UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Sistema de gestión de reportes para análisis de tendencias de calidad

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autora:

Lissandra Acosta Obregón

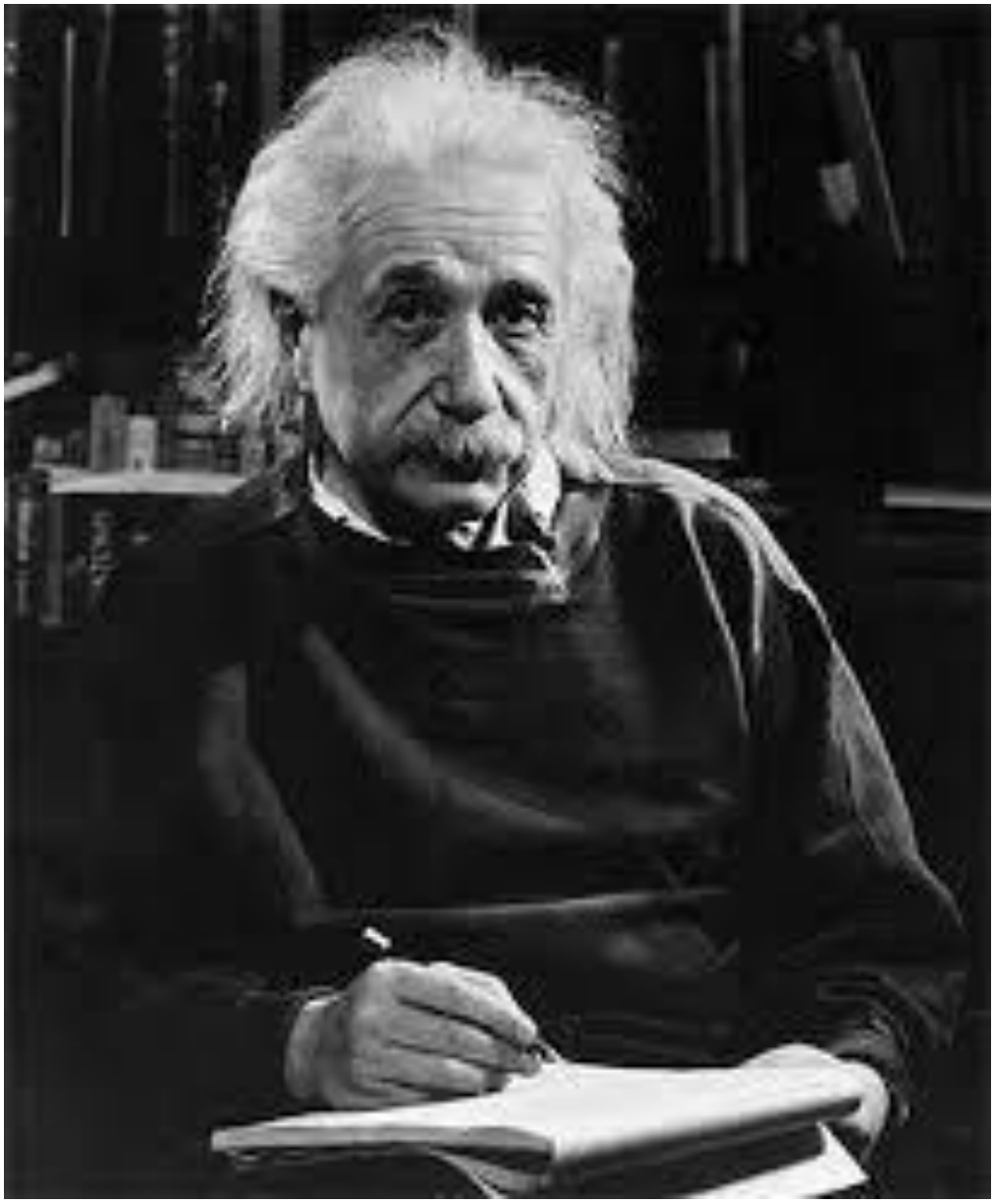
Tutores:

MSc. Yennifer Delgado Mesa

Ing. Paúl Rodríguez Leyva

La Habana, junio de 2019

"Año 60 de la Revolución"



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber"

Albert Einstein

AGRADECIMIENTOS

Primeramente le agradezco a quien a forjado mi camino y me ha dirigido por el sendero correcto, a Dios por haberme dado las fuerzas, la voluntad, paciencia y perseverancia necesaria para lograr el éxito en mis estudios.

