuestas	Muestra el mensaje “No se encontró el reporte”.
2	Se encuentra el identificador	Identificador, Respuestas	Muestra el mensaje “El reporte se ha modificado.
3	Se encuentra el identificador	Identificador, Respuestas	Muestra el mensaje que “El formulario contiene errores “

Una vez aplicada la prueba de caja blanca mediante la técnica del camino básico al método updateAction ($id), se obtuvo como resultado que se ejecutan al menos una vez cada camino independiente.

3.4.2 Pruebas de Aceptación

Las pruebas de aceptación permiten que el cliente valide todos los requisitos propuestos. Las pruebas las realiza el usuario final en lugar del responsable del desarrollo del sistema. Una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas, de hecho, la prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema (Pressman, 2006).

Para comprobar la calidad del componente Reportes se realizaron 3 iteraciones por el arquitecto del departamento de Desarrollo de Componentes del Centro CEIGE: **Abraham**

Calas. Las no conformidades (NC) encontradas se clasificaron en NC de la aplicación. En la siguiente figura se muestran las iteraciones realizadas y la cantidad de NC encontradas en cada iteración.

Figura 13. Relación de las No Conformidades. **Fuente:** Elaboración del autor.

3.5 Conclusiones parciales

En este capítulo se realizó la validación del componente Reporte arrojando resultados satisfactorios como:

- La confección del diagrama de componentes permitió conocer cómo interactúan los diferentes componentes de implementación.
- El uso de la notación lowerCamelCase y UpperCameCalse, como estándar de codificación, garantizó que la implementación se realizara de manera organizada, permitiendo un mayor entendimiento de la misma.
- Se realizaron pruebas de aceptación y unitaria, demostrando que las funciones del componente son efectivas, producen un resultado satisfactorio y el sistema cumple con los objetivos trazados.
- Se realizó la validación de la investigación mediante un pre-experimento, arrojando este resultado positivo, demostrando que el desarrollo del componente minimizar el tiempo de creación de un Reporte.

Conclusiones

A partir del desarrollo de la presente investigación se arribaron a las siguientes conclusiones:

- El análisis y estudio de las tecnologías y tendencias necesarias propuestas por el proyecto para el desarrollo del componente permitió obtener un producto con calidad.
- El uso del marco de trabajo Symfony2 ayudó a confeccionar una propuesta de solución que se ajustará a las pautas de codificación definidas por el proyecto para el desarrollo del componente.
- Se definieron 21 requisitos funcionales, 14 no funcionales los cuales fueron agrupados en 4 clasificaciones (Usabilidad, Disponibilidad, Software, Hardware), 21 casos de uso y 21 Historias de Usuario con sus respectivas descripciones, permitiendo especificar las características del componente y sirviendo de guía para los flujos y fases posteriores.
- Para comprobar la calidad y correcto funcionamiento del sistema, se diseñaron y ejecutaron casos de prueba, los que arrojaron resultados satisfactorios, demostrando el cumplimiento de los requisitos funcionales establecidos en la fase inicial del proceso de desarrollo del componente.
- El componente implementado para el para el marco de trabajo Bosón, como resultado de la investigación realizada, será de gran utilidad para los desarrolladores de la Universidad de las Ciencias Informáticas permitiéndole disminuir el tiempo en el proceso de creación de reportes.

Recomendaciones

Una vez concluida la investigación se proponen como recomendaciones para futuras versiones lo siguiente:

- Separar la consulta y la plantilla en entidades diferentes para que puedan ser reutilizables en otro reporte.
- Incluir una funcionalidad que permita agregarle gráficos a los reportes.
- Permitir la inclusión de parámetros dinámicos en las consultas y plantillas.

Bibliografías y Referencias

Álvarez, M. A. Características y ventajas de las CSS. [En línea: el 9 de junio del 2008] Características y ventajas de las CSS, 2008. [Consultado el: 15 de diciembre del 2015].Disponible en: http://www.desarrolloweb.com/articulos/introduccion-css3.html

Álvarez, M. A. Qué es MVC. [En línea: el 2 de enero del 2014]. [Consultado el: 15 de diciembre del 2015].Disponible en: http://www.desarrolloweb.com/articulos/que-es-mvc.html

Arizaca, R. E. Artefacto: Diagrama de Componentes. La Paz, Bolivia, 2009.

Buenas Tareas, Diseño Pre-Experimental [Consultado el: 25 de mayo de 2016]. [En línea 24 de mayo de 2011].Disponible:http://www.buenastareas.com/ensayos/Dise%C3%B1o-PreExperimentales/2225599.html.

Claudia García Suárez del Villar, Julio César Brito Rodríguez, Clara Elena Brizuela Herramienta para importar reportes [El línea: 2013]. Habana. ISSN: 2306-2495.

Chidamber, Shyam R. y Kemerer, Chris F. 1994.A Metrics Suite for Object Oriented Design. s.l.: IEEE Transactions on Software Engineering, 1994. vol. 20, no. 6

Crystal Solutions. *SAP Crystal Solutions: Essential BI for small business.* [En línea]. SAP Crystal Solutions, 2013. Disponible en: http://www.crystalreports.com

CIBERAULA. Una introducción a Apache. 4 de febrero de 2013 2010, nº Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro/.

Carolina, Slideshare, Diseño Experimental Puro [Consultado el: 25 de mayo de 2016]. [En línea 31 de agosto de 2010]. Disponible: http://es.slideshare.net/CARRROM/sintesis-metodos.

Eguiluz, J. (2013). Desarrollo web ágil con Symfony2. España: Gestor de publicaciones EasyBook, 2013.

