Universidad de las Ciencias Informáticas

Facultad 2

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

Título: Componente para la generación de reportes dinámicos en el sistema ABCD 3.0

Autor:

Osmani Alvarez Pérez

Tutores:

MSc. Eliana Bárbara Ril Valentin

Ing. Anniel Sánchez Verdecia

Ing. Luis Felipe Pérez De la Vega

La Habana, 21 de junio 2018

“Año 60 de la Revolución”

"Aprendí que el coraje no era la ausencia de miedo, sino el triunfo sobre él. El valiente no es el que siente miedo, sino el que vence ese temor"

Nelson Mandela

Agradecimientos

Hoy 21 de junio del 2018, después de 5 años de esfuerzo y sacrificio se cumplen unas de mis metas, hacerme Ingeniero en Ciencias Informáticas, demostrando que nada imposible, solo hay que proponérselo y verás que todo en la vida se puede. Durante esta travesía de altos estudios no estuve solo, estuve acompañado de todas esas personas que en todo este tiempo formaron parte de mi vida, por eso mis más sinceros agradecimientos.

Primeramente, quisiera agradecer a Dios por haberme dado la oportunidad de cumplir una de mis grandes metas.

A mis tutores Anniel, Baby y Luis Felipe por haberme acompañado durante esta travesía, por apoyarme y ayudarme a cumplir uno de mis mayores logros. A ti Anniel por confiar en mí, que, a pesar, de no tener las cosas bien claras de lo que teníamos que hacer desde un principio estuviste ahí conmigo, apoyándome en todo, aunque no supieras, muchas gracias. A ti Baby gracias por dedicarme tiempo, aunque el exceso de trabajo no te lo permitiera, por llenarme el Word de comentarios, por poder contar contigo cuando tenía dudas, muchas gracias.

Al tribunal por todos esos consejos que me dieron durante el desarrollo de la tesis, para que el documento y la defensa estuviera de acorde a un trabajo de diploma, gracias por todo.

A mi oponente Gleidis por revisarme el documento a la perfección y poder contar con ella en cualquier momento que me hiciese falta, gracias.

A todos los profesores que, durante estos 5 años de estudios, contribuyeron a mi formación como profesional.

A Yosamy por dedicarme tiempo, por convertirse en parte de mi tesis, por ser otro tutor más en la elaboración de ella, muchas gracias.

A mis compañeros de aula, a los que se quedaron en el camino y los que hoy como yo cumplen sus sueños, gracias a todos por acompañarme en esta aventura, que es la universidad: David, Adrian, Antonio, Naymaris, Jessie, Victor, Yailen, Dany.

A todos los que han compartido conmigo momentos inolvidables tanto de estudio como de fiestas, gracias por hacer más agradable estos 5 años en la universidad.

A Wilmer, mi compañero de cuarto, disculpa si a veces fui muy exigente con la limpieza, pero es que más regado que tú hay que mandarlo hacer, de verdad gracias por compartir esos 2 años con tu compañía.

A Maikel, gracias por todos esos repasos de programación, por molestarte cada vez que tenía problemas con la máquina. Espero que, aunque tengas mente de pececito, nunca te olvides que tuviste un compañero de apartamento llamado Osmani.

A Suri, por compartir tu amistad, por ser así como eres, cada cual tiene su forma de vivir y así es la tuya, por favor no cambies. Gracias por buscarme siempre la lengua y criticar mis chistes universitarios, aunque a veces no lo demuestres sé que tienes un corazón que no te cabe en el pecho, gracias por todo.

A Miguel, por brindarme tu sincera amistad, por siempre prestarme la llave del apartamento cuando la mía se quedaba adentro, por ser siempre mi compañero de trabajos en la PID, por nunca quedarte callado a las cosas que no están bien, gracias por todo.

A Mayde, por ser esa persona maravillosa que eres, por pasar momentos felices y compartir juntos. A pesar de que nos conocimos un poco tarde creo que no fue en vano, tu amistad fue una de las cosas más bonitas que me pasó en la universidad. Voy a extrañar esos momentos en que nos sentábamos y nos reíamos de los disparates que decíamos, de esas llamadas cada vez que estábamos aburridos o íbamos a comer algo, pero como te dije una vez, aunque la distancia nos separe lo más importante se queda para siempre, el recuerdo y para mí siempre serás mi bichito.

Dejo de último y no porque sean menos significantes para mí, sino porque gracias a ellos después de 5 años de preocupación, sacrificio y dedicación hoy son los protagonistas de mi sueño hecho realidad: mi familia.

A mi padre, por ser ejemplo a seguir, por apoyarme en todo lo que podía y cuando podía, por su preocupación, por su cariño, por ayudarme a cumplir este sueño, lo que para él era poco, para mí era mucho, gracias papá.

A Teté como así le llamamos, gracias por acogerme como un hijo más, por preocuparte de mis problemas, por apoyarme en todo lo que pudieras. Recuerdo que cuando viniste a formar parte de mi vida era mucho

más chiquito que tú y hoy ya es lo contrario, el tiempo pasa. A pesar de que nunca te lo haya dicho eres como una madre para mí, te quiero.

A mi madrina Dae, gracias por acogerme como tu ahijado, por tu dulzura. Aunque la distancia nos haya separado un poco, nunca dejaste de serlo para mí, siempre te tuve presente, siempre seré tu Chibirico, gracias por todo.

A mis hermanos Isy y Ariam, que, aunque no lo sean de sangre, son como hermanos para mí. Gracias por sus apoyos durante estos años de universidad, por nunca decir no puedo, por siempre estar ahí para lo que necesitaba, muchas gracias.

A mi hermano, por ser como un padre para mí, por cuidarme, apoyarme y siempre estar pendiente de mí, por ser mi sangre y por todo. Te quiero mi hermano.

Para terminar, agradezco a esas dos personas que son lo más importante en mi vida:

A ti mamá, por ser la mujer más maravillosa que he conocido, por tu apoyo incondicional, por aconsejarme, por quererme y mimarme, por darme todos los gustos por solo ver en mí una simple sonrisa, por ser un ejemplo de mujer luchadora e inteligente. Eres tú la que me daba ánimos cuando a veces me sentía un poco inseguro, logrando en mí esa fuerza de que, si se puede y bueno, hoy vemos ese fruto, por eso y por más este regalo es para ti. Te quiero.

A mi abuela, por ser la mejor abuela que podía tener, por preocuparse y estar siempre pendiente de mí, por ser esa persona que siempre me decía: mira a ver que no se te quede nada (la toalla), no te mojes con agua de lluvia, no te bañes acabado de comer, y son estas cosas las que te hacen especial. Por eso le doy gracias a Dios por tenerte a mi lado y contar con una abuela como tú. Un día me dijiste ojalá dure unos cuantos años más y poder verte hacerte ingeniero, pues así fue, hoy vez a tu nieto graduarse y cumplir su sueño después de 5 años de sacrificio, gracias por contar siempre contigo y cuidarme como lo has hecho hasta ahora, que Dios te proteja y te de fuerza para tenerte a mi lado unos años más. Te quiero.

A todos muchísimas gracias.

Dedicatoria

Quiero dedicar este trabajo de diploma a mi familia por estar junto a mí en estos 5 años de estudios:

A mis padres que son lo más apreciado que tengo en la vida, sin ellos yo no estuviera aquí en este mundo. Por confiar siempre en mí y estar seguros que no los defraudaría. Por haberme convertido en la persona que soy hoy, gracias a su excelente educación. Por todo el amor, cariño, dedicación y esfuerzo que han mostrado en todos estos años. Los quiero.

A mi hermano, por ser más que hermano, mi amigo. Por estar siempre a mi lado y cuidarme como su hermano menor.

A mi abuela por estar siempre ahí cuando la necesito y por siempre preocuparse de mí, gracias por existir y a la vida por tener una abuela así.

A Teté, por acogerme como un hijo. Por tratarme igual que a mis hermanos, por estar siempre pendiente de mis problemas y brindarme su apoyo incondicional, gracias.

A mis hermanos Isy y Ariam, por todo el apoyo que me han brindado durante estos 5 años de estudios, muchas gracias.

A mi padrastro por preocuparse por mí y formar parte de mi familia.

A todos muchas gracias.

Declaración de autoría

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____del año 2018.

Osmani Alvarez Pérez

_____

Firma autor

Ing. Anniel Sánchez Verdecia

_____

Firma tutor

Ing. Luis Felipe Pérez De la Vega

_____

Firma tutor

Msc. Eliana Bárbara Ril Valentin

_____

Firma tutora

Datos de contacto

Tutora:

MSc. Eliana Bárbara Ril Valentin

Graduada en: Ingeniería en Ciencias Informáticas, graduada del 2008.

Maestría: Máster en Gestión de proyectos informáticos en el 2012.

e-mail: ebril@uci.cu

Tutor:

Ing. Anniel Sánchez Verdecia

Graduado en: Ingeniería en Ciencias Informáticas, graduado del 2016.

e-mail: asverdecia@uci.cu

Tutor:

Ing. Luis Felipe Pérez De la Vega

Graduado en: Ingeniería en Ciencias Informáticas, graduado del 2016.

e-mail: lfperez@uci.cu

Resumen

Los generadores de reportes son herramientas complementarias de los sistemas de información, que se emplean para generar y obtener de forma organizada la información del reporte. El proyecto Automatización de Bibliotecas y Centros de Documentación (ABCD), actualmente cuenta con el sistema ABCD 3.0, el cual permite generar reportes de forma estática, lo que provoca que el usuario no tenga más conocimiento sobre los datos almacenados en la base de datos, impidiendo la toma de decisiones.

Para agilizar la obtención de dicha información en el módulo Estadística surge la necesidad de desarrollar un componente para la generación de reportes dinámicos. Para el desarrollo de la solución propuesta se utilizaron las herramientas y tecnologías previamente definidas por el proyecto ABCD: UML 2.0 como lenguaje de modelado, Visual Paradigm 8.0 como herramienta CASE, Java 7.0 como lenguaje de programación, Eclipse Kepler como entorno de desarrollo, AUP-UCI en su escenario 2 como metodología de desarrollo, además de PostgreSQL y PgAdmin III como gestor y administrador de base de datos respectivamente. Se implementaron las funcionalidades propuestas y se validaron mediante pruebas unitarias, funcionales y de aceptación. Las pruebas arrojaron resultados satisfactorios, lo cual demuestra la calidad de la solución propuesta. Una vez concluido el proceso de desarrollo se obtuvo como resultado un componente que permite la generación de reportes dinámicos sobre los datos almacenados en la base de datos Postgres del sistema ABCD v3.0 y que es capaz de mostrar todos los resultados obtenidos mediante las consultas SQL.

Palabras clave: biblioteca, componente, consultas SQL, reportes dinámicos

Abstract

The report generators are complementary tools of the information systems, which are used to generate and obtain the information of the report in an organized manner. The Automation of Libraries and Documentation Centers (ABCD) project, currently has the ABCD 3.0 system, which allows generating reports in a static manner, which causes the user to have no more knowledge about the data stored in the database, preventing decision making.

To speed up the obtaining of this information in the Statistics module, the need arises to develop a component for the generation of dynamic reports. For the development of the proposed solution, the tools and technologies previously defined by the ABCD project were used: UML 2.0 as a modeling language, Visual Paradigm 8.0 as a CASE tool, Java 7.0 as a programming language, Eclipse Kepler as a development environment, AUP- UCI in its scenario 2 as a development methodology, in addition to PostgreSQL and PgAdmin III as manager and database administrator respectively. The proposed functionalities were implemented and validated through unit, functional and acceptance tests. The tests yielded satisfactory results, which demonstrates the quality of the proposed solution. Once the development process was concluded, a component was obtained that allows the generation of dynamic reports on the data stored in the Postgres database of the ABCD v3.0 system and that is able to show all the results obtained through SQL queries.

Keywords: component, dynamic report , library, queries SQL

Índice

Índice de figuras

Índice de tablas

Introducción

Los sistemas de información son un conjunto de componentes que interaccionan entre sí para alcanzar un fin determinado, los cuales satisfacen las necesidades de información de dicha organización (Gardey, 2008). Estos componentes pueden ser personas, datos, recursos materiales en general, los cuales procesan la información y la distribuyen de manera adecuada, buscando satisfacer las necesidades de la organización. Los sistemas de información han sido de gran importancia para el apoyo a la toma de decisiones y controlar todo lo que en ella ocurre, estos se clasifican en dos tipos: los formales y los informales; los primeros utilizan como medio para llevarse a cabo estructuras sólidas como ordenadores y los segundos son más artesanales, usando medios más antiguos como el papel y el lápiz.

Dentro de las principales funcionalidades de los sistemas de información, se encuentra la de proveer toda la información que se genera durante los distintos procesos de gestión en las empresas, relacionados con el control y supervisión de los mismos. Para lograr un control eficiente en la gestión de estas tareas, se hace necesario buscar mecanismos para la generación de reportes. Los mismos deben ser capaces de consolidar la información adquirida y mostrarla en formatos entendibles por el personal que espera recibir la información.

Mediante los sistemas informáticos dedicados a la generación de reportes dinámicos se pueden mostrar los datos que están contenidos en una base de datos mediante una estructura organizada y sugerente. En adición a esto, constituyen una poderosa herramienta diseñada para cumplir con las cambiantes necesidades de información que demandan las empresas. La generación de reportes está centrada especialmente en los usuarios finales, es decir, en las personas que realizan el análisis de la información a través de los mismos. Esto les permite diseñar sus reportes en poco tiempo obteniendo una mejor visualización de la información gracias a los elementos (textos, imágenes, tablas) que se les pueden incluir a los mismos (Hernández, 2003).

En Cuba durante los últimos años se ha evidenciado el desarrollo de sistemas para la generación de reportes, entre estos se pueden mencionar la implementación del Generador Dinámico de Reportes del Centro de Tecnologías de Gestión de Datos (DATEC) de la Universidad de las Ciencias Informáticas (UCI), es una herramienta multiplataforma desarrollada con tecnologías web. Este sistema permite la creación y edición de informes/reportes extrayendo datos de una amplia gama de gestores de bases de datos (Moleiro y Gutierrez, 2015).

Introducción

Para la informatización del país, surge la UCI que cuenta con diferentes centros de desarrollo de software, uno de estos es el Centro de Informatización de la Gestión Documental (CIGED). El mismo tiene como objetivo el desarrollo de sistemas y servicios informáticos integrales de alta calidad y competitividad en la información o mejora de los procesos de gestión documental. Dicho centro cuenta con el proyecto para la Automatización de Bibliotecas y Centros de Documentación (ABCD), el cual se encarga de desarrollar un sistema web que permita informatizar los procesos relacionados con las distintas áreas que puedan existir en una biblioteca.

El sistema ABCD en su versión 3.0 cuenta con varios módulos tales como: Adquisición, Catalogación, Circulación, OPAC y Servicios web, Administración y Estadísticas. Dichos módulos tienen como función principal capturar la información perteneciente a los diferentes servicios bibliotecarios, que adicionalmente puede ser utilizada para la toma de decisiones. Uno de estos servicios es la generación de reportes estadísticos, ya que la gestión bibliotecaria se basa en gran medida en la disponibilidad de los mismos por lo que se deben generar oportunamente para realizar las evaluaciones de la operación de los servicios brindados.

Específicamente el módulo Estadísticas en la versión 3.0, se desarrolla de forma tal que se mantenga la misma flexibilidad y dinámica en cuanto a reportes y estadísticas se refiere, con respecto a la versión anterior. Este módulo permite generar reportes mediante consultas SQL (*Structured Query Language*) a la base de datos Postgres que se utiliza. Estas consultas solo pueden devolver un número (cantidad), que es presentado al usuario en una tabla donde se listan los indicadores estadísticos y el resultado de las consultas. Estos reportes no permiten devolver listas de un objeto determinado, objeto que normalmente se almacena en la base de datos del sistema ABCD. Esto provoca que el usuario no tenga conocimiento sobre todos los datos almacenados en la base de datos, imposibilitando conocer por ejemplo un listado de nombres de libros perdidos, rotos o con un estado determinado (prestado, ejemplar único). No permite seleccionar varios datos de un objeto en específico, así como de la tabla permisos, seleccionar los id, los nombres de los permisos y además porque sería bueno mostrarlos en una tabla, lo que no ayuda a la toma de decisiones.

Dada la situación problemática anteriormente descrita, se identifica como **problema a resolver:** ¿Cómo contribuir al proceso de generación de los reportes dinámicos del sistema ABCD v3.0?

Partiendo de este problema se define como **objeto de estudio:** proceso de generación de los reportes dinámicos en los sistemas de gestión informáticos.

Para resolver el problema identificado se ha propuesto como **objetivo general:** desarrollar un componente para la generación de reportes dinámicos en el sistema ABCD v3.0 que agilice la obtención de información. Enmarcado en el **campo de acción:** herramientas informáticas que implementan la generación de reportes dinámicos.

Para el cumplimiento del objetivo general de la investigación se definen las siguientes **tareas de la investigación:**

1. Elaboración de los fundamentos teóricos y metodológicos de la investigación relacionados con la generación de reportes dinámicos y herramientas informáticas que los implementan.
2. Definición de los requisitos asociados al negocio de la propuesta de solución para el sistema ABCD 3.0.
3. Diseño de los requisitos asociados al negocio de la propuesta de solución para el sistema ABCD 3.0.
4. Desarrollo del componente para la generación de reportes dinámicos en el sistema ABCD 3.0.
5. Validación del componente desarrollado a partir de los métodos de prueba definidos en la investigación.

Con el desarrollo del componente de generación dinámico de reportes se espera obtener el siguiente **aporte práctico*:*** componente de software que permita la generación de reportes dinámicos sobre los datos almacenados en la base de datos Postgres del sistema ABCD v3.0 y que sea capaz de mostrar todos los resultados obtenidos mediante las consultas SQL.

Para el desarrollo de la investigación se utilizan los siguientes **métodos teóricos**:

Analítico - sintético: la utilización de este método permitió la comprensión y funcionamiento de gestores de reportes dinámicos a través del análisis de documentos, libros, artículos y otras fuentes bibliográficas de diferentes autores, lo que permitió definir características principales de estos gestores.

Inductivo - deductivo: mediante este método se llegó a conclusiones sobre la generación de reportes dinámicos sobre los datos almacenados en la base de datos Postgresql.