Doy gracias a todos los profesores que ayudaron en mi formación como profesional durante los cinco años de la carrera, en especial a Yanet, Yordanka, Israel, Maikel, Nestor y Leiny.

En ese sentido le agradezco igualmente a mi tribunal de tesis, a mi oponente y a mis tribunales de los cortes ya que sus sugerencias me sirvieron de guía para lograr un excelente trabajo de diploma.

A mis tutores por haberme brindado la oportunidad de recurrir a sus capacidades, por su dedicación, motivación, criterio y sobre todo por haber tenido toda la paciencia del mundo para guiarme en el desarrollo de mi tesis.

Agradezco a mis compañeros de aula por tantos momentos de risas y por esas noches de estudio en especial a Maye, Liana, Mio, Leo, Luisito, Karel, Kuki,

Del mismo modo estoy muy agradecida con tres personitas muy importantes para mí, las cuales han sido las principales testigos de mi desarrollo durante todo el proceso de la universidad, les digo que en ustedes me llevo tres hermanas, gracias por su apoyo, por su comprensión, por saber estar siempre que las necesito y por demostrarme que la verdadera amistad sí existe. Mariana te agradezco tanto a ti como a tu familia el haberme abierto las puertas de tu casa y tratarme como una

más. Daniela te agradezco por demostrarme que la amistad no solo está en los buenos momentos. Aymee te agradezco por enseñarme que no importa el tiempo que pasemos sin vernos nuestra amistad siempre estará intacta.

A mis amigas Amanda y Arlet les agradezco por apoyarme siempre en cada decisión y por haber sido partícipe de cada momento de mi vida a pesar de la distancia.

Le agradezco a mi novio por haber sido mi mayor soporte en estos últimos dos años, por estar presente en cada momento cuando más lo necesité, por darme aliento cuando casi no me quedaban fuerza para seguir adelante y sobre todo por enseñarme el significado del amor verdadero.

A mi familia les estaré eternamente agradecida por ser el pilar principal que me impulsa cada día. A mi abuela Maruca le doy gracias por mimarme tanto y darme tanto amor. A mi tío Rene le agradezco por tanto cariño y apoyo. A mi tío Raulito le estoy muy agradecida por ser la persona que más confió en mi. A mis tías postizas (Tía Mory, Tía Titi, Tía Yari y Marta Maira) les agradezco por apoyarme cuando mi mamá estaba cumpliendo misión y velar por mi bienestar. A mi hermana Diana le agradezco por su atención y su preocupación.

Especialmente les agradezco a mis padres por todo su amor, su atención y su dedicación. Por enseñarme a ser cada día una mejor persona, quiero que hoy sean ustedes quienes obtengan todos los agradecimientos porque es por ustedes que estoy aquí, por el amor de hogar y la educación que me han dado, los amo. Muchas gracias a todos.

A los mejores maestros que he tenido le dedico este trabajo como regalo por guiarme en cada paso importante de mi vida y enseñarme el valor de luchar cada día por conseguir mis sueños.

Marleny y Raúl.