Explorable, Investigación Experimental [Consultado el: 25 de mayo de 2016]. [En línea 09 de octubre de 2013]. Disponible: https://explorable.com/es/investigacion-experimental.

Figueredo. Herramienta para importar reportes. [En línea 15 de junio de 2013].

Fast Reports. Herramienta de Generación de Informes. [Consultado el: 15 de marzo de 2016]. [En línea 21 de febrero 2013]. Disponible: https://www.fast-report.com/es/product/fast-report-net/

Fernández, R.J.,. 2010.Modelo de Datos. 2011.

Giraldo G., Gloria L., Acevedo O., Juan F. y Moreno N., David A. Revista Avances en Sistemas e Informática. [En línea] 03 de 12 de 2011. [Citado el: 20 de 03 de 2016.] Disponible en: http://www.redalyc.org/articulo.oa?id=133122679013. ISSN: 1657-7663.

Grosso, Andrés. 2011. Patrones-Grasp. [En línea] 21 de 3 de 2011. [Citado el: 15 de marzo de 2016.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

HERNANDEZ, I. A. S. Generador Automático de Reportes Dinámicos. 20 de enero de 2013 2003, Disponible en: http://www.cs.cinvestav.mx/tesisgraduados/2003/resumenIlianaAma.htm.

IEEE. 1998. Guide to Software Requirements Specifications. ANSI/IEEE Standard 830-1998. 1998.

JASPERSOFT CORPORATION. JasperReports Server. [En línea Jaspersoft Community, 2013.] Disponible en: http://community.jaspersoft.com/wiki/what-jasperreports-server.

JetBrains. [En línea 2016] [Citado el: 3 de Febrero de 2016.] https://www.jetbrains.com/phpstorm/features/.

Laborde, M. Generador Dinámico de Reportes V 2.0: Análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes. 2011, nº.

Leffingwell, Dean y Widrig, Don. "Using Software Engineering Techniques for Business Modeling. The Unified Modeling Language". Managing software requirements: a use case approach. United States of América: Pearson Education, Inc., 2003. 0-321-1224. 2003.

Lucia Negreira, Gloria Vich, Jessica Gamonal, Tatiana Rojas. Investigación Experimental [Consultado el: 25 de mayo de 2016]. [En línea 23 de noviembre de 2010]. Disponible: http://es.slideshare.net/gloriavich91/investigacin-experimental.

Lorenz, M. y Kidd, J. 1994.Object Oriented Metrics. Englewood, New Jersey: Prentice Hall, 1994

Mario García Salinero. Grid de datos desarrollado con AngularJS utilizando el patrón Modelo-Vista-Controlador. Septiembre 2014.

Martínez, Ivette carolina. 2005.Ingeniería del Software I. Universidad Simón Bolivar , Venezuela : s.n., 2005.

Damián Pérez Valdés MAESTROSDELWEB. ¿Qué es JavaScript? 4 de febrero de 2013 nº Disponible en: http://www.maestrosdelweb.com/que-es-javascript/.

Pearson Prentice Hall .Ingeniería de la web y patrones de diseño. [En línea 2005].

Pytel, Pablo, Uhalde, C, Ramón, Hugo Dionisio, Castello, H, Tomasello, M, Pollo Cattaneo, María Florencia, Britos, Paola Verónica, García Martínez, Ramón SEDICI. [En línea mayo del 2011]. Disponible: http://sedici.unlp.edu.ar/handle/10915/20070 [Consultado el: 28 de abril de 2016].

Nacho Pacheco. Doctrine 2 ORM Documentation. [En línea 03 de noviembre del 2011].

Pressman Roger S. Ingeniería del Software. Un enfoque práctico. Sexta Edición [Libro]. - [s.l.] : Mc Graw Hill, 2003. - 970-10-5473-3.

PHPnet. [En línea] [Citado el: 3 de Febrero de 2016.] Disponible: http://php.net/manual/en/intro-whatis.php

Rafael Felipe Bomate Gavio, Yenia Román Bu, Cinthya Rodríguez Hernández, Carlos Manuel Delgado Rivero, Manuel Cortés Cortés. GeReport: Sistema de Gestión de Reportes Dinámicos. [En línea] 1 de julio del 2014.

Raycraft. Herramienta de creación de informes web PHP dinámicos. [Consultado el: 15 de marzo de 2016]. [En línea 1 de octubre de 2015]. Disponible: http://www.fiuxy.net/programas-gratis/3231946-php-report-maker-v6-0-1-herramienta-de-creacion-de-informes-web-php-dinamicos.html.

Rodríguez Sánchez, T. 2014. Metodología de desarrollo para la Actividad productiva de la UCI. [Aut. Libro] T. RODRÍGUEZ SÁNCHEZ. Cuba: s.n., 2014.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2007. El Lenguaje Unificado de Modelado Manual de Referencia. S.l.: Addison-Wesley Iberoamérica, 2007. 9788478290871. 2007.

Sommerville, Ian. 2005. Ingeniería del software. Séptima Edición. Madrid. España: Pearson Educación. S. A., 2005. 84-7829-074-5. [En línea] 2005. [Citado el: 07 de noviembre de 2015.]

Torres, Abraham Calas. 2014. Excriba. *Ficha técnica del proyecto Bosón.* [En línea] 2014. [Citado el: 3 de Noviembre de 2015.] Disponible: https://excriba.prod.uci.cu/proxy/alfresco//api/node/content/workspace/SpacesStore/e5032d4e-1472-46bf-b554-e37016ed0009/CEIGE_Boson_Ficha%20tecnica%20del%20proyecto.odt