Modelación: la utilización de este método permitió realizar una reproducción simplificada de la realidad, a través de los artefactos presentados, modelados en la herramienta Visual Paradigm y que responden a la metodología de desarrollo de software empleada.

Para el desarrollo de la investigación se utiliza el siguiente **método empírico**:

Entrevista: Este método permitió tener más información sobre la problemática existente y las funcionalidades a desarrollar en el módulo Estadística del sistema ABCD v3.0 con especialistas del proyecto (Ver Anexo 1).

El documento se encuentra estructurado de la siguiente manera:

Capítulo 1. Fundamentación teórica sobre la generación de reportes dinámicos: En este capítulo se realiza una revisión bibliográfica del estado actual de la temática en estudio, con el objetivo de caracterizar y profundizar las herramientas, lenguajes, tecnologías y metodología que se requieren para dar cumplimiento al objetivo de la presente investigación.

Capítulo 2. Descripción del componente propuesto para la generación de reportes dinámicos: En este capítulo se genera un modelo de dominio donde se analizan las entidades y conceptos existentes en el contexto, donde se realiza la propuesta solución. Se explican los requisitos funcionales y no funcionales con los que debe cumplir el componente, así como la arquitectura propuesta para el componente a desarrollar, generando todos los artefactos necesarios (según la metodología utilizada).

Capítulo 3. Implementación y pruebas del componente propuesto: En este capítulo se explica a partir de los resultados del diseño, la implementación del componente. Se definen los estándares de codificación. Se abordan los resultados obtenidos en el desarrollo de la solución y las consideraciones finales mediante la realización de las pruebas pertinentes realizadas al software. Se define además la estrategia a seguir en las pruebas, especificando niveles, métodos y tipos de pruebas. Por último, se muestran los resultados obtenidos a partir del diseño de cada caso de prueba realizado.

Capítulo 1: Fundamentación teórica sobre la generación de reportes dinámicos

En el presente capítulo se define el marco teórico de la investigación. Se realiza un estudio de los conceptos fundamentales asociados a la presente investigación durante el desarrollo de la misma. Se describe la metodología de desarrollo de software y el ambiente de trabajo a emplear definido para los proyectos del centro CIGED, con el objetivo de explotar al máximo sus potencialidades en función de cumplir los objetivos propuestos.

1.1 Conceptos asociados a la investigación

A continuación, se muestran algunos de los conceptos de mayor importancia, relacionados a la investigación realizada:

Componente:

Un componente contiene un conjunto de clases que colaboran entre sí. Cada clase de un componente se ha elaborado completamente para incluir todos los atributos y las operaciones relevantes para su implementación (Pressman, 2005).

Para mostrar la información almacenada a los usuarios en la base de datos, se hace de **Reportes:**

Según (Merino, 2017), son documentos que pretenden transmitir una información contenida en la base de datos, puede ser de forma digital o impresos.

Para poder generar reportes se hace uso de un **Generador de reportes dinámicos:**

Según (Rodríguez, 2011) plantea que es una aplicación web, que permite a sus clientes consultar las bases de datos de sus organizaciones y poder generar reportes con la información que estos manejan, independientemente del Sistema Gestor de Base de Datos que utilicen ya sea MySQL, SQLite o PostgreSQL.

1.2 Estado actual de los reportes

Los reportes han sido el medio principal para obtener la información. Tanto reportes en papel como en el escritorio, permiten al usuario comunicar lo que está ocurriendo en una empresa. Los reportes tienen en

las bases de datos su principal fuente de alimentación y han brindado al usuario final la posibilidad de consultar y publicar lo que las bases de datos poseen. La limitante que siempre ha existido es este sentido es que al generar un reporte implica manejar algunas habilidades técnicas relacionadas con las bases de datos y las herramientas de software.

El usuario técnico realiza los reportes y para ello necesita conectarse a la base de datos, posteriormente, diseñar el formato requerido, y al final, obtener los datos. Tanto la conexión a la base de datos como a la construcción de formato del reporte son tareas de sistemas.

1.3 Herramientas de consultas y reportes

Las herramientas de consulta y reportes son una categoría de herramientas de inteligencia de negocios. Con estas herramientas de reportes orientadas al usuario final se pretende mejorar la obtención de información.

La herramienta de reportes, permite utilizar lenguaje SQL, uniones de tablas y nombres crípticos, al organizar los datos de la terminología de negocios (Nader, 2003). El resultado es que el usuario final o intermediario, tiene una vista mucho más parecida a su concepción del negocio, o al menos lo suficiente como para poder generar sus propios reportes y publicación de los mismos.

1.3.1 Características de una herramienta de reportes

Una herramienta de reportes orientada al usuario final debe también poseer algunas utilidades adicionales que faciliten la generación y publicación de reportes (Nader, 2003):

- **Intuitivo:** Como cualquier herramienta de la inteligencia de negocios, la característica común es su facilidad de uso. Con apoyo en interfaces gráficas y visuales, un usuario con una formación estándar podrá hacer uso de una herramienta para generar reportes.
- **Seguridad:** Deben brindar seguridad para el acceso a los reportes, tanto a nivel usuario como por grupos e, incluso, en el grado de profundidad de cada usuario a la información. Esto con la idea de que la información privada no sea accesible por cualquier persona.
- **Programación automática:** Generación de instrucciones para que los reportes se ejecuten automáticamente e incluso se distribuyan mediante correo electrónico.
- **Reportes dinámicos:** Permitir el ingreso de parámetros de valor que hagan un reporte flexible y dinámico en el momento de su ejecución.

1.4 Análisis de soluciones existentes que implementan generadores de reportes

En la actualidad existe en el mundo una gran cantidad de sistemas informáticos empleados en la generación de reportes. Estas son herramientas complementarias de los sistemas de información que utilizan una especie de lenguaje transparente para el usuario por medio del cual este realiza consultas a la base de datos y obtiene información de ella en forma de reporte. A continuación, se detallan algunas características de los sistemas estudiados:

Generador Dinámico de Reportes 1.8 (GDR v1.8)

El GDR 1.8 es una herramienta que brinda la posibilidad de controlar el funcionamiento periódico de una o varias entidades, mediante el diseño de cualquier tipo de reportes, incluyendo los tabulares (con gráficos incluidos), tabla pivote y cruzada, desde los gestores de bases de datos: SQL Server, SQLite, Oracle 10g, MySQL y PostgreSQL. Proporciona a los usuarios la ventaja de agilizar el proceso de la toma de decisiones. Esta aplicación se desarrolla sobre el marco de trabajo Symfony 1.1.7, además de ser multiplataforma. Implementada en el lenguaje de desarrollo PHP. La versión 1.8 del GDR cubre el ciclo básico de la generación de reportes y soluciona el problema de obtener los diferentes informes en los sistemas de gestión de la información que se desarrollan en cualquier entorno empresarial, incluyendo la UCI. Dentro de los tipos de gráficas que maneja se encuentran: de barra, de pastel, de línea y de curva (Bouly, 2013).

Sistema de Evaluación, Control y Reportes para el Laboratorio de EMAPA-I, Ciudad de Ibarra

El Sistema de Evaluación, Control y Reportes para el Laboratorio de EMAPA-I, es una solución a todos los requerimientos y procesos de comunicación, transferencia e integración de la información. Tiene como objetivos básicos, gestionar los volúmenes de información, con rapidez, exactitud y a la vez la generación de reportes, para facilitar la toma de decisiones y mejorar el control del laboratorio, en los ámbitos de análisis químicos, análisis microbiológicos, análisis físicos con sus respectivos catálogos de parámetros y sobre todo la administración del laboratorio. Los reportes del sistema se han estandarizado según los requerimientos de las áreas, para lo cual se utilizó la metodología RUP. Para la implementación de este sistema se utiliza como lenguaje de programación Java Serves Pages(JSP) con IDE NetBeans y de Base de Datos PostgreSQL (Quelal, 2011).

GeReport

GeReport es una herramienta destinada al diseño, generación y configuración de los reportes relacionados con los datos históricos almacenados en una fuente de datos. Además, luego de contar con toda la información del reporte es posible exportarlo 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 sistema se creó sobre un entorno web y se desarrolló siguiendo lo establecido por el Proceso Unificado de Desarrollo (RUP), utilizando UML como lenguaje de modelado. Para la gestión de la base de datos se seleccionó PostgreSQL. En la implementación se utilizaron los lenguajes de programación PHP, con CodeIgniter 2.0 como marco de trabajo del lado del servidor y JavaScript con Dojo Toolkit 1.8 para el trabajo del lado del cliente. Se utiliza en la Universidad de Cienfuegos por analistas y programadores del Grupo de Estudios y Desarrollo de Ingeniería y Sistemas, perteneciente a la Facultad de Ingeniería(Gavio et al., 2014) .

Koha

Es un Sistema Integrado de Gestión Bibliotecaria(SIGB) que data de 1999 y está apoyado por una comunidad de desarrollo activa a nivel latinoamericano y mundial. El sistema es completamente multiplataforma para el usuario de Koha (tanto el usuario de la biblioteca como el bibliotecario). Cuenta al igual que ABCD con un módulo OPAC que para acceder a él solo necesita el usuario un navegador web. Por otra parte, desde el punto de vista del servidor, si bien utiliza una arquitectura de tipo LAMP (sistema operativo Linux, servidor Apache, sistema gestor de bases de datos MySQL y lenguajes de programación PHP, Perl o Python). Este sistema se puede utilizar no sólo en equipos con Linux, sino que también hay versiones para otros sistemas operativos, como Windows o Solaris, e incluso puede utilizarse a partir de máquinas virtuales. Es un sistema que se basa en una arquitectura cliente-servidor, sirviendo cualquier navegador web como aplicación cliente, pues las interfaces de este programa son de tipo web. Interfaces en plural, pues se habla de dos distintas: por una parte, la destinada al usuario y que comprende el OPAC, y la destinada a los bibliotecarios, dentro de la cual encontramos los distintos módulos del programa (Prieto, 2012).

En Koha el módulo encargado de realizar las estadísticas es el módulo Informes. Existen dos posibilidades de obtener informes: por un lado se encuentran los informes predefinidos que facilitan informes genéricos de uso (Adquisiciones, Usuarios, Catálogo, Préstamos), informes de inactividad (usuarios que no realizan préstamos, libros no prestados), *Top list* (usuarios que más préstamos realizan,

los libros más prestados) u otros informes (libros perdidos, catálogo por tipo de documento, préstamos por tipo de usuario). Por otro lado, se facilita la confección de informes más personalizados mediante un asistente para informes. El tipo de reporte que usa Koha es tabular. Permite seleccionar varios campos o todos a la vez. Da la opción de realizar operaciones matemáticas con los elementos de las filas y las columnas. Permite además escoger el orden en que se desean imprimir los datos. Además, el usuario puede escribir sus propias consultas SQL (Bidopia y Argüelles, 2016).

ABCD v1.2

ABCD v1.2 es un SIGB de código abierto que realiza las funciones principales de una biblioteca, es decir, adquisiciones, catalogación, gestión de préstamos, control de publicaciones periódicas, estadísticas y servicios de información. Es una aplicación web, multilingüe, que soporta búsquedas en servidores Z39.50. Usa para el almacenamiento de datos la familia extendida del software ISIS. Utiliza como gestor de bases de datos C-ISIS.

Cuenta con un módulo Estadística ofrece varias funcionalidades en su pantalla principal que permiten la realización de las estadísticas, las cuales serán mostradas mediante reportes al usuario. La primera de estas funciones es la creación de una nueva tabla, para ello debe definirse una tabla identificando qué valores se utilizan en la dirección horizontal (filas) y cuáles en la dirección vertical (columnas). La segunda funcionalidad permite la utilización de una tabla existente. Estas tablas se listan en la lista de opciones por sus criterios filas / columna. Permite la generación de la salida la cual implica dos fases: la creación de la tabla con los valores y la creación de cuadros gráficos de salida o la exportación de la tabla a otros formatos externos.

Tabla 1:Comparación entre los sistemas existentes (Fuente: Elaboración propia)

Sistemas	Base de Datos	Lenguaje de Programación	Sistema Operativo	Entorno de Desarrollo
Generador Dinámico de Reportes 1.8	MySQL PostgreSQL	PHP	Software libre	Multiplataforma

Sistema de Evaluación, Control y Reportes para el Laboratorio de EMAPA-I	PostgreSQL	Java	Open Source	NetBeans
GeReport	PostgreSQL	PHP JavaScript XHTML	Software libre	Dojo CodeIgniter
Koha	MySQL	PHP Python	Linux Windows	Multiplataforma
ABCD v1.2	C-ISIS	Java	Linux	Eclipse
ABCD v3.0	PostgreSQL	Java	Linux	Eclipse

Después de un análisis a los sistemas existentes, se llega a la conclusión que estos no cumplen con todas las características que debe tener el sistema ABCD v3.0 por las razones siguientes: el Generador Dinámico de Reportes 1.8 es una aplicación que no utiliza ningunas de las características con las que debe cumplir el sistema ABCD 3.0. Sin embargo, el Sistema de Evaluación, Control y Reportes para el Laboratorio de EMAPA-I utiliza como base de datos PostgreSQL y como lenguaje de programación Java, pero con diferente entorno de desarrollo. GeReport utiliza la misma base de datos, pero no el mismo lenguaje de programación, ni el mismo entorno de desarrollo. Koha utiliza diferente base de datos y lenguaje de programación. A pesar de que el sistema ABCD v1.2 presenta una mayor semejanza con la versión actual, este sistema utiliza como gestor de base de datos C-ISIS, mientras que la actual utiliza como gestor de base de datos Postgresql.

1.5 Sistema Gestor de Base de Datos (SGBD)

Un SGBD es un programa de ordenador que facilita una serie de herramientas para manejar bases de datos y obtener resultados (información) de ellas. Además de almacenar la información, se le pueden hacer consultas sobre esos datos, obtener listados impresos, generar pequeños programas de mantenimiento de la base de datos, o ser utilizado como servidor de datos para programas más complejos realizados en cualquier lenguaje de programación. Permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarios para el almacenamiento y búsqueda de la información del modo más eficiente posible (Iruela, 2016).

Utilidades de un SGBD (Sánchez, 2017):

- Herramientas para administrar y crear la estructura física de la base de datos.
- Herramientas para la manipulación de los datos.
- Herramientas para la gestión de la comunicación de la base de datos.
- Herramientas de instalación y configuración de la base de datos.
- Herramientas para la exportación e importación de datos.

1.5.1 PostgreSQL

PostgreSQL es un SGBD distribuido bajo la licencia BSD (*Berkeley Software Distribution* o Distribución de software Berkeley) y su código fuente se encuentra disponible libremente. Dentro de sus características se encuentran las siguientes (Agüero y Leyva, 2014):

- PostgreSQL soporta consultas SQL declarativas, control de concurrencia multiversión, soporte multiusuario, transacciones, optimización de consultas, herencia y arreglos.
- La integridad referencial es utilizada para garantizar la coherencia de datos entre relaciones aparejadas.
- PostgreSQL soporta lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL.

PostgreSQL se ejecuta en casi todos los principales sistemas operativos y su documentación está bien organizada, pública y libre. Proporciona soporte a las características de las bases de datos profesionales (Ejemplo: disparadores, procedimientos almacenados, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas materializadas).

1.6 Herramientas y tecnologías a utilizar

En el presente epígrafe se describirá el lenguaje de modelado, de programación, entorno de desarrollo, herramienta de modelado y administrador de base de datos que son los definidos para el proyecto ABCD.

1.6.1 Lenguaje de modelado

El modelado de sistemas de software es una técnica que ayuda al ingeniero de software a visualizar el sistema que desea construir. De ahí que los modelos pueden utilizarse para la comunicación con el cliente, grupo de desarrollo y otros factores que intervienen en el proceso de desarrollo (Bouly, 2013).

Para la presente investigación se asume el lenguaje de modelado utilizado en el proyecto ABCD: Lenguaje Unificado de Modelado (UML)**.**

El Lenguaje Unificado de Modelado, por sus siglas en inglés *Unified Modeling Language*, permite visualizar, especificar, construir y documentar un software. Se aplica en el desarrollo de software para dar soporte a una metodología de desarrollo del mismo, sin especificar qué metodología usar. Como lenguaje, es usado para la comunicación y un mejor entendimiento entre el diseñador y el programador (Larman, 2000).

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. Un modelo UML indica qué es lo que supuestamente hará el sistema, pero no cómo lo hará (Piñeiro, 2014).

1.6.2 Lenguaje de programación

Un lenguaje de programación es básicamente un sistema estructurado de comunicación, similar al humano, el cual nos permite comunicarnos por medio de signos, ya sean palabras, sonidos o gestos. Refiriéndonos a los aparatos, este sistema está organizado para que se entiendan entre sí y a su vez interprete las instrucciones que debe ejecutar (Morales, 2014) .

Para la presente investigación se asume el lenguaje de programación utilizado en el proyecto ABCD:**JAVA 7.0.**

La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera es interpretado por una máquina virtual. Java es un lenguaje orientado a objetos de propósito general, se puede utilizar para construir cualquier tipo de proyecto. Presenta varias semejanzas con lenguajes como C, C++ y C#. Este lenguaje está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red. Está creado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows, sobre arquitecturas distintas y con sistemas operativos diversos.

Características de Java:

- **Simple**: Basado en el lenguaje C++, pero donde se eliminan muchas de las características orientadas a objeto que se utilizan esporádicamente y que creaban frecuentes problemas a los programadores.
- **Orientado al objeto**: Java brinda soporte a las técnicas de desarrollo orientado a objeto.
- **Distribuido**: Java se ha diseñado para trabajar en ambiente de redes y contienen una gran biblioteca de clases para la utilización del protocolo TCP/IP, incluyendo HTTP y FTP.
- **Dinámico**: Exige que se compile de nuevo la aplicación al cambiar una clase madre Java. Utiliza un sistema de interfaces que permite aligerar esta dependencia.

Java es el lenguaje de programación del lado del servidor que se va a utilizar para el desarrollo del componente estadístico y por consecuente del módulo Estadísticas. Su uso posibilitó la programación de las clases asociadas a la capa lógica de negocio y almacenamiento.

1.6.3 Entorno de desarrollo

Un Entorno de Desarrollo Integrado (IDE) es una aplicación informática compuesta por un conjunto de herramientas de programación, encargadas de facilitar el trabajo de los desarrolladores de software a la hora de llevar a cabo una aplicación. Permite dedicarse a un único lenguaje de programación o hacer uso de varios de ellos. Los IDE´s proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Basic, Object Pascal (Bouly, 2013).