DECLARACIÓN DE AUDITORÍA

Declaro por este medio que yo: Lissandra Acosta Obregón, con carné de identidad 96100803934, soy la autora principal del trabajo final de tesis de pregrado que se titula: “Sistema de gestión de reportes para análisis de tendencias de calidad”. Autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

_____

Firma de la Autora

Lissandra Acosta Obregón

_____ _____

MSc. Yennifer Delgado Mesa Ing. Paúl Rodríguez

RESUMEN

La generación de reportes es una tarea fundamental en los sistemas que gestionan información, todo sistema que trabaje con gran volumen de datos necesita obtener reportes para poder llegar a conclusiones o a estimaciones. El Centro de Ideoinformática (CIDI) no cuenta con un sistema de reportes que se encargue de mostrar la información relacionada con la gestión de calidad de los productos elaborados en el mismo. Es por ello que el objetivo fundamental de esta investigación se basa en desarrollar un sistema de reportes, con tecnologías que contribuyan al desarrollo de la soberanía tecnológica y apoye el proceso de toma de decisiones en el manejo de información de tendencia. En la investigación se realiza el estudio de los lenguajes, herramientas y tecnologías necesarias para el desarrollo de la solución definidas por CIDI. Se definen los principales conceptos y se realiza un estudio de sistemas homólogos conservando las principales características que se adecuan a las necesidades del cliente. Además, se ejecutan las pruebas para el aseguramiento de la calidad del producto. Concluida la investigación se obtiene como resultado final un sistema que brinda información en forma de reportes y le permite al asesor de calidad de CIDI suministrar la información necesaria para la realización del informe de tendencias.

Palabras claves: calidad, generación de reportes, reporte, tendencias.

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo tecnológico alcanzado en las últimas décadas ha traído consigo un aumento en la automatización de procesos de la vida cotidiana mediante el uso del *software*. El mismo constituye el equipamiento lógico e intangible de un ordenador y es desarrollado mediante distintos lenguajes de programación, que permiten controlar el comportamiento de una máquina. Según Moraga y Cartes-Velásquez (2015) “Durante el proceso de desarrollo de *software* se pueden cometer varios errores como la mala estimación del tiempo y del costo, escatimar en el control de calidad, diseño inadecuado y confiar demasiado en tecnologías-herramientas no exploradas previamente”.

Un *software* es exitoso si se realiza una correcta gestión de la calidad, implicando la utilización de metodologías o procedimientos para el análisis, diseño, programación y prueba del *software.* Permitiendo uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba y que a la vez eleven la productividad, tanto para la labor de desarrollo, como para el control de la calidad del *software*. Es imprescindible tener en cuenta la calidad y control durante todas las etapas del ciclo de vida del *software* (Méndez, 2013).

Una forma de dar seguimiento a la calidad de los proyectos es la generación de reportes. Los mismos constituyen la conclusión de una investigación previa y adoptan una estructura de problema-solución en base a una serie de preguntas. En el caso de los informes impresos, el texto debe ir acompañado por gráficos, diagramas, tablas de contenido y notas al pie de página. En el ámbito de la informática, los reportes son informes que organizan y muestran la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios (Mera Paz, 2016). Por todo ello, se entiende que estos documentos sean tan importantes en cualquier empresa ya que permiten tener sus propias bases de datos y realizar recopilaciones de lo que se ha producido en un período determinado.

El 16 de agosto del 2012, tomando en cuenta el desarrollo alcanzado por la Industria Cubana del *Software* y el proceso de reorganización de la actividad del Ministerio de Comunicaciones (MINCOM), se crea el Centro Nacional de Calidad de *Software* (CALISOFT). Este centro como organización enfocada a contribuir al desarrollo de la industria, propone al MINCOM normativas, técnicas, políticas, procedimientos y estándares para las entidades que desarrollan aplicaciones informáticas. Tiene como objeto social la responsabilidad de evaluar la conformidad con las normas y emitir certificación a productos informáticos

nacionales o importados; brindar servicios de consultorías asociadas a la ingeniería y calidad de aplicaciones informáticas; controlar y auditar el uso de normativas técnicas, procedimientos, documentos estandarizados y buenas prácticas para el desarrollo de aplicaciones y sistemas informáticos en el país (CALISOFT, 2016).