Un IDE presenta las siguientes características(Quintuña, 2010) :

- Multiplataforma.
- Soporte para diversos lenguajes de programación.
- Integración con Sistemas de Control de Versiones.
- Extensiones y Componentes para el IDE.
- Importar y Exportar proyectos
- Manual de Usuarios y Ayuda.

Para la presente investigación se asume el IDE de desarrollo utilizado en el proyecto ABCD: **Eclipse.**

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar un proyecto. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados

como el IDE de Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (Quintuña, 2010). Es un proyecto de código abierto y gratuito, con una gran base de usuarios y documentación disponible. A continuación, se enuncian algunas de sus ventajas:

- Es un IDE con muchas distribuciones y soporta una gran cantidad de lenguajes.
- Es una herramienta de código abierto. Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- Provee la utilidad de comenzar el programa con los plugins especificados, permitiendo acceder a distintas aplicaciones sin necesidad de levantar todas a la vez al momento de ejecutarlo.
- La plataforma Eclipse está construida en base a plugins. Este mecanismo permite desarrollar, integrar y correr nuevo plugins.

1.6.4 Herramienta de modelado

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero (Valderrama, 2016).

Las herramientas CASE poseen grandes ventajas que permiten aumentar la productividad en el desarrollo de un software:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.
- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la base de datos.

Para la presente investigación se asume la herramienta de modelado utilizada en el proyecto ABCD: **Visual Paradigm 8.0.**

Visual Paradigm es una herramienta CASE que utiliza UML como lenguaje de modelado. Soporta el ciclo de vida completo de desarrollo de un software. Permite realizar ingeniería directa o inversa sobre el software y es capaz de, a partir de un modelo relacional en diferentes sistemas gestores de base de datos, desplegar todas las clases asociadas a las tablas (Piñeiro, 2014). Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, generación del código fuente de los programas y la documentación de los mismos.

Esta herramienta genera diseños centrado en casos de uso y enfocados al negocio contribuyendo así a un software de mayor calidad. Usa un lenguaje estándar y común a todo el equipo de desarrollo que facilita la comunicación. Mantiene capacidades de ingeniería directa e inversa y posibilita que el modelo y el código permanezcan sincronizados en todo el ciclo de desarrollo.

Las ventajas que proporciona Visual Paradigm para UML 8.0 son (Reuse, 2013) :

- Dibujo: Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Coherencia entre diagramas: Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Generación de código: Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- Generación de informes: Permite generar diversos informes a partir de la información introducida en la herramienta.

1.6.5 Administrador de base de datos

Un administrador de base de datos dirige o lleva a cabo todas las actividades relacionadas con el mantenimiento de un gestor de base de datos. Las responsabilidades incluyen el diseño, implementación y mantenimiento del sistema de base de datos; el establecimiento de políticas y procedimientos relativos a la gestión y la seguridad.

Para la presente investigación se asume el administrador de base de datos utilizado en el proyecto ABCD: **PgAdmin III.**

PgAdmin es una herramienta de código abierto para la administración de bases de datos. Está diseñada para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas sql hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de Postgresql y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows y Linux.

1.7 Metodología de desarrollo

Las metodologías de desarrollo de software constituyen un conjunto de procedimientos y técnicas para ayudar a los equipos de desarrollo de software con la documentación durante el proceso de concepción de un software. Describen paso a paso todas las actividades a realizar para lograr el producto informático deseado y definen las personas que participan en el desarrollo de las actividades, así como su papel en las mismas (Gimson, 2012).

1.7.1 AUP-UCI

La metodología de desarrollo de software Variación de AUP para la UCI, es una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP (Proceso Ágil Unificado) y está definida por la universidad como el documento rector de la actividad productiva. Se decide utilizar esta metodología, ya que es utilizada por el proyecto ABCD para el desarrollo de los sistemas.

De las 4 fases que propone la metodología AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la universidad, que la metodología AUP-UCI quede formada por tres fases, (Inicio, Ejecución y Cierre). Las características de estas fases son (Romeu, 2016):

- Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planificación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- Cierre: En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Descripción de las disciplinas

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de

negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran cada una de estas disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían Gestión de la configuración (CM), Planeación de proyecto (PP) y Monitoreo y control de proyecto (PMC).

Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución (Sánchez, 2015). A continuación, se muestran las fases en la figura 1:

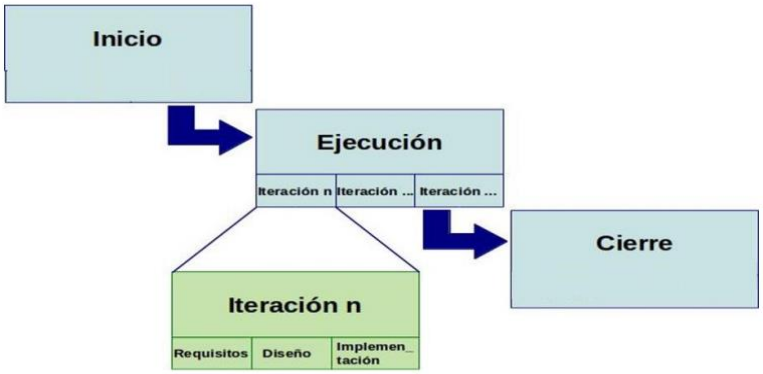


Figura 1: Fases e iteraciones

El escenario de la disciplina Requisitos que se aplica a ABCD es el Escenario No2 para proyectos que modelen el negocio con Modelo Conceptual (MC) solo pueden modelar el sistema con Casos de Usos del Sistema (CUS), el mismo plantea:

Escenario No2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información (Sánchez, 2015).

Figura 2: Escenario No2 de la metodología AUP-UCI

Conclusiones del capítulo

En este capítulo se arribaron a las siguientes conclusiones:

1. El estudio realizado sobre los conceptos fundamentales facilitó sentar las bases teóricas de la investigación.
2. El uso de la metodología de desarrollo de software permitió organizar y guiar por fases de desarrollo la solución propuesta.
3. El estudio de las herramientas y tecnologías a utilizar permitió una mayor comprensión de las mismas para su correcto uso.

Capítulo 2: Descripción del componente propuesto para la generación de reportes dinámicos

A continuación, se muestra el modelado conceptual de la solución propuesta, así como los requisitos funcionales y no funcionales que debe cumplir el componente a desarrollar. A partir de estos se modela el componente quedando representado en un diagrama de casos de usos del sistema y sus descripciones textuales. Además, se modelan los artefactos definidos por la metodología AUP-UCI en su escenario 2 como son los diagramas de clases del diseño.

2.1 Propuesta de solución

Con el objetivo de agilizar y facilitar la generación de reportes estadísticos en el sistema ABCD v3.0, se requiere la implementación de un componente para la generación de los mismos. Dicho componente debe desarrollarse en el módulo de Estadísticas, el cual permite gestionar reportes asociados a la gestión de procesos de la biblioteca, incluyendo la administración de indicadores, variables, tablas y reportes.

El componente proporcionará al módulo de Estadísticas, incorporar nuevas funcionalidades, capaces de cumplir las necesidades actuales del mismo. Permitirá al usuario mediante consultas SQL a la base de datos Postgre mostrar mediante una tabla varios datos de un objeto en específico, así como conocer los nombres de usuarios que han sido sancionados, los libros que se encuentran prestados, rotos o en mal estado, las fechas de devolución de un libro, etc, lo cual facilitará mediante la generación de reportes dinámicos, la toma de decisiones por parte del bibliotecario. A continuación, se muestra en la fig. 3 dicha propuesta de solución:

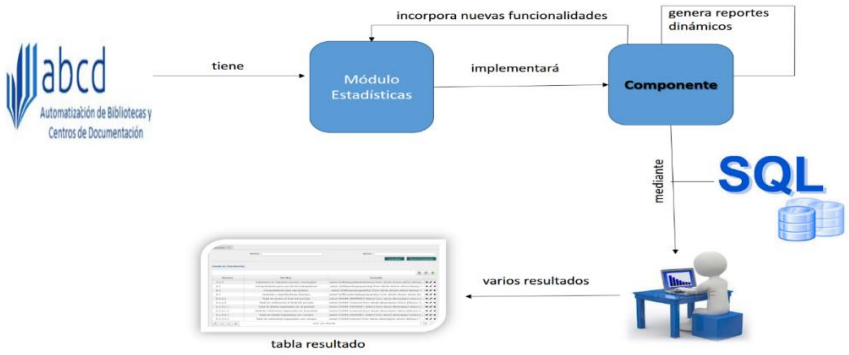


Figura 3: Propuesta de solución (Fuente: Elaboración propia)

2.2 Modelo conceptual

El modelo conceptual (también conocido como modelo de dominio) captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los conceptos que existen o los eventos que suceden en el entorno en el que trabaja el sistema (Quesada, 2014). A continuación, se muestra el modelo de dominio de la presente investigación:

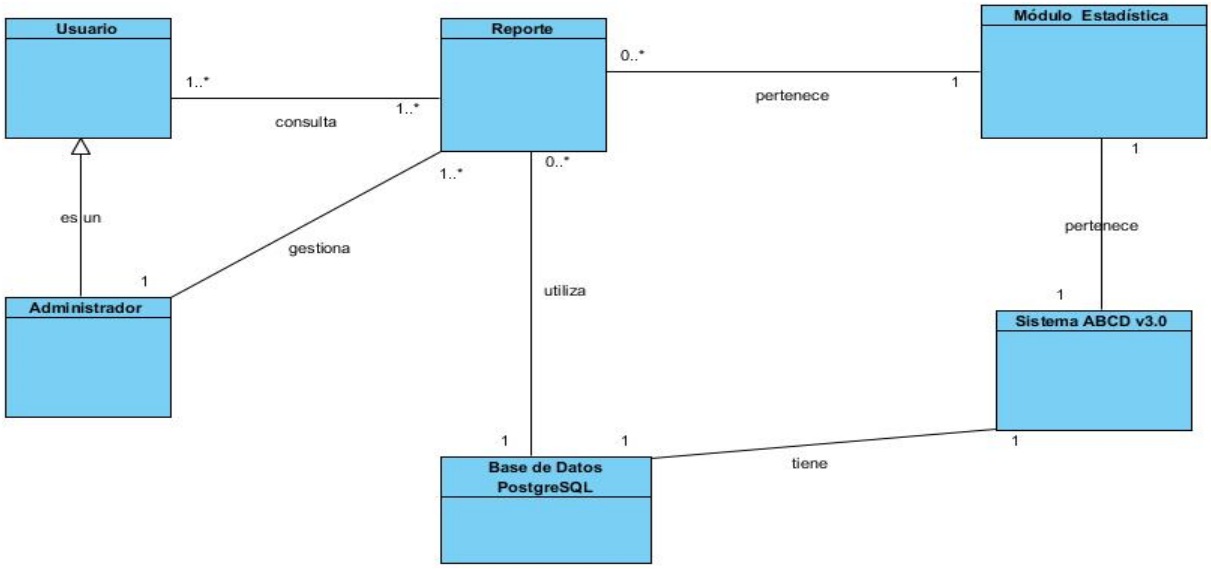


Figura 4: Modelo de dominio (Fuente: Elaboración propia)

2.2.1 Conceptos del modelo de dominio

Administrador: hace referencia a la persona que posee el rol de administrador, este tiene todos los permisos para realizar cualquier cambio dentro del componente para la generación de reportes dinámicos en el sistema ABCD 3.0.

Base de Datos PostgreSQL: la base de datos a la cual le serán aplicadas las consultas SQL, sobre la cual trabajará el usuario.

Módulo Estadística: es el módulo del sistema ABCD v3.0 que va hacer uso del componente para realizar los reportes dinámicos.

Reporte: tipo de contenido que representa un informe que organiza y exhibe la información necesitada por el usuario.

Sistema ABCD v3.0: es el sistema encargado de automatizar los procesos y servicios que brinda una biblioteca a los usuarios.

Usuario: representa todas las personas autenticadas en el sistema y que acceden al mismo con el fin de generar los reportes.

2.3 Requisitos de software

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software. Tanto el desarrollador como el cliente tienen un papel activo en la ingeniería de requisitos, ya que pueden realizar un conjunto de actividades que son denominadas análisis. El análisis de requisitos permite especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software (Pressman, 2011).

2.3.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones con las cuales el sistema debe cumplir e indican su comportamiento, es decir, son las funciones que el sistema debe realizar (Quesada ,2014). A continuación, se listan y describen los requisitos funcionales.

Tabla 2:Descripción de los requisitos funcionales (Fuente: Elaboración propia)

Nº	Requisito	Descripción
1	**Registrar Reporte Dinámico**	El sistema a través de una interfaz debe permitir entrar los datos necesarios para crear un nuevo reporte, luego procesarlos y adicionar el reporte al sistema.
2	**Modificar Reporte Dinámico**	El sistema a través de una interfaz debe permitir luego de listar los reportes, mostrar los campos para la edición del reporte seleccionado.

3	**Eliminar Reporte Dinámico**	El sistema a través de una interfaz debe permitir luego de listar los reportes del sistema, seleccionar un reporte y eliminarlo.
4	**Mostrar Reporte Dinámico**	El sistema a través de una interfaz debe mostrar un listado de los reportes existentes en el mismo y visualizar los mismo.
5	**Consultar Reporte Dinámico**	El sistema a través de una interfaz permite realizar la búsqueda en el sistema y restringe el listado de reportes según los criterios definidos.
6	**Buscar Reporte Dinámico**	Permite limpiar el valor en el campo de búsqueda inicialmente introducido y muestra el listado de reportes con todos los existentes en el sistema.
7	**Exportar a hoja de cálculo el Reporte**	Permite exportar y/o visualizar en formato de hoja de cálculo el listado de reportes visualizado.
8	**Exportar a PDF el Reporte**	Permite exportar y/o visualizar en formato PDF el listado de reportes visualizado.
9	**Generar Reporte Dinámico**	Esta funcionalidad mediante una interfaz permite mostrar los campos para generar un reporte dinámico, que haya sido registrado anteriormente en el sistema.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son los requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste, como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Los requisitos no funcionales representan

aquellos atributos que debe exhibir el sistema pero que no son funcionalidades específicas (Quesada, 2014).

RnF 1. Usabilidad: el sistema ABCD proporcionará una interfaz con facilidad de uso para usuarios con conocimientos mínimos de informática. Los grupos de botones y vínculos deben estar organizados por la funcionalidad, con el objetivo de facilitar al usuario la interacción con el sistema. Los mensajes para interactuar con los usuarios y los de error deben ser informativos e intuitivos y no deben revelar información interna.

RnF 2. Seguridad: el sistema ABCD debe garantizar la protección de los datos almacenados. Para ello se establecerán dos roles (Administrador y Usuario) garantizando que cada persona tenga acceso solamente a las funcionalidades según los permisos de su rol.

RnF 3. Confiabilidad: la base de datos será actualizada en tiempo real en caso de actualización, modificación o inserción de datos.

RnF 4. Requerimientos del Software: el servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos: Sistema Operativo: GNU/Linux, PostgreSQL 9.3 o superior, Máquina Virtual de Java (JVM) 1.7 (implementación de Oracle) y PgAdmin III o algún administrador para PostgreSQL.

RnF 5. Requerimientos del Hardware: la PC servidor microprocesador Intel Core i3-4005 a 1.70 GHz, 4GB de memoria RAM, espacio libre en disco duro de 20 GB para la instalación del sistema y la arquitectura del sistema debe ser de 64 bit.

2.4 Definición de los actores

Los actores del sistema no son más que sujetos, objetos o un mismo sistema que interactúa con este de forma externa (Quesada, 2014), que se relaciona con éste ya que solicita sus funcionalidades.

Tabla 3: Descripción de actores del sistema (Fuente: Elaboración propia)

Actor	Descripción
Usuario	Es la persona autenticada en el sistema y que accede al mismo con el fin de consultar o buscar un reporte y generarlo en la base de datos.

Administrador	Es la persona autenticada en el sistema y que accede al mismo con el fin de registrar, modificar, mostrar, eliminar y generar los reportes que se encuentren en la base de datos, los cuales podrán ser exportados en formato Excel o PDF.

2.5 Diagrama de casos de usos del sistema

El diagrama de casos de uso documenta el comportamiento de un sistema desde el punto de vista del usuario. Se utiliza para describir las funcionalidades de un software y documentar su comportamiento.

2.5.1 Definición de los casos de usos del sistema

- ✓ CU1. Gestionar Reporte Dinámico.
- ✓ CU2. Consultar Reporte Dinámico.
- ✓ CU3. Buscar Reporte Dinámico.
- ✓ CU4. Exportar a hoja de cálculo el Reporte.
- ✓ CU5. Exportar a PDF el Reporte.
- ✓ CU6. Generar Reporte Dinámico.

Diagrama de casos de uso del sistema

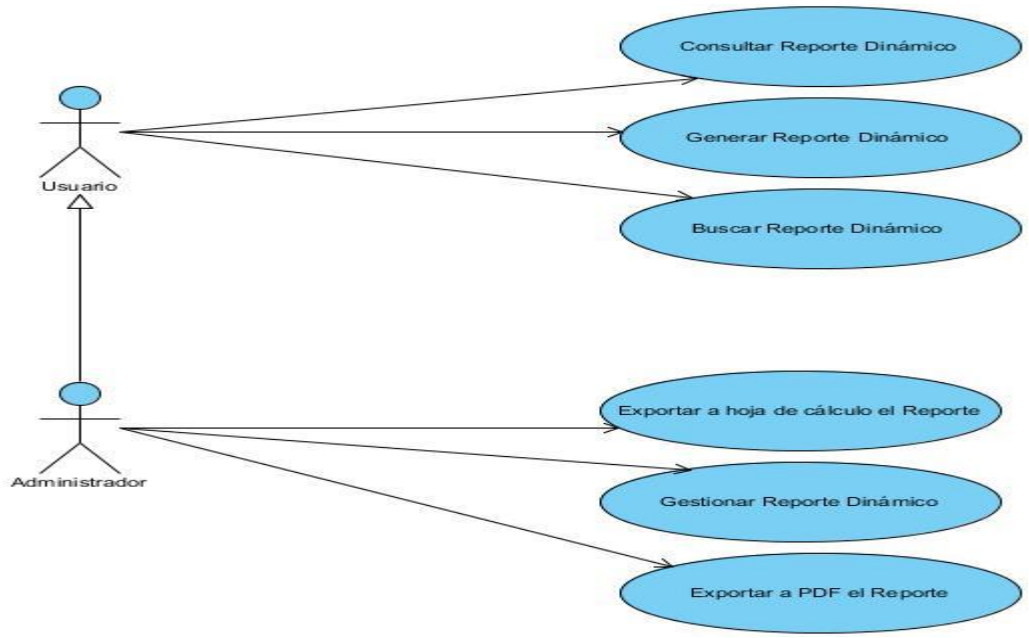


Figura 5:Diagrama de casos de uso del sistema (Fuente: Elaboración propia)

2.5.2 Descripción de casos de usos del sistema

Tabla 4:Descripción del CU Gestionar Reporte Dinámico (Fuente: Elaboración propia)

Caso de Uso	***Gestionar Reporte Dinámico***	
Objetivo	Gestionar en el sistema los datos de los reportes, teniendo la opción de crearlos, modificarlos, eliminarlos y mostrarlos en cualquier momento.	
Actores	Administrador(Inicia): crea, modifica, muestra y elimina los reportes en el sistema.	
Resumen	El caso de uso inicia cuando el administrador selecciona en el menú la opción Administrar Reporte Dinámico, el sistema muestra una interfaz, donde permite las acciones de crear, editar, mostrar y eliminar un reporte.	
Complejidad	Media	
Prioridad	Alta	
Precondiciones	El sistema debe estar instalado y ejecutándose correctamente. El administrador debe estar autenticado en el sistema.	
Postcondiciones	Se creó, se editó, se eliminó correctamente y se visualizó el reporte.	
Flujo de eventos		
Flujo básico Gestionar Reporte Dinámico		
	Actor	**Sistema**
1.	Indica que desea crear, modificar, mostrar o eliminar un reporte.	
2.		a) Si el usuario decide crear un reporte. Ver Sección 1: Registrar reporte dinámico. b) Si el usuario decide mostrar un reporte. Ver Sección 2: Mostrar reporte dinámico. c) Si el usuario decide modificar un reporte. Ver Sección 3: Modificar reporte dinámico. d) Si el usuario decide eliminar un reporte. Ver Sección 4: Eliminar reporte dinámico.
Sección 1: "Registrar Reporte Dinámico"		
Flujo básico <Registrar Reporte Dinámico >		

	Actor	Sistema
1.	Selecciona la opción que le permite “Registrar” un nuevo reporte.	
2.		Muestra una interfaz para introducir los datos del nuevo reporte: • Nombre • Consulta Permite las opciones: • “Aceptar” • “Cancelar”
3.	Selecciona e introduce los datos deseados.	
4.	Selecciona el botón “Aceptar”.	
5.		Verifica que se hayan introducido los campos obligatorios.
6.		Verifica que los datos introducidos sean correctos.
7.		Se muestra un mensaje de información “El Reporte ha sido creado correctamente”.
8.		Actualiza el listado de reportes dinámico con el nuevo elemento.
Flujos alternos <Registrar Reporte Dinámico >		
4ª. Cancelar		
1.	Presiona el botón “Cancelar”.	
2.		Muestra un mensaje de confirmación: “¿Está seguro que desea cancelar la operación?”. Una vez confirmado el mensaje se cancela la operación. Permite las opciones: • Aceptar • Cancelar
3.	Selecciona el botón “Aceptar”.	
4.		Cancela la operación de Registrar el reporte

		dinámico.
5.	Selecciona el botón “Cancelar”.	
6.		Vuelve a la interfaz de Registrar el reporte dinámico.
5ª. Campos obligatorios vacíos		
1.		Muestra un mensaje de error: “Existen campos obligatorios vacíos, por favor complete estos campos”.
6ª. Datos incorrectos		
1.		Muestra un mensaje de error: “La sentencia SQL es incorrecta”.
Sección 2: “Mostrar Reporte Dinámico”		
Flujo básico <Mostrar Reporte Dinámico >		
	Actor	**Sistema**
1.	Selecciona la opción que le permite “Mostrar” los datos de un reporte.	
		Muestra los datos del reporte seleccionado: • Nombre • Consulta Permite la opción: • “Cerrar”
2.	Selecciona la opción que permite “Cerrar” la vista actual.	
3.		Cierra la operación de Mostrar Reporte Dinámico.
Sección 3: “Modificar Reporte Dinámico”		
Flujo básico < Modificar Reporte Dinámico >		
	Actor	**Sistema**
1.	Selecciona la opción que le permite “Modificar” los datos de un reporte.	

2.		Permite modificar los datos del reporte seleccionado: • Nombre • Consulta Permite las opciones: • "Aceptar" • "Cancelar"
3.	Modifica los datos deseados.	
4.	Selecciona el botón "Aceptar".	
5.		Verifica que se hayan introducido los campos obligatorios.
6.		Verifica que los datos modificados sean correctos.
7.		Valida los nuevos metadatos del reporte y se muestra un mensaje de información "El Reporte ha sido modificado correctamente".
Flujos alternos <Modificar Reporte Dinámico >		
4ª. Cancelar		
1.	Presiona el botón "Cancelar".	
2.		Muestra un mensaje de confirmación: "¿Está seguro que desea cancelar la operación?". Una vez confirmado el mensaje se cancela la operación. Permite las opciones: • Aceptar • Cancelar
3.	Selecciona el botón "Aceptar".	
4.		Cancela la operación de Modificar el reporte dinámico.

5.	Selecciona el botón “Cancelar”.	
6.		Vuelve a la interfaz de Registrar el reporte dinámico.
5ª. Campos obligatorios vacíos		
1.		Muestra un mensaje de error: “Existen campos obligatorios vacíos, por favor complete estos campos”.
6ª. Datos incorrectos		
1.		Muestra un mensaje de error: “La sentencia SQL es incorrecta”.
Sección 4: “Eliminar Reporte Dinámico”		
Flujo básico <Eliminar Reporte Dinámico >		
	Actor	**Sistema**
1.	Selecciona la opción que le permite “Eliminar” el reporte seleccionado.	
2.		Muestra un mensaje de confirmación “¿Seguro que desea eliminar todos los elementos del reporte?”. Y permite: • “Aceptar” • “Cancelar”
3.	Selecciona el botón “Aceptar”.	
4.		Elimina el reporte seleccionado.
5.		Se muestra un mensaje de información “El Reporte ha sido eliminado correctamente”.
6.		Actualiza el listado de reportes dinámicos.
Flujos alternos <Eliminar Reporte Dinámico >		
2ª . Cancelar		
1.		Cancela las operaciones realizadas sobre el reporte.

Para un mayor entendimiento de las descripciones de los casos de usos del sistema, se pueden observar en los anexos (Ver Anexo 2).

Descripción del componente propuesto para la generación de reportes dinámicos

Capítulo 2

2.6 Arquitectura del componente

El diseño arquitectónico se ha descrito como un proceso de varios pasos en el cual las representaciones de la estructura de los datos y el programa, las características de la información y el detalle procedimental se sintetizan a partir de los requisitos (Pressman, 2005).

Cuando se habla de arquitectura de software, se hace alusión a la especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos; los requerimientos que debe satisfacer el mismo y las restricciones a las que está sujeto. La arquitectura está compuesta por propiedades no funcionales del sistema y su impacto sobre la calidad del mismo; las reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas. Es la estructura del sistema e incluye los componentes fundamentales del mismo. Proporciona un marco de referencia para guiar de manera más organizada la construcción del software entre los analistas, diseñadores, programadores y demás miembros del equipo de desarrollo (Pressman, 2012).

El componente que se desarrolla, utiliza una arquitectura N capas, ya que así lo requiere el sistema ABCD v3.0. La arquitectura n capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada. Los elementos que la integran se agrupan de forma lógica según la funcionalidad que reciben o suministran. Así, los elementos se limitarán a recibir peticiones de datos mientras que otros interactúan con el usuario y su función es fundamentalmente solicitar a otros elementos la información que el usuario precisa (Rodríguez, 2011).

Principales características que presenta la arquitectura N capas (Ortiz, 2012):

- Descomposición de los servicios de forma que la mayoría de interacciones ocurre solo entre capas vecinas.
- Las capas de una aplicación pueden residir en la misma máquina o pueden estar distribuidos entre varios equipos.
- Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces.
- Cada nivel agrega las responsabilidades y abstracciones del nivel inferior.

A continuación, se muestra en la siguiente figura, como es la interacción entre las capas de la arquitectura del componente a desarrollar:

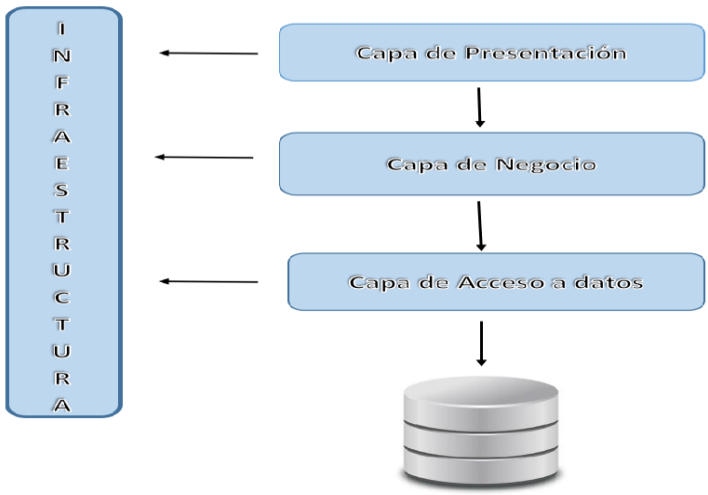


Figura 6: Dependencia entre las capas de la arquitectura en Capas (Fuente: Elaboración propia)

A continuación, se describen las capas principales de un patrón de arquitectura n capas (Lizardo, 2010):

Capa de presentación: Referente a la interacción entre el usuario y el software. Puede ser tan simple como un menú basado en líneas de comando o tan complejo como una aplicación basada en formas. Su principal responsabilidad es mostrar información al usuario, interpretar los comandos de este y realizar algunas validaciones simples de los datos ingresados.

Capa de negocio: También denominada Lógica de Dominio, esta capa contiene la funcionalidad que implementa la aplicación. Involucra cálculos basados en la información dada por el usuario y datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos. Se puede diseñar la lógica de la capa de negocios para uso directo por parte de componentes de presentación o su encapsulamiento como servicio y llamada a través de una interfaz de servicios que coordina la conversación con los clientes del servicio o invoca cualquier flujo o componente de negocio.

Capa de acceso a datos: Esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas por la aplicación. Estos pueden ser monitores transaccionales, otras aplicaciones, sistemas de mensajerías. Para el caso de aplicaciones empresariales, generalmente está representado por una base de datos, que es responsable por el almacenamiento persistente de información.

Capa de infraestructura: Esta capa es adyacente a todas las demás. Comprende todos aquellos servicios susceptibles de ser adquiridos desde cualquiera de las capas lógicas del sistema. Está constituida por los nodos de red que realizan la conmutación y encaminamiento de paquetes.

2.7 Modelo de diseño

El modelo del diseño se utiliza para documentar el diseño de un sistema. Es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de la implementación, tienen impacto en el sistema a considerar.

2.7.1 Diagrama de clases del diseño

Los diagramas de clases del diseño utilizando estereotipos web especifican la estructura de clases de un sistema, así como sus relaciones. Definen de forma correcta las relaciones de dependencia, generalización y asociación de clases que constituyen el sistema. A continuación, se muestran los diagramas de clases del diseño de la solución.

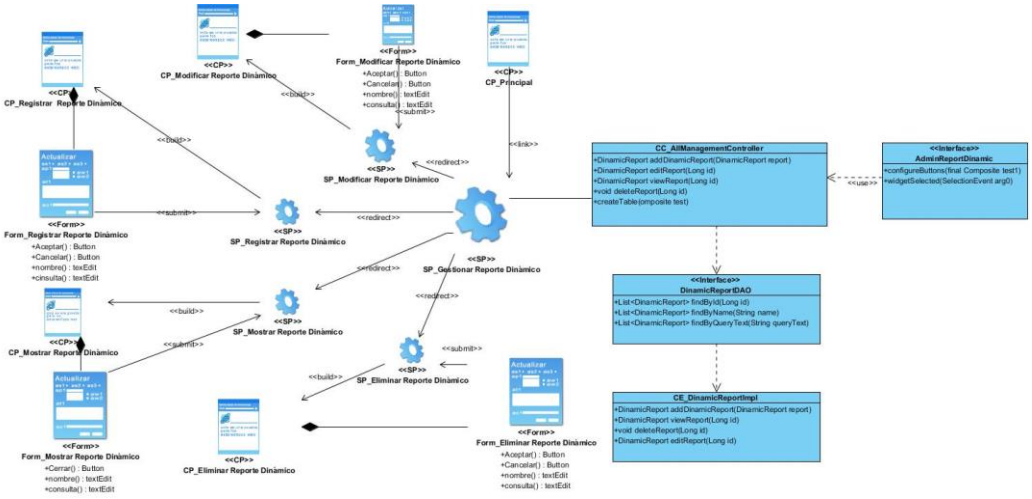


Figura 7: Diagrama de clases del diseño Gestionar Reporte Dinámico (Fuente: Elaboración propia)

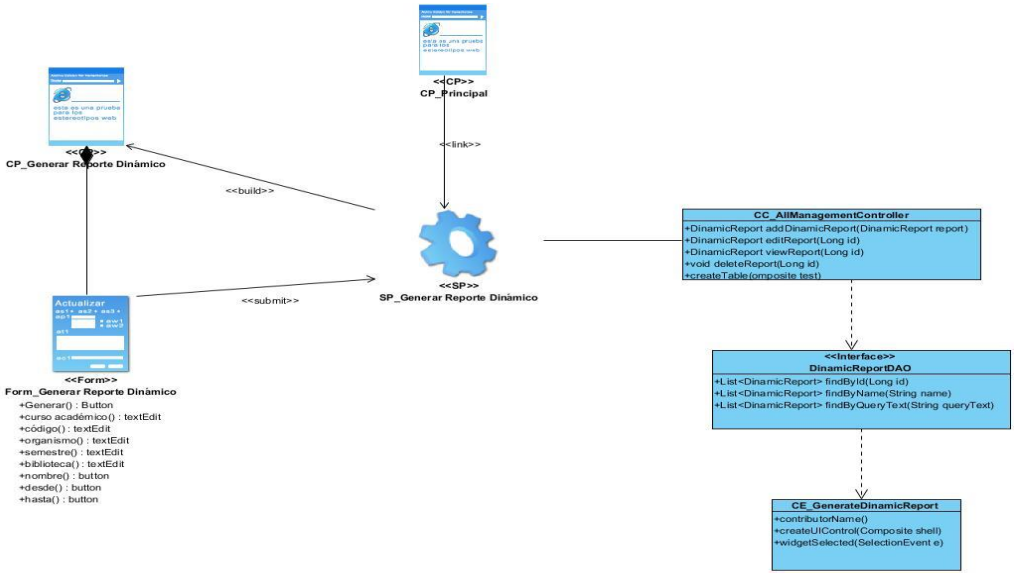


Figura 8: Diagrama de clases del diseño Generar Reporte Dinámico (Fuente: Elaboración propia)

2.8 Patrones de diseño utilizados en el desarrollo del componente

Los patrones de diseño proporcionan la facilidad de prevenir problemas y mejorar la legibilidad y claridad del código. Son una solución a un problema en un contexto, en donde destacan tres aspectos: el contexto, en el cual el patrón es utilizado; el problema que se está intentando solventar; y por último la solución que trata de aplicar una solución para alcanzar los objetivos (Alpízar, Rodriguez y Bolaños, 2017).

2.8.1 Patrones GRASP

Los patrones generales de software para asignación de responsabilidades (GRASP), describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Ayudan a entender el diseño del objeto esencial y aplicar el razonamiento para el diseño de una forma sistemática, racional y explicable. Los patrones GRASP sirven para asignar correctamente las responsabilidades en el diseño.

Los patrones GRASP a utilizar en la propuesta son:

Experto: Es asignar una responsabilidad al experto en información que es la clase que posee la información necesaria para cumplir con dicha responsabilidad (Larman, 2000). El patrón experto es usado en todas las clases ya que cada una posee la información correspondiente para realizar la tarea que le corresponde. Un ejemplo se evidencia en la clase *DinamicReportImpl*, la cual contiene todo lo referente a la gestión de los reportes dinámicos.

Bajo acoplamiento: el diseño se debe realizar de forma tal que la dependencia entre clases sea baja. De esta forma se tiene clases menos dependientes, logrando una reducción del impacto de los cambios y garantizando un mayor grado de reutilización. Se ve reflejado en la clase *DinamicReportImpl* que solo depende de la clase *AllManagementController y de DinamicReportDAO.*

Alta cohesión: asigna una responsabilidad de manera que la cohesión permanezca alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Esta caracteriza a las clases con responsabilidades altamente relacionadas y que no lleva a cabo gran cantidad de trabajo. Se ve reflejado en todas las clases implementadas. Específicamente en la clase *DinamicReportImpl* sus métodos son: addDinamicReport, viewReport, deleteReport y editReport.

Interface: este patrón define los parámetros de un tipo interfaz, todas las clases o instancias que quieran utilizar ese comportamiento deben implementar dicha interfaz. Se ve reflejado en la clase *AdminReportDinamic*.

Controlador: se evidencia en la clase controladora, cuya responsabilidad es interactuar con las clases de acceso a datos, obtener la información necesaria y dar respuesta a las peticiones del cliente controlando así el flujo central de acciones del sistema. Se ve reflejado en la clase *AllManagementController*, la cual es la encargada de manejar las peticiones realizadas por los usuarios sobre los reportes.

Patrón creador: el patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento (Larman, 2000). Ejemplo, la clase *DinamicReportImpl* en la que se crean las instancias de la clase reporte dinámico.