La Universidad de las Ciencias Informáticas (UCI) comienza a tener una presencia productiva en la industria del *software* con el desarrollo de proyectos informáticos, fundamentalmente relacionados con la informatización de la sociedad y la exportación; por tal motivo, más de la mitad de los estudiantes se encuentran incorporados a proyectos productivos e investigativos de *software*. En el proceso de desarrollo de *software* de la UCI es prioridad medir la calidad de los productos, con este objetivo surge la Dirección de Calidad de *Software* que basa sus funciones en evaluar las aplicaciones y sistemas informáticos elaborados en los centros de producción; realizar consultorías y evaluaciones a los procesos de desarrollo de *software*, así como contribuir a la preparación de especialistas en temas asociados a la calidad de software.

En cada centro productivo de la UCI existen especialistas capacitados para desempeñar el rol asesor de calidad lo cuales examinan el cumplimiento de los siguientes requisitos (Mejoras del Proceso del *Software*, 2018):

1. Revisiones de adherencia a procesos y productos. Este tipo de actividad ayuda a prevenir la aparición de defectos en el código mediante la utilización de técnicas como: listas de chequeo, revisión documental y entrevistas.
2. Revisiones Técnicas Formales. Es un método probado para eliminar defectos de manera temprana y proporcionar una visión de valor sobre los productos de trabajo y los componentes de producto que están siendo desarrollados y mantenidos.
3. Pruebas. Actividades que representan una revisión final de las especificaciones, del diseño y de la codificación, con el objetivo de encontrar los posibles fallos que puedan existir en el producto de trabajo a evaluar.
4. Validaciones. Es la comprobación de que se han cumplido los requisitos para su uso específico y la idoneidad de lo esperado. El objetivo debe enmarcarse en demostrar que se construyó lo correcto.

Para garantizar que se cumpla la gestión de la calidad en los proyectos productivos de la Universidad, los asesores de calidad de cada centro de desarrollo realizan informes, en los cuales analizan las tendencias

de los procesos y productos de trabajo generados en cada proyecto productivo de los mismos. Actualmente este análisis de tendencias en el Centro de Ideoinformática se realiza de forma manual mediante varios *Excel* y no existen reportes que analicen los comportamientos de los indicadores de cada proyecto, lo que ocasiona que se puedan cometer errores en la redacción del reporte y no se permita controlar el comportamiento de los indicadores. La información no está centralizada y trae consigo que ocurran errores relacionados con la optimización de los tiempos, la pérdida de información referente al análisis de tendencia, atrasos en la entrega del informe de tendencia, se disminuye la productividad del trabajador y no se permite gestionar el tiempo de trabajo de manera que se obtenga la mayor productividad posible. Por otro lado, el flujo de la documentación de dicho proceso es complejo teniendo en cuenta que la información queda obsoleta cada cierto tiempo y realizar este análisis cada vez que se necesite supondría recolectar nuevamente un gran cúmulo de información por parte de los asesores. En la Universidad existe el sistema GESPRO que ofrece entre sus funcionalidades el seguimiento y el control de la calidad de los proyectos. El proceso de generación de reportes de dicho sistema presenta imperfecciones pues no permiten obtener todos los datos que se necesitan, se desconocen las fechas críticas, no apoya la toma de decisiones en el manejo de información de tendencias, se derrocha el tiempo al no realizar mantenimientos más efectivos de los recursos, los procesos cotidianos no generan sinergia en el proyecto, es decir, se aumente los niveles de integración y se tomen decisiones acertadas y ajustadas a la realidad.

Teniendo en cuenta la importancia que tiene la gestión de calidad durante el desarrollo de *software* y el papel que juegan los reportes en la misma surge el siguiente **problema de investigación:** ¿Cómo gestionar los reportes de tendencias de calidad para mejorar el proceso de análisis del estado de la calidad de los proyectos del Centro de Ideoinformática?

Se define como **objeto de estudio** el proceso de generación reportes para el análisis de tendencias de calidad. Se enmarca el **campo de acción** en el proceso de generación de reportes para análisis de tendencias de calidad en los proyectos del Centro de Ideoinformática.

A partir del problema de investigación se define como **objetivo general:**

Desarrollar un sistema de gestión de reportes de tendencias de calidad basado en técnicas de procesamiento de datos que mejore el proceso de análisis del estado de la calidad de los proyectos del Centro de Ideoinformática.