2.8.2 Patrones GOF

Los patrones de diseño GOF (*The Gang of Four*) son combinaciones de componentes, casi siempre clases y objetos que por experiencia se sabe que resuelven ciertos problemas de diseño comunes. Es una solución a un problema recurrente en el diseño de software, adaptada para resolver un problema general de diseño en un contexto particular. Se clasifican en tres categorías principales (Guerrero, Suárez y Gutiérrez, 2013):

- **Creacionales:** encierran conocimientos acerca de cuáles son las clases concretas que usa el componente. Tratan de la inicialización y configuración de clases y objetos.

- **Estructurales:** tratan con la composición de las clases y objetos. Tratan de desacoplar interfaz e implementación de clases y objetos.
- **Comportamiento:** caracterizan el modo en que las clases u objetos interactúan y distribuyen responsabilidades. Tratan de las interacciones dinámicas entre sociedades de clases y objetos.

El patrón GOF a utilizar en la propuesta es:

Objeto de Acceso a Datos (DAO): este patrón de diseño divide más las responsabilidades en la aplicación de tal forma que tendremos unas clases que se encargaran de la lógica de negocio y otras clases la responsabilidad de persistencia. Abstrae y encapsula los detalles de implementación a la fuente de datos y minimiza el nivel de acoplamiento entre clases, reduciendo la complejidad de realizar cambios (Álvarez, 2014). Se ve reflejado en la clase *DinamicReportDAO*.

Conclusiones del capítulo

En este capítulo se arribaron a las siguientes conclusiones:

1. La realización del modelo conceptual proporcionó una mayor comprensión de los conceptos asociados al negocio.
2. Mediante el levantamiento de requisitos se determinaron las funcionalidades que el sistema debe cumplir.
3. La revisión de los casos de usos, facilitaron el entendimiento del flujo del sistema
4. El empleo de patrones de diseño garantizó una solución que tiene como premisa la reutilización de código y la solución a problemas.

Capítulo 3: Implementación y pruebas del componente propuesto

En el presente capítulo se explican a partir de los resultados del diseño, la implementación del componente en el sistema ABCD v3.0. Se definen las pruebas de funcionalidad, unitarias y de aceptación al componente, mostrando además los resultados obtenidos durante la realización de las mismas.

3.1 Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físico (Hernandez, 2013).

3.2 Diagrama de despliegue

El diagrama de despliegue es un diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Permite modelar la disposición física o topología de un sistema, muestra el hardware y los componentes instalados en el hardware y permite mostrar las conexiones físicas entre el hardware y las relaciones entre componentes (Hernandez, 2013). A continuación, se muestra en la siguiente figura el diagrama de despliegue, donde se visualiza la distribución de los componentes de software en los nodos físicos.

Figura 9: Diagrama de despliegue (Fuente: Elaboración propia)

Se encuentra el nodo PC el cual representa las computadoras que utilizarán los usuarios para interactuar con la aplicación. Se comunica con el Servidor de aplicaciones Virgo v3.6.3 a través del protocolo HTTP por el puerto 8080, donde es montado el componente. Este servidor se conecta al servidor de bases de datos PostgreSQL donde se almacena la base de datos del sistema, mediante el protocolo TCP/IP por el puerto 5432.

3.3 Estándares de codificación

Los estándares de código son parte de las llamadas buenas prácticas o mejores prácticas. Estas son un conjunto no formal de reglas que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas pueden incrementar la calidad del código.

A continuación, se muestran los estándares de codificación que se utilizaron para la implementación del componente para la generación de reportes dinámicos y continuar con la línea del proyecto:

3.3.1 Upper Camel Case y Lower Camel Case

Existen un conjunto de nomenclaturas, pero la de excelencia en el mundo del lenguaje java, es Camel Case, que consiste en utilizar las mayúsculas como separadores de palabras.

Si comienza con mayúscula, se denomina *UpperCamelCase*, de lo contrario, *LowerCamelCase*. Por ejemplo, la frase “gestionar reporte dinámico” pasada a UpperCamelCase sería “GestionarReporte Dinámico” y a LowerCamelCase sería “gestionarReporteDinámico”.

3.3.2 Variables

Al declarar las variables y constantes, el nombre empleado debe permitir que con solo leerlo se conozca el propósito de la misma.

El nombre que se le asigna a las variables debe comenzar con la primera letra en minúscula e identificará el tipo de dato al que se refiere. En caso de que sea un nombre compuesto, la segunda palabra comenzará con letra inicial mayúscula.

Ejemplo: nameReport.

3.3.3 Constantes

Las constantes dentro de una clase se deben declarar todas en mayúsculas y las diferentes palabras que la compongan, separadas por barras bajas (_).
Ejemplo: LABEL_GENERATE_DINAMIC_REPORT.

3.3.4 Métodos

Los métodos de las clases deben estar escritos en *LowerCamelCase*. Se delimitará el cuerpo del método con llaves {}, la de apertura en la misma línea que se declara el método separado por un espacio, aunque puede estar en la siguiente línea. La llave de cierre debe estar en una línea diferente y no debe existir más código que la propia llave de cierre.

Ejemplo: private void createTable () {

//código

}

3.3.5 Clases

Los nombres de las clases deben estar escritos en *UpperCamelCase*. La indentación de las mismas debe ser de una tabulación con respecto al margen izquierdo, mientras que los campos y los métodos deben tener una sangría con respecto a la clase.

Ejemplo: public class AdminReportDinamic {

//código

}

3.4 Implementaciones relevantes en Java

A continuación, se muestra la implementación del método *CreateUI*, el cual se encuentra en la clase *DinamicReportAddEditableArea* perteneciente al componente **cu.uci.abcd.statistic.ui** donde se implementa la interfaz del requisito funcional Registrar reporte dinámico.

```
@Override
public Composite createUI(final Composite parent, IGridViewEntity entity, IVisualEntityManager arg2) {

    top = parent;
    setDimension(parent.getParent().getParent().getBounds().width);
    addComposite(parent);

    topGroup = new Composite(parent, SWT.NORMAL);
    topGroup.setData(RWT.CUSTOM_VARIANT, "gray_background");
    addComposite(topGroup);
    buildMessage(topGroup);
    headerLabel = new Label(topGroup, SWT.NORMAL);
    addHeader(headerLabel);

    addSeparator(new Label(topGroup, SWT.SEPARATOR | SWT.HORIZONTAL));
```

```
    nameLabel = new Label(topGroup, SWT.NORMAL);
    add(nameLabel);
    textlabelNameReport = new Text(topGroup, SWT.NONE);
    add(textlabelNameReport);
    validator.applyValidator(textlabelNameReport, "textlabelNameReport", DecoratorType.ALPHA_NUMERICS_SPACES, true, 255);
    validator.applyValidator(textlabelNameReport, "textlabelNameReport1", DecoratorType.REQUIRED_FIELD, true);
    br();

    queryLabel = new Label(topGroup, SWT.NORMAL);
    add(queryLabel);
    textQuery = new Text(topGroup, SWT.NONE);
    add(textQuery);
    validator.applyValidator(textQuery, "textQuery", DecoratorType.REQUIRED_FIELD, true);

    return parent;
}
```

Figura 10: Implementación del método createUI (Fuente: Elaboración propia)

A continuación, se muestra un fragmento de la implementación del método *CreateTabla,* el cual se encuentra en la clase *AdminReportDinamic* perteneciente al componente **cu.uci.abcd.statistic.ui** donde se implementa la tabla para el caso de uso Gestionar reporte dinámico.

```
private void createTable(Composite test) {
    tabla = new SecurityCRUDTreeTable(test, SWT.NONE);
    add(tabla, Percent.W100);
    tabla.addListener(SWT.Resize, new Listener() {
        private static final long serialVersionUID = 8817895862824622805L;
        @Override
        public void handleEvent(Event event) {
            refresh();
        }
    });
    tabla.setEntityClass(DinamicReport.class);
    tabla.setVisualEntityManager(new MessageVisualEntityManager(tabla));
    tabla.setSaveAll(false);
    tabla.setAdd(true, new DinamicReportAddEditableArea(controller));
    tabla.setWatch(true, new DinamicReportWatchEditableArea(controller));
    tabla.setUpdate(true, new DinamicReportUpdateEditableArea(controller));
    tabla.setDelete("deleteReportID");
    tabla.setPageSize(10);

    tabla.setSearchHintText(AbosMessages.get().BUTTON_SEARCH);
    tabla.setAddButtonText(AbosMessages.get().BUTTON_ADD);
    tabla.setSaveAllButtonText(AbosMessages.get().BUTTON_SAVE);
    tabla.setCancelButtonText(AbosMessages.get().BUTTON_CANCEL);

    CRUDTreeTableUtils.configUpdate(tabla);
```

Figura 11: Fragmento de la implementación del método createTabla (Fuente: Elaboración propia)

Implementación y pruebas del componente propuesto

Capítulo 3

3.5 Pruebas de software

Es el conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. Por tanto es necesario definir un conjunto de pasos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba. Las pruebas son el último bastión para la evaluación de la calidad y de manera más pragmática, el descubrimiento de errores (Pressman, 2011).

3.5.1 Estrategia de prueba

La estrategia de prueba es lo primero que se debe realizar antes de ejecutar las pruebas de un software, ya que mediante esta quedan plasmado los niveles de prueba a tratar, así como los tipos de pruebas que se deben llevar a cabo por cada nivel, los métodos de pruebas a aplicar y las técnicas a utilizar por el método.

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requieran. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Una estrategia de prueba de software debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto (Pressman, 2010).

3.5.2 Niveles de prueba

Las pruebas se aplican a diferentes tipos de destinos, en fases o niveles diferentes de esfuerzo de trabajo. Estos niveles suelen distinguirlos los roles que están más capacitados para diseñar y dirigir las pruebas, donde las técnicas son más adecuadas para realizar la prueba en cada nivel. Es importante asegurarse de que hay un equilibrio entre los diferentes esfuerzos de trabajo. Existen varios niveles de pruebas como: pruebas de desarrollador, independiente, unidad, integración, sistema y aceptación, y cada nivel contiene una técnica de prueba específica según los atributos de calidad que se deseen verificar (Jorrin, 2007).

A continuación, se muestran los niveles de pruebas que se aplicarán para el desarrollo del componente para comprobar que el sistema da respuesta a los requisitos funcionales definidos anteriormente:

- **Nivel de unidad:** consisten en aislar cada parte del programa (clases, módulos, objetos, paquetes, subsistemas) para comprobar que cada una de esas partes funciona correctamente por separado.

- **Nivel de sistema:** en este nivel las pruebas tienen como propósito ejercitar profundamente el sistema para verificar que se han integrado todos los elementos del sistema (hardware, software) y que realizan las funciones adecuadas. Los casos de prueba a diseñar en este nivel de pruebas, deben cubrir los aspectos funcionales y no funcionales del sistema.
- **Nivel de aceptación:** en este nivel se evidencian las pruebas realizadas por el usuario en un entorno muy similar al de producción para demostrar que el sistema cumple las especificaciones funcionales y requisitos del cliente. Son las últimas pruebas realizadas donde el cliente prueba el software y verifica que cumpla con sus expectativas, estas son fundamentales por lo cual deben incluirse obligatoriamente en el plan de pruebas de software.

3.5.3 Tipos de pruebas

Existen diferentes tipos de pruebas que se pueden aplicar para verificar que el componente cumple con todos los requisitos identificados en el proceso de análisis y comprobar su correcto funcionamiento. A continuación, se explican los tipos de pruebas seleccionados:

Pruebas unitarias: enfocan los esfuerzos de verificación en la unidad más pequeña del diseño de software: el componente o módulo de software. Al usar la descripción del diseño de componente como guía, las rutas de control importantes se prueban para descubrir errores dentro de la frontera del módulo. Se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente. Este tipo de pruebas puede realizarse en paralelo para múltiples componentes (Pressman, 2010).

Pruebas funcionales: son un proceso de control de calidad que consiste en asegurar el cumplimiento de un sistema o componente con requerimientos funcionales. El objetivo principal de las pruebas funcionales es analizar el producto terminado y determinar si hace todo lo que debería hacer y si lo hace correctamente. Este tipo de pruebas entran dentro de lo que se llaman pruebas de caja negra: aquí no se centra en cómo se generan las respuestas del sistema, solo se analizan los datos de entrada y los resultados obtenidos(Oterino, 2014).

Pruebas de aceptación: son realizadas con el cliente y define su aceptación del sistema. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos. Estas pruebas no se realizan durante el desarrollo, sino una vez pasadas todas las pruebas por parte del desarrollador. Pueden realizarse durante un período de semanas o meses, y mediante ellas

se descubren errores acumulados que con el tiempo puedan degradar el sistema. La mayoría de los constructores de productos de software usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer sólo el usuario final es capaz de encontrar (Pressman, 2010).

La *prueba alfa* se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador, se realizan en un ambiente controlado.

La *prueba beta* se realiza en uno o más sitios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba beta y los reporta al desarrollador periódicamente. En ocasiones se realiza una variación de la prueba beta, llamada prueba de aceptación del cliente, cuando el software se entrega a un cliente bajo contrato. El cliente realiza una serie de pruebas específicas con la intención de descubrir errores antes de aceptar el software del desarrollador.

3.5.4 Método de prueba y técnica

Dentro de los métodos de prueba existen dos fundamentales: el método de *Caja Blanca* y de *Caja Negra*.

Caja Blanca: se basa en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles. En ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que: 1) garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez, 2) revisen todas las decisiones lógicas en sus lados verdadero y falso, 3) ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas y 4) revisen estructuras de datos internas para garantizar su validez (Pressman, 2010).

El método de caja blanca aplica la técnica de la **prueba de ruta o trayectoria básica**. Este método de ruta básica permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba (Pressman, 2010).

Caja Negra: se refiere a las pruebas que se llevan a cabo en la interfaz del software. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del software. Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa (Pressman, 2010).

Las pruebas de caja negra pretenden encontrar estos tipos de errores (Pressman, 2010):

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en las estructuras de datos o en el acceso a bases de datos externas.
- Errores de comportamiento o rendimiento.
- Errores de inicialización y de terminación.

Partición de equivalencia: Consiste en clasificar las entradas de datos del sistema en grupos que presentan un comportamiento similar, por lo cual serán procesados de la misma forma. Se pueden definir particiones tanto para valores válidos como no válidos.

La técnica que se va a emplear para probar el componente para generar reportes dinámicos en el módulo de Estadísticas del sistema ABCD 3.0 es la técnica de **partición de equivalencia,** ya que esta técnica de prueba de caja negra divide el dominio de entrada de un programa en clases de datos, a partir de las cuales se deriva los casos de prueba.

3.5.5 Diseño de caso de prueba

Los casos de prueba incluyen todas las funciones que el programa es capaz de realizar. Deben tener en cuenta el uso de todo tipo de datos de entrada/salida, cada comportamiento esperado, todos los elementos de diseño, y cada clase de defecto.

Un caso de prueba es un conjunto de acciones con resultados y salidas previstas basadas en los requisitos de especificación del sistema; sus componentes son (Aristegui, 2010):

1. Propósito de la prueba o descripción del requisito que se está probando.
2. Método o forma como se probará.

3. Versión o configuración de la prueba, versión de la aplicación en prueba, el hardware, el software, el sistema operativo, los archivos de datos.
4. Resultados acciones y resultados esperados o entradas y salidas.
5. Documentación de la prueba y sus anexos.

Para realizar las pruebas funcionales se utilizaron juegos de datos válidos e inválidos haciendo uso de la técnica de partición de equivalencia. A continuación, se muestran los casos de pruebas de caja negra aplicado a los CU:

Tabla 5: Caso de Prueba de Caja negra Gestionar Reporte Dinámico (Fuente: Elaboración propia)

Escenario	Variables		Descripción	Respuesta del sistema	Flujo central
	1	2			
Sección 1: Registrar reporte dinámico					
EC 1.1 Registrar reporte dinámico satisfactoriamente.	V	V	El componente permite adicionar un nuevo reporte dinámico con los datos correctos a la base de datos.	El administrador ingresa los datos correctamente y se muestra un formulario con los datos del reporte registrado.	1-Introducir el nombre del reporte y la consulta sql. 2-Presionar el botón de Aceptar.
EC 1.2 Datos incorrectos.	I	V	El administrador introduce el nombre del reporte o una sentencia SQL que no cumple con la estructura definida a la hora de adicionar el reporte.	Muestra el mensaje de error: “Existen datos incorrectos, por favor rectifíquelos” o “La sentencia SQL es incorrecta”.	1-Introducir el nombre del reporte o consulta de forma incorrecta. 2-Presionar el botón de Aceptar.
	V	I			
EC 1.3 Campos vacíos.	V	I	El administrador registra un reporte dejando campos vacíos.	Muestra el mensaje de error: “Existen campos obligatorios vacíos, por favor complete estos campos”.	1-No introducir el nombre del reporte o la consulta SQL. 2-Presionar el botón de
	I	V			

					Aceptar.
Sección 2: Modificar reporte dinámico					
EC 2.1 Modificar reporte dinámico satisfactoriamente.	V	V	El componente permite modificar un reporte dinámico con los datos correctos a la base de datos.	El administrador modifica los datos correctamente y se muestra el reporte con los datos modificado.	1-Modificar el nombre del reporte y la consulta SQL. 2-Presionar el botón de Aceptar.
EC 2.2 Datos incorrectos.	V	I	El administrador modifica el nombre del reporte o una sentencia SQL que no cumple con la estructura definida a la hora de adicionar el reporte.	Muestra el mensaje de error: "Existen datos incorrectos, por favor rectifíquelos" o "La sentencia SQL es incorrecta".	1-Modificar el nombre del reporte o consulta de forma incorrecta. 2-Presionar el botón de Aceptar.
	I	V			
EC 2.3 Campos vacíos.	V	I	El administrador modifica un reporte dejando campos vacíos.	Muestra el mensaje de error: "Existen campos obligatorios vacíos, por favor complete estos campos".	1-No introducir el nombre del reporte o la consulta SQL. 2-Presionar el botón de Aceptar.
	I	V			
Sección 3: Eliminar reporte dinámico					
EC 3.1 Eliminar reporte dinámico satisfactoriamente.	El administrador selecciona el reporte que desea eliminar.			Muestra el mensaje de advertencia: "Ha seleccionado la opción que eliminará los datos introducidos hasta el momento. ¿Desea	1-Selecciona el reporte que desea eliminar. 2-Presiona el botón de Aceptar.

		continuar?”.	
Sección 4: Mostrar reporte dinámico			
EC 4.1 Mostrar reporte dinámico.	El administrador selecciona el reporte que desea mostrar sus datos.	Permite mostrar los datos del reporte selecciondo.	1-Selecciona el reporte que desea mostrar sus datos. 2-Presiona el botón de Cerrar.

Tabla 6:Caso de Prueba de Caja negra Consultar Reporte Dinámico (Fuente: Elaboración propia)

Escenario	Variables 1	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Consultar reporte dinámico satisfactoriamente.	V	El componente permite realizar la búsqueda en el sistema y restringe el listado de reportes según los criterios definidos.	El administrador ingresa un nombre de un reporte existente el cual quiere mostrar sus datos.	1-Introducir el nombre del reporte. 2-Presionar el botón de Consultar.
EC 1.2 Sin coincidencias.	I	El componente permite realizar la búsqueda en el sistema y restringe el listado de reportes según los criterios definidos.	Muestra el mensaje informativo: “No se encontraron coincidencias”.	1-Introducir el nombre del reporte. 2-Presionar el botón de Consultar.

Tabla 7: Caso de Prueba de Caja negra Buscar Reporte Dinámico (Fuente: Elaboración propia)

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Nueva búsqueda del reporte dinámico.	El componente permite buscar un nuevo reporte dinámico con los datos correctos a la base de datos.	El componente permite limpiar los datos en los campos de búsqueda inicialmente introducidos y muestra el listado	1-Introducir el nombre del reporte . 2-Presionar el botón de Nueva búsqueda.

		de reportes con todos los existentes en el sistema..	

Tabla 8: Caso de Prueba de Caja negra Exportar a hoja de cálculo el reporte (Fuente: Elaboración propia)

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Exportar a hoja de cálculo el reporte.	El componente permite exportar un reporte dinámico con todos sus datos en formato Excel.	Permite "Abrir con" el archivo generado y se muestra en el formato de hoja de cálculo el reporte visualizado. Permite "Guardar archivo", especificando el directorio donde se almacena.	1-Seleccionar la acción a realizar. 2-Presionar el botón de Aceptar.

Tabla 9: Caso de Prueba de Caja negra Exportar a PDF el reporte (Fuente: Elaboración propia)

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Exportar a PDF el reporte.	El componente permite exportar un reporte dinámico con todos sus datos en formato PDF.	Permite "Abrir con" el archivo generado y se muestra en el formato PDF el reporte visualizado. Permite "Guardar archivo", especificando el directorio donde se almacena.	1-Seleccionar la acción a realizar. 2-Presionar el botón de Aceptar.

Tabla 10: Caso de Prueba de Caja negra Generar Reporte (Fuente: Elaboración propia)

Escenario	Variables			Descripción	Respuesta del sistema	Flujo central
	8	9	10			
EC 1.1 Generar Reporte	V	V	V	El componente permite adicionar un nuevo reporte	El administrador ingresa los datos correctamente y se muestra un formulario	1-Introducir los datos para generar el reporte.

Dinámico satisfactoriamente.				dinámico con los datos correctos a la base de datos.	con los datos del reporte registrado.	2-Presionar el botón de Generar.
EC 1.2 Datos incorrectos.	I	V	V	El administrador o usuario introducen datos incorrectos a la hora de generar el reporte.	Si la fecha final es menor que la fecha inicial se muestra el mensaje de error: "La Fecha Final debe ser mayor o igual que la Fecha Inicial".	1-Introducir el rango de fecha incorrecta a la hora de generar el reporte. 2-Presionar el botón de Generar.
	V	I	V			
	V	V	I			
EC 1.3 Campos vacíos.	I	V	V	El administrador o usuario dejan campos obligatorios vacíos a la hora de generar el reporte.	Muestra el mensaje de error: "Existen campo obligatorios vacíos, por favor complete estos campos".	1-No introducir en nombre del reporte a generar. 2-Presionar el botón de Generar.
	V	I	V			
	V	V	I			

3.6 Resultados de las pruebas

Para validar el correcto funcionamiento del software se realizaron las pruebas de caja negra a través de los casos de prueba basados en los casos de uso. Este proceso permitió verificar el cumplimiento de los requisitos funcionales del componente, donde los resultados de las pruebas que no fueron satisfactorios pasaron a ser no conformidades.

En la siguiente tabla se muestran las no conformidades que se identificaron en cada iteración, asociadas a cada caso de uso. En la primera iteración se identificaron 11 NC (No Conformidades), de ellas 9 fueron funcionales, una de validación y una de ortografía. En la segunda iteración 6 NC, de ellas 5 funcionales y una de validación. En la tercera iteración no se encontraron NC, quedando demostrado que los errores encontrados fueron resueltos en su totalidad, para un total de 17 NC. A continuación, se muestran las no conformidades que se identificaron en cada iteración:

Tabla 11: Resultados de las pruebas de funcionalidad (Fuente: Elaboración propia)

No. de Iteraciones	Total de no conformidades	No Procede	Resueltas
1ra Iteración	11	0	11

2da Iteración	6	0	6
3ra Iteración	0	0	0

A continuación, se muestra un gráfico (ver figura 9) con el número de iteraciones realizadas y la cantidad de NC detectadas en cada iteración basándose en la tabla anterior.

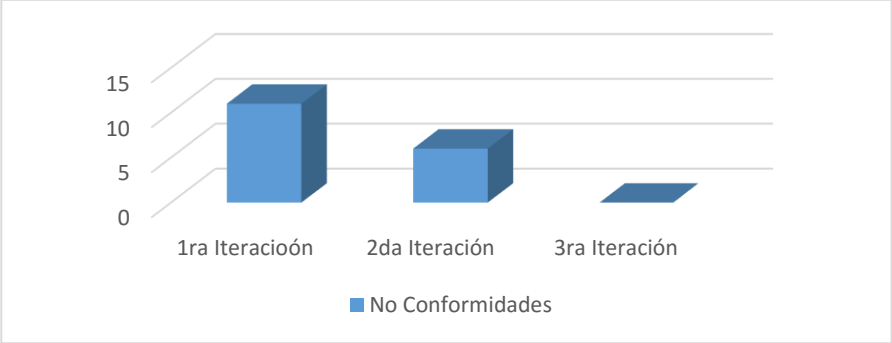


Figura 12: Gráfica de no conformidades de las pruebas funcionales (Fuente: Elaboración propia)

3.6.1 Pruebas unitarias

A continuación, se muestra el caso de prueba correspondiente al método **widgetSelected,** el cual se localiza en la clase **DinamicReportAddEditableArea**. El objetivo de este método es adicionar un nuevo reporte dinámico al sistema. Se selecciona este método teniendo en cuenta la importancia que representa para el resultado de este trabajo.

Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Para poder elaborar el grafo de flujo, primero se deben enumerar las sentencias del código.

A continuación, se muestra el código fuente referente al método descrito anteriormente y cómo fue aplicada la Técnica de Camino Básico al mismo:

```
public void widgetSelected(SelectionEvent e) {
1 if (validator.decorationFactory.isRequiredControlDecorationIsVisible()) {
    showErrorMessage(cu.uci.abos.core.l10n.AbosMessages.get().MSG_ERROR_FIELD_REQUIRED);
    } else if (validator.decorationFactory.AllControlDecorationsHide()) { 2
       DinamicReport report = new DinamicReport();
       report.setName(textlabelNameReport.getText().replaceAll(" +", " ").trim());
       report.setQueryText(textQuery.getText().replaceAll(" +", " ").trim());
       Report exist = ((AllManagementController) controller).getReport().findReportByName(report.getName());
        if (exist == null) { 3
          if (((AllManagementController) controller).getDinamicReport().validate(report)) 4
              ((AllManagementController) controller).getDinamicReport().addDinamicReport(report);
              manager.add(new BaseGridViewEntity<DinamicReport>(report));
                  manager.refresh();
                      } else { 5
                          showErrorMessage(AbosMessages.get().MESSAGE_INCORRECT_SQL);
                      }
                  } else { 6
                      showErrorMessage(AbosMessages.get().MESSAGE_NO_ADD);
                  }

              } else { 7
                  showErrorMessage(cu.uci.abos.core.l10n.AbosMessages.get().MSG_ERROR_INCORRECT_DATA);
          }
8 }
```

Figura 13: Código del método widgetSelected (Fuente: Elaboración propia)

Posteriormente se procede a la elaboración del grafo de flujo teniendo en cuenta dicha enumeración:

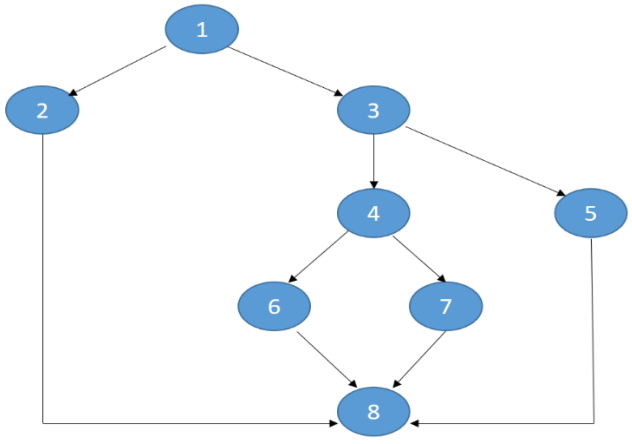


Figura 14: Representación del grafo de flujo de camino básico del método widgetSelected (Fuente: Elaboración propia)

La complejidad ciclomática es la métrica de software con que se define la cantidad de caminos independientes de cada una de las funcionalidades del programa y provee el límite superior para el

número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (Pressman 2010).

La complejidad ciclomática se basa en la teoría gráfica y se calcula de dos maneras distintas, donde, para que el cálculo sea correcto, todas deben arrojar el mismo resultado:

El número de regiones corresponde a la complejidad ciclomática “V (G)”.

V (G)= R Donde R es la cantidad total de regiones.

V (G)= 4

V (G)= E –N + 2 Donde E es el número de aristas y N número de nodos.

V (G)=10-8+ 2

V (G)= 4

V (G)= P +1 Donde P es la cantidad de nodos predicados

V (G)= 3 +1=4

El valor V (G) expresa la cantidad de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes 4 caminos:

Camino básico 1: 1-3-4-6-8

Camino básico 2: 1-3-4-7-8

Camino básico 3: 1-3-5-8

Camino básico 4: 1-2-8

Cada camino independiente es un caso de prueba a realizar, de forma que se garantiza que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. En el caso anterior se calcularon cuatro caminos básicos, por tanto, surge la necesidad de hacer igual número de casos de prueba. A continuación, se muestra uno de los casos de pruebas realizados:

Tabla 12: Caso de prueba de caja blanca para el camino básico 1 (Fuente: Elaboración propia)

Entrada	Nombre del reporte y consulta SQL.
Resultados Esperados	Se adiciona el nuevo reporte dinámico en el sistema.
Condiciones	Introducir los datos del nombre del reporte correctamente. La sentencia de la consulta SQL debe ser válida.

Tabla 13: Caso de prueba de caja blanca para el camino básico 2 (Fuente: Elaboración propia)

Entrada	Nombre del reporte y consulta SQL.
Resultados Esperados	Se adiciona el nuevo reporte dinámico en el sistema.
Condiciones	No se debe dejar campos vacíos obligatorios sin llenar. La sentencia de la consulta SQL debe ser válida.

Tabla 14: Caso de prueba de caja blanca para el camino básico 3 (Fuente: Elaboración propia)

Entrada	Nombre del reporte y consulta SQL.
Resultados Esperados	Se adiciona el nuevo reporte dinámico en el sistema.
Condiciones	No se debe dejar campos vacíos obligatorios sin llenar. La sentencia de la consulta SQL debe ser válida.

Tabla 15: Caso de prueba de caja blanca para el camino básico 4 (Fuente: Elaboración propia)

Entrada	Nombre del reporte y consulta SQL.
Resultados Esperados	Se adiciona el nuevo reporte dinámico en el sistema.
Condiciones	No se debe dejar campos vacíos obligatorios sin llenar. Introducir los datos del nombre del reporte correctamente. La sentencia de la consulta SQL debe ser válida.

Después de haber realizado las pruebas unitarias mediante la técnica de Camino básico al código seleccionado, se puede afirmar que la prueba concluye de forma satisfactoria, pues se obtuvo el resultado esperado.

3.6.2 Pruebas de aceptación

Para realizar la prueba de aceptación se empleó la técnica de prueba alfa, por el cliente y con especialistas del Centro de Informatización de la Gestión Documental. Se evaluaron las funcionalidades del componente basándose en la especificación de requisitos del expediente de proyecto. Los interesados verificaron cada requisito funcional y comprobaron si estos correspondían con los requerimientos planteados.

Implementación y pruebas del componente propuesto

Capítulo 3

Después de concluida las pruebas, se liberó el componente de generación de reportes dinámicos, entregándole al equipo de desarrollo la carta de aceptación en la que consta que el componente está apto para ser utilizado y cumple con las expectativas planteadas (Ver Anexo 5).

Conclusiones del capítulo

En este capítulo se arribaron a las siguientes conclusiones:

1. El uso de los estándares de codificación permitió lograr un estilo claro y organizado del código durante la fase de construcción.
2. Las pruebas de software a través del método de caja negra y caja blanca, aplicando la técnica de partición de equivalente y trayectoria básica respectivamente, permitieron conocer las no conformidades presentes en el componente.
3. Una vez finalizado el proceso de implementación se trazó un plan de estrategias de pruebas (pruebas unitarias, funcionales y aceptación), lo cual verificó el funcionamiento y la calidad del componente.

Conclusiones

Al concluir el desarrollo de la presente investigación se arribaron a las siguientes conclusiones:

1. Los fundamentos teóricos y metodológicos referentes a la generación de reportes dinámicos, así como los principales conceptos y tecnologías asociadas, permitió sentar las bases del proceso investigativo y conocer la lógica de negocio del sistema ABCD 3.0.
2. Se diseñó la propuesta de solución del componente, a partir de los artefactos correspondientes a la metodología de desarrollo AUP-UCI en su escenario 2, teniendo en cuenta el patrón arquitectónico N capas, lo cual facilitó su implementación.
3. Se desarrolló el componente para el módulo de Estadísticas del sistema ABCD 3.0, el cual permite la generación de reportes dinámicos a partir de los resultados obtenidos mediante las consultas SQL.
4. La validación del componente mediante la aplicación de las pruebas de software: unitarias, funcionales y de aceptación arrojaron resultados satisfactorios, quedando demostrado que el componente cumple con los requisitos funcionales.

Recomendaciones

Una vez concluida la investigación, la implementación y las pruebas correspondientes al componente para generar reportes dinámicos en el módulo de Estadísticas del sistema ABCD 3.0 se recomienda:

- Incorporar una nueva funcionalidad en el componente desarrollado que permita asociarle un indicador registrado por el administrador al reporte dinámico registrado en el sistema.
- Incorporar una nueva funcionalidad en el componente desarrollado que permita la representación gráfica de datos de los reportes generados.

Referencias bibliográficas

1. AGÜERO, Y.V. y LEYVA., M.R., 2014. *Herramienta para la generación de datos ficticios en bases de datos PostgreSQL y MySQL.* S.l.: Universidad de las Ciencias Informáticas.
2. ALPÍZAR, J.C.M., RODRIGUEZ, C.O. y BOLAÑOS, L.E., 2017. Patrones de diseño.
3. ÁLVAREZ, C., 2014. Patrones de Diseño (Active Record vs DAO). *2 de mayo del 2018* [en línea]. Disponible en: https://www.genbetadev.com/java-j2ee/patrones-de-diseno-active-record-vs-dao#.
4. ARISTEGUI, J.L., 2010. LOS CASOS DE PRUEBA EN LA PRUEBA DEL SOFTWARE . , no. 3, pp. 27-34.
5. BIDOPIA, I. y ARGÜELLES, I.F., 2016. GENERACIÓN DE ESTADÍSTICAS SOBRE DATOS NO RELACIONALES (J-ISIS).Universidad de las Ciencias Informáticas.
6. BOULY, Y.E., 2013. Componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g.Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.
7. GARDEY, J.P.P. y A., 2008. Definición de sistema de información. *15 de enero del 2018* [en línea]. Disponible en: https://definicion.de/sistema-de-informacion/.
8. GAVIO, R.F.B., BU, Y.R., HERNÁNDEZ, C.R., DELGADO, C.M., RIVERO y CORTÉS, M.C., 2014. GeReport: Sistema de Gestión de Reportes Dinámicos.
9. GIMSON, L., 2012. Metodologías ágiles y desarrollo basado en conocimiento.
10. GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E., 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *8 de junio del 2018* [en línea]. Disponible en: https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-07642013000300012.
11. HERNÁNDEZ, I.A.S., 2003. Generador Automático de Reportes Dinámicos.
12. HERNANDEZ, L., 2013. Modelo de Implementación. *22 de marzo del 2018* [en línea]. Disponible en: http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.
13. IRUELA, J., 2016. Los gestores de bases de datos más usados. *18 de abril del 2018* [en línea]. Disponible en: https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.
14. JORRIN, M.G., 2007. Proceso de pruebas para la liberación de productos software.Tesis de maestría. Ciudad de La Habana.
15. LARMAN, C., 2000. UML y Patrones. , vol. 2da Edición.