En función de cumplir con el objetivo planteado se formula la **Hipótesis:**

El desarrollo de un sistema de gestión de reportes de tendencias de calidad basado en técnicas de procesamiento de datos mejora el proceso de análisis del estado de la calidad de los proyectos del Centro de Ideoinformática.

Variable independiente

Un sistema de gestión de reportes de tendencias de calidad basado en técnicas de procesamiento de datos.

Variable dependiente

Proceso de análisis del estado de la calidad de los proyectos del Centro de Ideoinformática.

Dimensiones

Pertinencia

Eficiencia

Para dar cumplimiento a la hipótesis planteada se definieron los siguientes **objetivos específicos:**

1. Identificar los referentes teóricos metodológicos fundamentales que sustentan la investigación relacionados con el desarrollo de sistemas de gestión de reportes para análisis de tendencias de calidad.
2. Analizar el estado de la gestión de reportes para análisis de tendencias de calidad.
3. Implementar las funcionalidades del sistema de gestión de reportes para análisis de tendencias de calidad.
4. Validar las funcionalidades del sistema gestión de reportes para análisis de tendencias de calidad.

Para la resolución de la problemática anterior se utilizaron los **métodos de investigación** científica, tanto teóricos como empíricos.

Los métodos teóricos utilizados son:

1. **Analítico-Sintético**: Su objetivo en la investigación es analizar las teorías y documentos, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.
2. **Análisis Histórico-Lógico**: Su objetivo en la investigación es constatar teóricamente cómo ha evolucionado la gestión de calidad de *software* en un período de tiempo, en toda su trayectoria o en un fragmento temporal de la lógica de su desarrollo.
3. **Método hipotético-deductivo**: Para la elaboración de la hipótesis de la investigación.

Los **métodos empíricos** utilizados son:

1. **Entrevista**: Se emplea en los encuentros con el cliente para conocer las necesidades del desarrollo de la propuesta de solución, definir funcionalidades y las restricciones que se imponen.
2. **Observación**: Posibilita obtener conocimiento acerca del funcionamiento de los sistemas existentes en la actualidad para la gestión de reportes de tendencia de calidad.

Este documento se encuentra estructurado de la siguiente manera: Introducción, tres Capítulos que conforman el cuerpo del documento, Conclusiones, Recomendaciones, Bibliografía y Anexos.

Primer capítulo ‘‘Fundamentación teórica del sistema de gestión de reportes para análisis de tendencias de calidad’’: se exponen los conceptos asociados al dominio de la investigación. Se realiza una breve caracterización del sistema existente, el cual no da solución al problema planteado. También resume las herramientas, metodologías y lenguajes de programación que se van a utilizar en el desarrollo de dicho sistema.

Segundo capítulo ‘‘Análisis y diseño de sistema de gestión de reportes para análisis de tendencias de calidad’’: su objetivo principal es presentar la propuesta web que da solución al problema de gestión de reportes para análisis de tendencia de calidad. Posee las características y funcionalidades del sistema a partir de los requisitos funcionales y no funcionales identificados, relaciona los patrones de diseño, la arquitectura a utilizar en el sistema, en conjunto con otros elementos que componen la solución.

Tercer capítulo ‘‘Implementación y pruebas del sistema de gestión de reportes para análisis de tendencias de calidad’’ se basa en la presentación de los resultados de la implementación y las pruebas

realizadas a la aplicación.Teniendo en cuenta la importancia que tiene la gestión de calidad durante el desarrollo de *software* y el papel que juegan los reportes en la misma surge el siguiente problema de investigación: ¿Cómo gestionar los reportes de tendencias de calidad para mejorar el proceso de análisis del estado de la calidad de los proyectos del Centro de Ideoinformática?