16. LIZARDO, M.E.A., 2010. Introducción al Patrón de Arquitectura por Capas. *26 de abril del 2018* [en línea]. Disponible en: https://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/.
17. MERINO, J.P.P. y M., 2017. Definición de reporte. *7 de diciembre del 2017* [en línea]. Disponible en: https://definicion.de/reporte/.
18. MOLEIRO, L.J.P. y GUTIERREZ, A.A.S., 2015. Herramienta informática de generación de reportes dinámicos basados en Apache Solr para el Sistema para repositorios digitales REPXOS 3.0. Wells.
19. MORALES, R., 2014. Lenguajes de programación: ¿qué son y para qué sirven? *8 de junio del 2018* [en línea]. Disponible en: https://colombiadigital.net/actualidad/articulos-informativos/item/7669-lenguajes-de-programacion-que-son-y-para-que-sirven.html.
20. NADER, I.J., 2003. Sistema de Apoyo Gerencial Universitario.
21. ORTIZ, H., 2012. Guia de Arquitectura en N Capas. *9 de mayo del 2018* [en línea]. Disponible en: http://guiaarquitecturancapas.blogspot.com/2012/11/estilos-arquitecturales.html.
22. OTERINO, A.M. del C.G., 2014. ¿Pruebas de integración, funcionales, de carga…? ¡Qué jaleo! ¿Qué diferencias hay? *14 de mayo del 2018* [en línea]. Disponible en: http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html.
23. PIÑEIRO, G.S.I., 2014. Sistema Informático para la evaluación de atributos de calidad en componentes biométricos. [en línea], vol. 3, pp. 17. Disponible en: https://www.3ciencias.com/wp-content/uploads/2014/03/SISTEMA-INFORMÁTICO-PARA-LA-EVALUACIÓN-DE-ATRIBUTOS-DE-CALIDAD-EN-COMPONENTES-BIOMÉTRICOS.pdf.
24. PRESSMAN, R., 2005. Ingeniería de software. Un enfoque práctico.
25. PRESSMAN, R.S., 2010. Ingeniería de Software, un enfoque práctico. , vol. SÉPTIMA ED.
26. PRESSMAN, R.S., 2011. Ingenieria de software, un enfoque practico.
27. PRIETO, J.Á.P., 2012. Sistemas Integrados de Gestión Bibliotecaria libres y de código abierto. , pp. 63.
28. QUELAL, G.P.J., 2011. SISTEMA DE EVALUACIÓN, CONTROL Y REPORTES PARA EL LABORATORIO DE EMAPA-I, PARROQUIA DE CARANQUI UTILIZANDO HERRAMIENTAS LIBRES.
29. QUESADA, J.A.L., 2014. Lenguajes de marcas y sistemas de gestión de información. Universidad de Murcia. *23 de enero del 2018* [en línea]. Disponible en: http://dis.um.es/~lopezquesada/documentos/IES_1415/LMSGI/curso/material.html#ut5.

30. QUINTUÑA, C.G.A. y B.R., 2010. Sistema Generador de reportes dinámicos para web, configurable para las plataformas de bases de datos más conocidas. Proyecto previo a la obtención del título de Ingeniero en Sistemas.
31. REUSE, K., 2013. Knowledge Reuse.
32. RODRÍGUEZ, B., 2011. Arquitectura de Aplicaciones N-capas. *8 de junio del 2018* [en línea]. Disponible en: http://blog.uca.edu.ni/blanca/2011/05/31/arquitectura-de-aplicaciones-n-capas/.
33. RODRÍGUEZ, J.C.B., 2011. MODEL DESIGNER FOR DYNAMIC REPORT GENERATOR 2.0.
34. ROMEU, I.D., 2016. SISTEMA INFORMÁTICO DE GESTIÓN DE INDICADORES PARA LA FORMACIÓN DE ESTUDIANTES POTENCIALMENTE TALENTOSOS EN LA UCI. , pp. 1-18.
35. SÁNCHEZ, J., 2017. Introducción a los Sistemas Gestores de Bases de Datos. *5 de febrero del 2018* [en línea]. Disponible en: https://jorgesanchez.net/presentaciones/bases-de-datos/pdf/sistemas-gestores-de-bases-de-datos.pdf.
36. SÁNCHEZ, T.R., 2015. Metodología de desarrollo para la Actividad productiva de la UCI. , pp. 1-16.
37. VALDERRAMA, J. sebastián, 2016. Herramienta CASE. *27 de abril del 2018* [en línea]. Disponible en: https://www.mindmeister.com/es/742289673/herramientas-case.

Bibliografía

1. MORALES, R., 2014. Lenguajes de programación: ¿qué son y para qué sirven? *8 de junio del 2018* [en línea]. Disponible en: https://colombiadigital.net/actualidad/articulos-informativos/item/7669-lenguajes-de-programacion-que-son-y-para-que-sirven.html.
2. RODRÍGUEZ, B., 2011. Arquitectura de Aplicaciones N-capas. *8 de junio del 2018* [en línea]. Disponible en: http://blog.uca.edu.ni/blanca/2011/05/31/arquitectura-de-aplicaciones-n-capas/.
3. GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E., 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *8 de junio del 2018* [en línea]. Disponible en: https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-07642013000300012.
4. LARMAN, C., 2000. UML y Patrones., vol. 2da Edición.
5. TAHIR, M., KHAN, F., BABAR, M., ARIF, F. y KHAN, S., 2016. Framework for Better Reusability in Component Based Software Engineering. [en línea], Disponible en: https://www.researchgate.net/profile/Fazlullah_Khan2/publication/304353982_Framework_for_Better_Reusability_in_Component_Based_Software_Engineering/links/576d0c2208aedab13b86c716.pdf.
6. HEE YOUNG CHUNG, JEONGJUN PARK, SEOK WON LEE, H.C., 2016. No TitleTBM risk management system considering predicted ground condition ahead of tunnel face: Methodology development and application. [en línea]. Disponible en: https://koreauniv.pure.elsevier.com/en/publications/tbm-risk-management-system-considering-predicted-ground-condition.
7. SAXENA, R., 2017. Formal Transformation of UML Diagram: Use Case, Class, Sequence Diagram with Z Notation for Representing the Static and Dynamic Perspectives of System. [en línea], vol. 409, pp. 25-38. Disponible en: https://link.springer.com/chapter/10.1007/978-981-10-0135-2_3.
8. MIRAKHORLI, M., 2015. Software architecture reconstruction: Why? What? How? [en línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/7081885/.
9. OTERINO, A.M. del C.G., 2014. ¿Pruebas de integración, funcionales, de carga…? ¡Qué jaleo! ¿Qué diferencias hay? *14 de mayo del 2018* [en línea]. Disponible en: http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html.

10. ARISTEGUI, J.L., 2010. LOS CASOS DE PRUEBA EN LA PRUEBA DEL SOFTWARE., no. 3, pp. 27-34.
11. ÁLVAREZ, C., 2014. Patrones de Diseño (Active Record vs DAO). *2 de mayo del 2018* [en línea]. Disponible en: https://www.genbetadev.com/java-j2ee/patrones-de-diseno-active-record-vs-dao#.
12. 2018, 22 de mayo del, 2017. Pruebas de Caja Negra y un enfoque práctico. [en línea]. Disponible en: https://testingbaires.com/pruebas-caja-negra-enfoque-practico/.
13. LIZARDO, M.E.A., 2010. Introducción al Patrón de Arquitectura por Capas. *26 de abril del 2018* [en línea]. Disponible en: https://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/.
14. ORTIZ, H., 2012. Guía de Arquitectura en N Capas. *9 de mayo del 2018* [en línea]. Disponible en: http://guiaarquitecturancapas.blogspot.com/2012/11/estilos-arquitecturales.html.
15. VARGAS, M., 2017. ¿Que son las pruebas de aceptación? [en línea]. Disponible en: https://losandestraining.com/2017/08/23/que-son-las-pruebas-de-aceptacion/.
16. ECHAVARRIA, R.A.R., 2014. Pruebas de Integración. [en línea]. Disponible en: https://prezi.com/0mpgx-lmytat/pruebas-de-integracion/.
17. SIMÕES, G.S., 2016. TRAZABILIDAD DE REQUERIMIENTOS: EVITA HASTA LAS MUERTES. [en línea], Disponible en: http://www.fattocs.com/files/es/presentaciones/Trazabilidad-10-2016-GuilhermeSimes.pdf.
18. ECHEVERRÍA, D., 2015. Plantilla de matriz de trazabilidad de requisitos. [en línea]. Disponible en: http://www.pmoinformatica.com/2015/05/matriz-de-trazabilidad-de-requisitos.html.
19. GONZALEZ, A., 2015. Patrón Singleton. [en línea]. Disponible en: http://codecriticon.com/patron-singleton/.
20. ALPÍZAR, J.C.M., RODRIGUEZ, C.O. y BOLAÑOS, L.E., 2017. Patrones de diseño.
21. PRESSMAN, R.S., 2010. Ingeniería de Software, un enfoque práctico., vol. SÉPTIMA ED.
22. HERNANDEZ, L., 2013. Modelo de Implementación. *22 de marzo del 2018* [en línea]. Disponible en: http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.
23. BIDOPIA, I. y ARGÜELLES, I.F., 2016. GENERACIÓN DE ESTADÍSTICAS SOBRE DATOS NO RELACIONALES (J-ISIS). Universidad de las Ciencias Informáticas.
24. ALVAREZ, W.A.R. y LISET SCHERY SÁNCHEZ, 2017. Guía de Instalación & Configuración ABCD 3.0 para Linux., pp. 30.

25. PRIETO, J.Á.P., 2012. Sistemas Integrados de Gestión Bibliotecaria libres y de código abierto., pp. 63.
26. SÁNCHEZ, T.R., 2015. Metodología de desarrollo para la Actividad productiva de la UCI., pp. 1-16.
27. JALÓN, J.G. de, 2000. Aprenda Java como si estuviera en primero.
28. MAURICIO, R., REYES, R., LORENA, V., GONZÁLEZ, G., ING, T. y RAMOS, B., 2016. «DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA INTEGRADO AL SIPGA (SISTEMA INTEGRADO PARA GESTIÓN ACADÉMICA) PARA EL DEPARTAMENTO DE CONSEJERÍA ESTUDIANTIL Y REPORTES PARA EL ÁREA ADMINISTRATIVA DE UNA INSTITUCIÓN DE EDUCACIÓN SECUNDARIA». UNIVERSIDAD DE GUAYA., pp. 136.
29. OTERO, M.C.R., [sin fecha]. Guía de uso de mendeley.
30. PIÑEIRO, G.S.I., 2014. Sistema Informático para la evaluación de atributos de calidad en componentes biométricos. [en línea], vol. 3, pp. 17. Disponible en: https://www.3ciencias.com/wp-content/uploads/2014/03/SISTEMA-INFORMÁTICO-PARA-LA-EVALUACIÓN-DE-ATRIBUTOS-DE-CALIDAD-EN-COMPONENTES-BIOMÉTRICOS.pdf.
31. SÁNCHEZ, J., 2017. Introducción a los Sistemas Gestores de Bases de Datos. *5 de febrero del 2018* [en línea]. Disponible en: https://jorgesanchez.net/presentaciones/bases-de-datos/pdf/sistemas-gestores-de-bases-de-datos.pdf.
32. GAVIO, R.F.B., BU, Y.R., HERNÁNDEZ, C.R., DELGADO, C.M., RIVERO y CORTÉS, M.C., 2014. GeReport: Sistema de Gestión de Reportes Dinámicos.
33. DELGADO, Y.C. y GONZÁLEZ, J.A.T., 2014. GESTACAD. SISTEMA PARA LA GESTIÓN ACADÉMICA.
34. QUELAL, G.P.J., 2011. SISTEMA DE EVALUACIÓN, CONTROL Y REPORTES PARA EL LABORATORIO DE EMAPA-I, PARROQUIA DE CARANQUI UTILIZANDO HERRAMIENTAS LIBRES.
35. BOULY, Y.E., 2013. Componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g. La Habana.
36. PRESSMAN, R., 2005. Ingeniería de software. Un enfoque práctico.
37. RODRÍGUEZ SÁNCHEZ, T., 2015. Metodología de desarrollo para la Actividad productiva de la UCI. Universidad de las Ciencias Informáticas, La Habana, Cuba.
38. MERINO, J.P.P. y M., 2017. Definición de reporte. *7 de diciembre del 2017* [en línea]. Disponible en: https://definicion.de/reporte/.

39. REUSE, K., 2013. Knowledge Reuse.
40. JOAN, A.M. y A., 2012. Generador Dinámico de Reportes. La Habana.
41. GIMSON, L., 2012. Metodologías ágiles y desarrollo basado en conocimiento.
42. ORIHUELA, S. y SÁNCHEZ, L.M., 2013. Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes. La Habana.
43. VALDERRAMA, J. Sebastián, 2016. Herramienta CASE. *27 de abril del 2018* [en línea]. Disponible en: https://www.mindmeister.com/es/742289673/herramientas-case.
44. RODRÍGUEZ, J.C.B., 2011. MODEL DESIGNER FOR DYNAMIC REPORT GENERATOR 2.0.,
45. DOUGLAS, B., 2007. Java para estudiantes.
46. QUINTUÑA, C.G.A. y B.R., 2010. Sistema Generador de reportes dinámicos para web, configurable para las plataformas de bases de datos más conocidas. Proyecto previo a la obtención del título de Ingeniero en Sistemas.
47. RUMBAUGH, J. y GRADY, B., 2000. El lenguaje unificado de modelado.
48. QUESADA, J.A.L., 2014. Lenguajes de marcas y sistemas de gestión de información. Universidad de Murcia. *23 de enero del 2018* [en línea]. Disponible en: http://dis.um.es/~lopezquesada/documentos/IES_1415/LMSGI/curso/material.html#ut5.
49. PRESSMAN, R.S., 2011. Ingeniería de software, un enfoque práctico.
50. IRUELA, J., 2016. Los gestores de bases de datos más usados. *18 de abril del 2018* [en línea]. Disponible en: https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.
51. PORTO, J.P. y MERINO, M., 2012. Definición de Lenguaje de Programación. *15 de enero del 2018* [en línea]. Disponible en: https://definicion.de/lenguaje-de-programacion/.
52. GARDEY, J.P.P. y A., 2008. Definición de sistema de información. *15 de enero del 2018* [en línea]. Disponible en: https://definicion.de/sistema-de-informacion/.
53. CABRERA, E.M.F., CHIRINO, R.R., SÁNCHEZ, R.G. y INFANTE, R.A.J., 2015. Sistema automatizado para la gestión de la información de pacientes con ictus. Revista de la Facultad de Educación 2015.
54. HERNÁNDEZ, I.A.S., 2003. Generador Automático de Reportes Dinámicos.
55. GONZÁLEZ, R.A.H.L. y S.C., 2011. El PROCESO DE INVESTIGACIÓN CIENTÍFICA.
56. SANTILLÁN, Á.G. y TORRES, J.M., 2007. Diseño y desarrollo reportes en formato PDF a partir de base de datos remotas en tiempo real con php y Mysql.

57. VIKI, C.G.A., 2010. SISTEMA GENERADOR DE REPORTES DINÁMICOS PARA WEB, CONFIGURABLE PARA LAS PLATAFORMAS DE BASES DE DATOS MÁS CONOCIDAS.
58. GACHET, D.O. de, 2010. INSTRUCTIVO PARA LA GENERACIÓN DE CONTRATOS POR COSTOS ADICIONALES Y ASOCIACIÓN CON LA CERTIFICACIÓN PRESUPUESTARIA.
59. PUPO, A.A.B., 2013. Pyramide: plugin del Eclipse para la evaluación de la estabilidad en los subsistemas de Cedrux.
60. BOULY, Y.E., 2013. Componentes para la generación de reportes dinámicos en el GDR sobre bases de datos en Access y Oracle 11g.Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.
61. AGÜERO, Y.V. y LEYVA., M.R., 2014. *Herramienta para la generación de datos ficticios en bases de datos PostgreSQL y MySQL.* S.l.: Universidad de las Ciencias Informáticas.
62. ROMEU, I.D., 2016. SISTEMA INFORMÁTICO DE GESTIÓN DE INDICADORES PARA LA FORMACIÓN DE ESTUDIANTES POTENCIALMENTE TALENTOSOS EN LA UCI., pp. 1-18.
63. MARTÍNEZ, P.A.H., 2015. Sistema informático para la administración visual del Servidor Dinámico de Reportes.
64. NADER, I.J., 2003. Sistema de Apoyo Gerencial Universitario.

Anexos

Anexo 1: Datos de la entrevista realizada

Tabla 18: Entrevista (Fuente: Elaboración propia)

Entrevista	
Objetivos	➢ Definir los requisitos funcionales y no funcionales del sistema. ➢ Definir la arquitectura del componente. ➢ Definir la problemática del módulo de Estadísticas.
Personas entrevistadas	Ing. Luis Carlos Alvarez Fernández Ing. Leandro Tabares Martín
Lugar de la entrevista	Laboratorio 101, Departamento de Programación, Docente 1, UCI
Modo de realización	Diálogo
Aspectos a tener en cuenta	➢ Arquitectura del sistema ABCDv3.0. ➢ Funcionamiento del módulo de Estadísticas, deficiencias. ➢ Características del componente a desarrollar.
Preguntas abordadas en las entrevistas	1) ¿Qué arquitectura presenta el sistema ABCD v3.0? 2) ¿Cuáles son las características del componente que se desea desarrollar? 3) ¿Qué deficiencias presenta el módulo de Estadísticas para generar un reporte? 4) ¿Cuáles son los requisitos funcionales con los cuales debe cumplir el componente? 5) ¿Qué metodología de desarrollo de software emplea el proyecto ABCD?

Anexo 2: Descripción de los CUS

Anexo 2.1: *Consultar Reporte Dinámico*

Caso de Uso	***Consultar Reporte Dinámico***	
Objetivo	Buscar el reporte en la Base de Datos.	
Actores	Administrador, Usuario	
Resumen	El caso de uso inicia cuando el Usuario o Administrador escoge la opción de Consultar un reporte. Se muestra el reporte existente en el sistema.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El reporte debe estar creado.	
Postcondiciones	Se mostró todos los datos del reporte.	
Flujo básico <Consultar Reporte Dinámico >		
	Actor	**Sistema**
1.		Muestra una interfaz para introducir los datos del reporte a consultar: • Nombre Permite la opción: • "Consultar"
3.	Introduce los datos deseados para realizar una búsqueda.	
4.	Selecciona la opción que le permite "Consultar" los datos de un reporte.	
5.		Verifica que exista coincidencia con algún reporte.
6.		Muestra el reporte con sus daros que coincide con el nombre introducido en el sistema.
Flujos alternos <Consultar Reporte Dinámico >		
5ª.No existe coincidencia		
1.		Muestra el mensaje informativo: "No se encontraron coincidencias".

Anexo 2.2: *Buscar Reporte Dinámico*

Caso de Uso	***Buscar Reporte Dinámico***	
Objetivo	Buscar el listado de reportes en la Base de Datos.	
Actores	Administrador, Usuario	
Resumen	El caso de uso inicia cuando el Usuario o Administrador escoge la opción de Nueva búsqueda. Se muestra el listado con todos los reportes existentes en el sistema.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El reporte debe estar creado.	
Postcondiciones	Se mostró el listado de reportes con todos los existentes en el sistema.	
Flujo básico <Buscar Reporte Dinámico >		
	Actor	**Sistema**
1.		Muestra una interfaz para introducir los datos del reporte a consultar: • Nombre Permite la opción: • “Nueva búsqueda”
3.	Introduce los datos deseados para realizar una búsqueda.	
4.	Selecciona la opción que le permite “Nueva Búsqueda” los datos de un reporte.	
6.		Limpia los datos en los campo de búsqueda inicialmente introducido y muestra el listado de reportes existentes en el sistema.

Anexo 2.3: *Exportar a hoja de cálculo el Reporte*

Caso de Uso	**Exportar a hoja de cálculo el Reporte**
Objetivo	Exportar el reporte al formato de hoja de cálculo.
Actores	Administrador, Usuario
Resumen	El caso de uso inicia cuando el Administrador escoge la opción de Exportar a hoja de cálculo. Se muestra el reporte con todos sus datos exportados en

	formato de hoja de cálculo.
Complejidad	Media
Prioridad	Media
Precondiciones	El reporte debe estar creado.
Postcondiciones	Se mostraron todos los datos de los reportes exportados en formato de hoja de cálculo.

Flujo básico < Exportar a hoja de cálculo el Reporte >

	Actor	Sistema
1.	Selecciona la opción que le permite “Exportar a hoja de cálculo”.	
2.		Permite “Abrir con” el archivo generado y se muestra en el formato de hoja de cálculo el reporte visualizado. Permite “Guardar archivo”, especificando el directorio donde se almacena. Permite las opciones: • “Aceptar” • “Cancelar”
3.	Selecciona el botón “Aceptar”.	
4.		Abre el reporte o lo guarda en formato de hoja de cálculo.
5.	Selecciona el botón “Cancelar”.	
6.		Cancela la operación de Exportar a hoja de cálculo el Reporte.

Anexo 2.4: *Exportar a hoja de cálculo el Reporte*

Caso de Uso	**Exportar a PDF el Reporte**
Objetivo	Exportar el reporte al formato de PDF.
Actores	Administrador, Usuario
Resumen	El caso de uso inicia cuando el Administrador escoge la opción de Exportar a PDF. Se muestra el reporte con todos sus datos exportados en formato de PDF.
Complejidad	Media
Prioridad	Media
Precondiciones	El reporte debe estar creado.
Postcondiciones	Se mostraron todos los datos de los reportes exportados en formato de PDF.

Flujo básico < Exportar a PDF el Reporte >		
	Actor	**Sistema**
1.	Selecciona la opción que le permite “Exportar a PDF”.	
2.		Permite “Abrir con” el archivo generado y se muestra en el formato de hoja de cálculo el reporte visualizado. Permite “Guardar archivo”, especificando el directorio donde se almacena. Permite las opciones: • “Aceptar” • “Cancelar”
3.	Selecciona el botón “Aceptar”.	
4.		Abre el reporte o lo guarda en formato de PDF.
5.	Selecciona el botón “Cancelar”.	
6.		Cancela la operación de Exportar a PDF.

Anexo 2.5: *Generar Reporte Dinámico*

Caso de Uso	***Generar Reporte Dinámico***	
Objetivo	Generar reportes en la Base de Datos, según los campos especificados.	
Actores	Administrador, Usuario	
Resumen	El caso inicia cuando el Usuario o Administrador escoge la opción de Generar reporte. Se muestra los campos para generar un reporte.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El reporte debe estar creado.	
Postcondiciones	Se mostró mediante una tabla los datos del reporte generado en el sistema.	
Sección 1: "Generar Reporte Dinámico"		
Flujo básico <Generar Reporte Dinámico >		
	Actor	**Sistema**
1.		Muestra una interfaz para introducir los datos del nuevo reporte: • Curso académico • Código • Organismo • Semestre • Biblioteca • Nombre • Fecha inicio • Fecha fin Permite la opción: • "Generar"
2.	Introduce los datos deseados para generar el reporte.	
3.	Selecciona el botón "Generar".	
4.		Verifica que se hayan introducido los campos obligatorios.
5.		Verifica que los datos introducidos sean correctos.

6.		Muestra una tabla con todos los datos pedidos en la consulta SQL a la hora de Registrar un reporte dinámico. Permite las opciones: • Exportar a hoja de cálculo • Exportar a PDF • Cancelar
Flujos alternos < Generar Reporte Dinámico >		
4ª. Campos obligatorios vacíos		
1.		Muestra el mensaje de error: “Existen campos obligatorios vacíos, por favor complete estos campos”.
5ª. Datos incorrectos		
1.		Muestra el mensaje de error: “La Fecha Final debe ser mayor o igual que la Fecha Inicial”.
4ª. Cancelar		
1.		Cancela las operaciones realizadas sobre el reporte, limpiando la tabla para volver a generar un reporte.

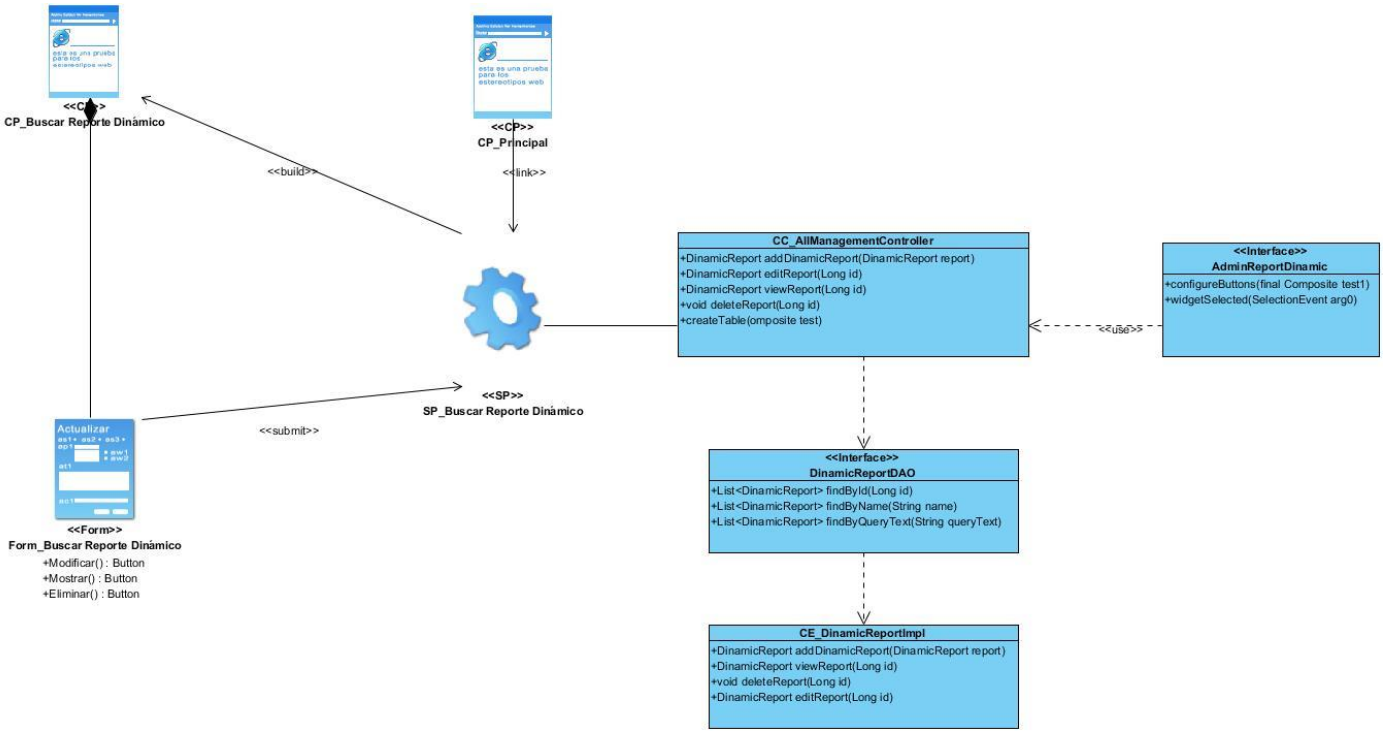
Anexo 3: Descripción de las variables de los casos de usos

No.	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Nombre del reporte	campo de texto	No	Es el nombre del reporte dinámico a registrar, para el cual solo se permiten letras y números o letras acompañadas de números. -Datos Válidos: 14, reporte, consul16. -Datos Inválidos: report-12, consulta*, permiso_34.
2	Consulta	campo de texto	No	Es la consulta sql con la cual se registra el reporte dinámico a generar. -Datos Válidos: Select id as ID from abcdn. dreport -Datos Inválidos: Delete * from abcdn.dreport
3	Curso académico	campo de texto	No	Es la identificación del curso académico en el cual se registra el reporte para generarlo.
4	Código	campo de texto	No	Es el número de identificación del reporte para ser generado.
5	Organismo	campo de texto	No	Es el organismo al cual pertenece el reporte el cual será generado.
6	Semestre	campo de texto	No	Es el nombre del semestre de curso del reporte para el cual será generado.
7	Biblioteca	campo de texto	No	Es el nombre de la biblioteca para la cual será generado el reporte dinámico.
8	Nombre	campo de selección	No	Se selecciona el reporte de la lista de reportes registrados en el sistema, el cual será posteriormente generado. -Datos Válidos: Se tiene que seleccionar un nombre para poder generar el reporte. -Datos Inválidos: No se selecciona un nombre de un reporte para ser generado.
9	Desde	campo de selección	No	Se selecciona la fecha de inicio para la cual se genera el reporte dinámico.

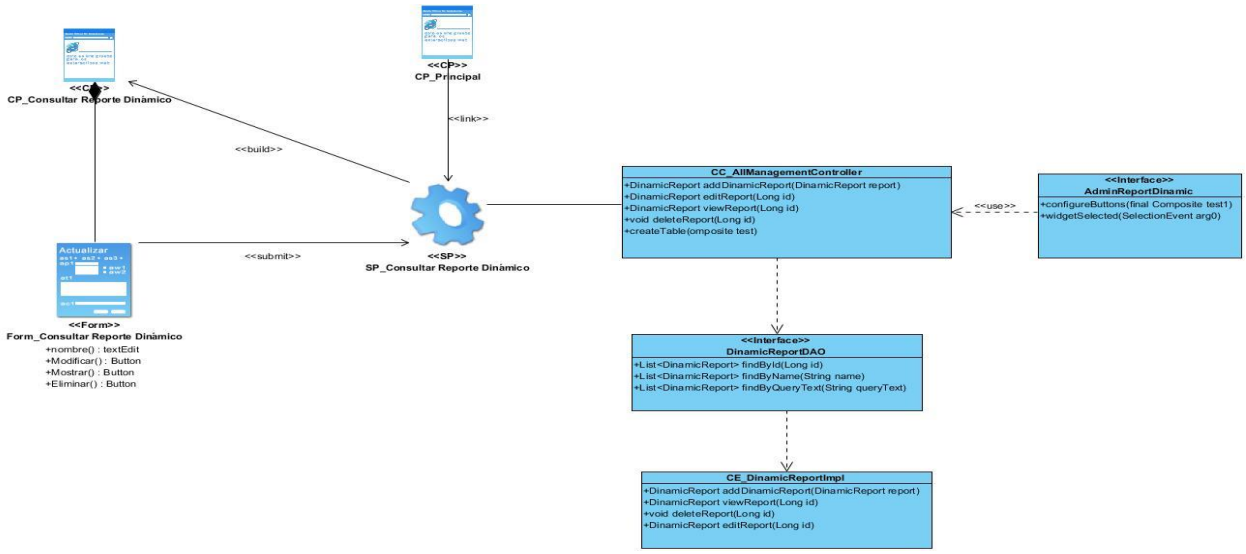
				-Datos Válidos: La fecha tiene que ser menor o igual que la fecha fin. -Datos Inválidos: La fecha es mayor que la fecha de fin.
10	Hasta	campo de selección	No	Se selecciona la fecha de fin para la cual se genera el reporte dinámico. -Datos Válidos: La fecha tiene que ser mayor o igual que la fecha de inicio. -Datos Inválidos: La fecha es menor que la fecha de inicio.

Anexo 4: Diagramas de clases del diseño

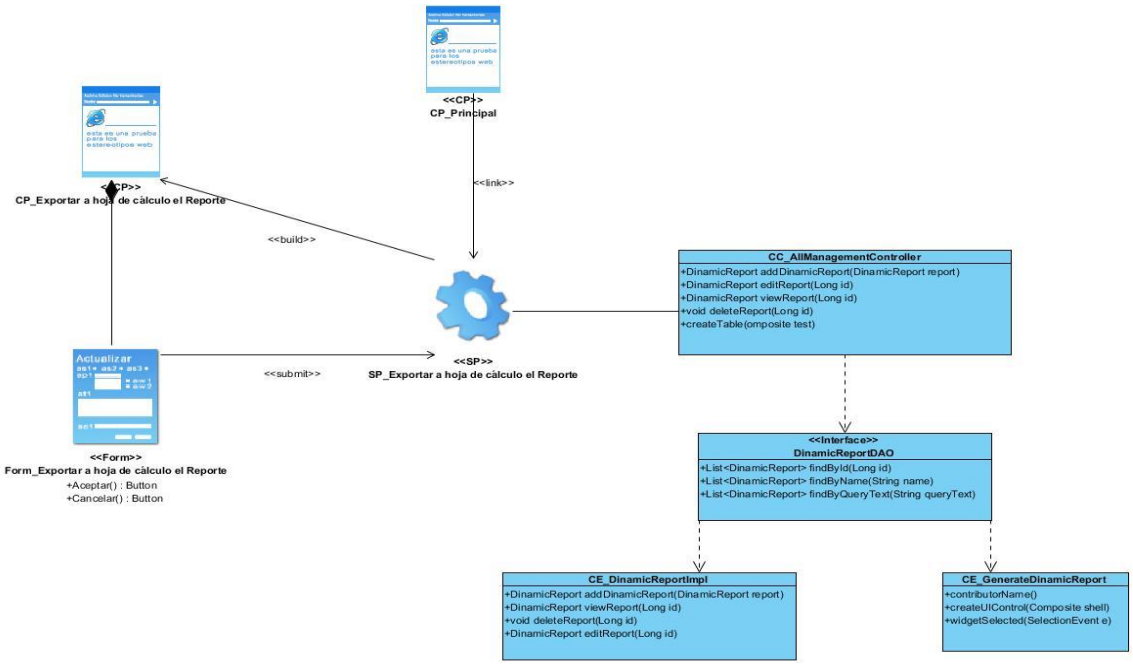
Anexo 4.1: Diagrama de clases del diseño Buscar Reporte Dinámico



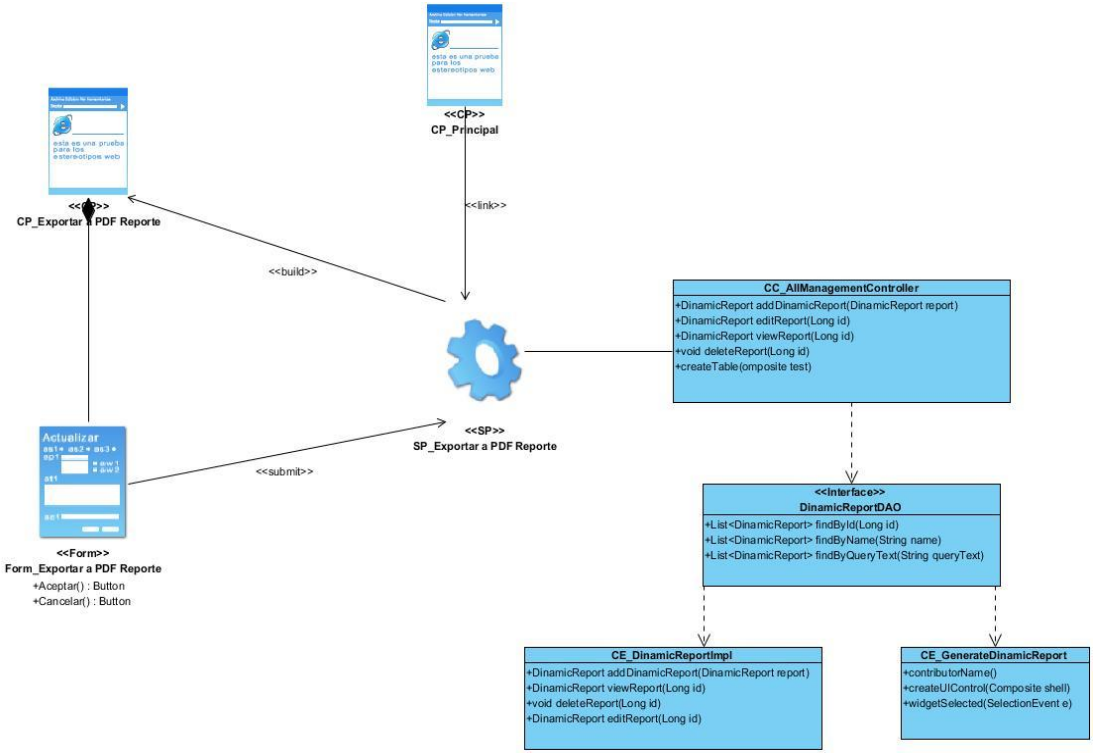
Anexo 4.2: Diagrama de clases del diseño Consultar Reporte Dinámico



Anexo 4.3: Diagrama de clases del diseño Exportar a hoja de cálculo el Reporte



Anexo 4.4: Diagrama de clases del diseño Exportar a PDF el Reporte



Anexo 5: Carta de Aceptación

UCI Universidad de las Ciencias Informáticas

Carta de aceptación

CARTA DE ACEPTACIÓN

En cumplimiento con la fase de desarrollo y en función de la ejecución del proyecto: Componente para la generación de reportes dinámicos en el sistema ABCD 3.0 para el Departamento de Componentes del Centro de Informatización de la Gestión Documental (CIGED) de la Facultad 2, se hace entrega de los productos que se relacionan a continuación:

- ***Generador de reportes dinámicos para el sistema ABCD 3.0***

Entrega	**Recibe:**	**Recibe:**
Nombre y apellidos:	**Nombre y apellidos:**	**Nombre y apellidos:**
Osmani Álvarez Pérez	Luis Carlos Álvarez Fernández	Anniel Sánchez Verdecia
Cargo: Tesista	**Cargo:** Jefe de Departamento	**Cargo:** Jefe de Proyecto
Firma:	Firma:	Firma:

Fecha: _____11/6/2018_____