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA DEL SISTEMA DE GESTIÓN DE REPORTES PARA ANÁLISIS DE TENDENCIAS DE CALIDAD

Introducción

En el presente capítulo se presentan los principales conceptos relacionados con la calidad del *software*. Se caracterizan algunas técnicas y herramientas que facilitan la generación de reportes para análisis de tendencia de calidad mediante la automatización de acciones, así como algunos sistemas informáticos que gestionan y le dan seguimiento a la generación de reportes para análisis de tendencia de calidad. Igualmente se describen las tecnologías a utilizar para el desarrollo de la solución dentro del entorno de desarrollo.

1.1. Conceptos asociados a la investigación

Calidad de *software*

El concepto de calidad de *software*, según (Betancourt Agüero, 2013) se asocia a la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo plenamente documentados y con las características implícitas que se espera de todo *software* desarrollado profesionalmente. Lo cual fomenta la aplicación de procesos estandarizados y criterios necesarios en cada una de sus etapas, así se promueve que el avance en el ciclo de vida del *software* minimice el riesgo de fracaso del proyecto. Por su parte, el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, 1990) define calidad de *software* como “el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”, denotando que el énfasis radica en los requisitos específicos del sistema y en la búsqueda de la satisfacción del cliente.

Características de la calidad de *software*

El control y la garantía de la calidad son actividades esenciales en cualquier negocio que elabora productos de consumo. En la actualidad, toda compañía tiene mecanismos que garantizan la calidad en sus productos. El proceso de desarrollo de *software* se rige por las siguientes características para crear productos con calidad (Mayo & Fernández, 2016):

Funcionabilidad: que el usuario pueda utilizar el *software.*

Confiabilidad: que los datos sean íntegros.

Usabilidad: fácil de usar, fácil de aprender a usar.

Portabilidad: compatible con otras plataformas.

Compatibilidad: visible y ejecutable en la plataforma que se ejecute.

Corrección: capaz de darle mantenimiento.

Eficiente: hace lo que debe bien, lo hace a tiempo y no derrocha recursos.

Robustez: que se mantenga en el rito que debe.

Oportunidad: fácil de acceder, en cualquier momento.

Reporte

Se denomina reporte a un cuerpo de información destinado a servir de análisis sobre un tópico determinado. Un reporte puede revestir diversas formas, ya sea como escrito, charla, informe televisivo o película documental. El uso de reportes se extiende en el plano gubernamental, en el privado, en el área educativa, en el campo científico, etcétera. Un reporte puede hacer uso de gran variedad de formas de representar la información para cumplir con su cometido (Oyarce, Orellana, & Flores, 2016).

Sistema de generación de reportes

En todo sistema de información es fundamental contar con una herramienta adicional cuyo objetivo sea la de generar reportes. Los sistemas generadores de reportes tienen en las bases de datos su principal fuente de alimentación, ya que a partir de la información almacenada en la misma se realizan consultas para obtener información en forma de reporte. A diferencia de las consultas tradicionales, con los generadores de reportes se puede definir el diseño y la forma en que la información será visualizada, es por ello que se compone por un diseñador de reportes y por un motor de generación de reportes, donde el primero se encarga de brindar las herramientas para diseñar la apariencia del informe y el segundo accede a la fuente de datos, obtiene los necesarios y los introduce en el diseño de la plantilla del reporte con la que luego se puede realizar ciertas operaciones como: imprimir, enviar por correo electrónico o guardar a un archivo (Churo, Rodrigo , Alpala, & Carla, 2010).

Tendencia

Se considera que una tendencia (del latín *tendens* o *tendentis*, que a su vez deriva del verbo *tendo* o *tendere*, que significa "extender", "tensar", "dirigirse a") es una corriente o una preferencia generalizada por la cual las personas se inclinan a preferir determinados fines o determinados medios por sobre otros**.** Las tendencias pueden consistir en un sentido general a la predisposición que tienen las cosas, los animales o las personas hacia una situación específica. Del mismo modo se considera que una tendencia es una fuerza física que permite que un cuerpo se incline o deslice hacia otro. Otro significado del término refiere a una manera específica de pensar, entender o razonar un aspecto particular del pensamiento (Pinto, Uribe Tirado, Gómez Díaz, & Cordón, 2011).

La tendencia estadística

Dentro del campo de la estadística, se considera que una muestra estadística, presenta una serie de casos, cada uno con un comportamiento particular. A partir del comportamiento de todos los casos registrados, se pueden realizar cálculos matemáticos para poder determinar la frecuencia de una cierta conducta, a las mediciones que se pueden obtener a través de éstos cálculos se los denomina tendencias estadísticas (Suárez Ibujes & Tapia Zambrano, 2013).

1.2. Herramientas para la generación de reportes

La generación de reportes es un componente esencial en todo sistema informático, donde se benefician usuarios directos e indirectos. En la actualidad existen en el mercado informático varias herramientas que brindan soporte para la generación de reportes. A continuación, se analizan algunas de ellas.

1.2.1. *Software* internacionales

Software Dream Report

El *software Dream Report* TM de Eurotherm es una solución integrada de informes para sistemas de automatización industrial. Está diseñada para ser una solución sencilla capaz de extraer datos de casi cualquier fuente y de generar informes automáticamente para cualquier usuario, en cualquier lugar. Basado en tecnologías modernas, el *software Dream Report* se adapta perfectamente a las aplicaciones de procesos continuos y por lotes. *Dream Report* permite también exportar la información a Microsoft *Excel* y, de forma automática, imprimirlo, guardarlo y publicarlo a la web. En el modo automático, la generación y distribución de los informes puede lanzarse con un evento o programarse previamente. En el

modo manual, una aplicación adicional (*Dynamic Report GeneratorTM*) permite programar la generación del informe o generar un informe bajo pedido en cualquier momento (Eurotherm, 2018).

IBM *Rational Quality Manager*

Rational Quality Manager es un *software* de licencia privativa sobre la gestión de calidad. El mismo se utiliza para la planificación integral de pruebas, el testeo manual y la integración con herramientas automatizadas de testeo. *Rational Quality Manager* ayuda a garantizar que los procesos de negocios cumplan con los estándares y las normas industriales, corporativas y departamentales. Durante el ciclo de vida de prueba, se dispone de las herramientas necesarias para obtener una medición minuto a minuto de la calidad del *software* y las métricas del proyecto. Se puede utilizar para producir registros confiables sobre los resultados de las pruebas y la historia del proyecto. El plan de prueba es un documento dinámico y vivo que evoluciona junto con el proyecto, es un contrato de gestión de calidad que describe claramente los objetivos del proyecto y los criterios de entrada y salida, mientras que también se rastrean los elementos prioritarios para realizar su validación, para la descripción gráfica del proyecto utiliza gráficos de barra, gráficos pastel y tablas (Kelly, 2010).

***Hewlett Packard Quality Center* (HP QC)**

Ofrece una plataforma de calidad del *software* completa, unificada y ampliable. Puede implementar su gestión de la calidad y establecer procesos coherentes y repetibles para gestionar todos los aspectos de la calidad del *software*, incluyendo gestión de requisitos, gestión de pruebas y los procesos de negocio para entornos y aplicaciones IT. HP QC incluye un módulo de panel integrado que centraliza los informes de calidad para que pueda tomar decisiones en tiempo real basadas en el estado de las aplicaciones en todos los proyectos e iniciativas de control de calidad permitiendo descargar los informes en formato *Word*. (opshub integration & Migration Solutions, 2014):

Redmine

Es una aplicación de *software* para la gerencia de proyectos, puede funcionar en diversas plataformas y bases de datos de distintos proveedores. Ahorra tiempo y esfuerzo y sirve para llevar una organización de toda la información de modo centralizado e independiente. Permite ahorrar comunicaciones por e-mail si bien el avance del proyecto y las tareas pueden ser notificadas por e-mail a la vez que quedan

almacenadas No es una herramienta de gestión de *software Testing perse*, sin embargo, se puede usar para organizar el ciclo de calidad de *software* como si fuera un proyecto (easyRedmine, 2018).

1.2.2. *Software* Nacional

Generador de Reportes Dinámicos PATDSI

Módulo de control y seguimiento: se destaca por un cuadro de mando integral que permite analizar tanto un portafolio de proyectos como proyectos independientes. Con facilidades para la identificación del camino crítico, la cadena crítica y un conjunto de indicadores que incluyen indicadores como el Índice de rendimiento de la programacion (IRP), índices de rendimiento de costos (IRC) y otros que garantizan una mirada completa al estado del proyecto respecto a: calidad, costos, tiempo, recursos humanos y contratos. Incluyen más de 60 reportes y permite la generación de otros nuevos aumentando la flexibilidad del sistema. La plataforma permite la generación de reportes desde dos plataformas: la PATDSI Generador de Reportes Dinámicos y desde el paquete Libre Office. Esta segunda posibilidad con la comodidad de la integración a un paquete ofimático multiplataforma más conocido por los clientes (Piñero, 2011).

Tabla 1 Resumen del análisis de los sistemas homólogos estudiados. Fuente: Elaboración Propia

Herramienta	**Formato de Salida**	**Licencia**	**Representación del Reporte**	***Open Source***
Software Dream Report	Pdf, *Excel*	Privativa	Representación textual, Tablas, Barras y pastel, Gráficos	NO
IBM *Rational Quality Manager*	*Excel*	Privativa	Tablas de pastel, tablas de barra y gráficos	NO
***Hewlett Packard Quality Center* (HP QC)**	*Word*	Privativa	Informes y tablas	NO

Redmine	Pdf	Pública	Gráficos Gantt, tablas de pastel, tabla de barra	SI
Generador de Reportes Dinámicos PATDSI	*Excel*, pdf, HTML	Pública	Tablas de pastel, tablas de barra	SI

El estudio de los sistemas informáticos existentes a nivel nacional e internacional permitió observar y analizar los diferentes formatos para la presentación de las salidas de información. Se pudo observar que estos módulos de reportes tienen características y funcionalidades similares, ya que fueron creados con el mismo propósito de ayudar a los usuarios a obtener reportes de información almacenada en una base de datos, además de poder exportarlos y ser impresos. Las diferencias radican en el tipo de licencia, el acceso a código fuente, el formato de salida de los reportes y la representación del reporte. Los sistemas analizados anteriormente no se pueden utilizar para resolver el problema planteado porque algunos de ellos tienen licencia privativa. Redmine es el utilizado actualmente por GESPRO y no se tuvo acceso a la modificación de los módulos del mismo. El generador de reportes dinámicos PATDSI realiza búsquedas repetitivas y procesamiento de la información para cada nivel de la estructura administrativa de la organización, que conlleva a la prolongación de la ejecución de la tarea. Conocer las funcionalidades que realizan los sistemas estudiados ayudó al mejor entendimiento del objeto de estudio, determinando que se pueden utilizar algunas funcionalidades de dichos sistemas como el formato de descarga el cual puede ser en pdf, *Excel*, *Word*, que la representación de los datos sea en diagramas, tablas de contenidos, gráficos Gantt, así como la variedad de colores para identificar el nivel en que se encuentran los proyectos.

1.3. Metodología

Se basa en un conjunto de procedimientos racionales utilizados para alcanzar el objetivo o la gama de objetivos que rige una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos. Con frecuencia puede definirse la metodología como el estudio o

elección de un método pertinente o adecuadamente aplicable a determinado objeto (Gómez & Roquet, 2012).

Scrum

Scrum es una metodología ágil enfocada a la gestión de proyectos. Sus principales características se pueden resumir en dos: el desarrollo de sprint o iteraciones y reuniones a lo largo del desarrollo. Las iteraciones en Scrum tienen una duración máxima de 30 días y el resultado de cada uno de ellas define un incremento del producto a desarrollar. La evolución del proyecto por la metodología se define a través de reuniones diarias donde el trabajo del día anterior es revisado por el equipo, previendo además la labor a realizar el día siguiente (Schwaber & Sutherland, 2013).

Programación Extrema (XP)

Es una metodología ágil de desarrollo de *software* que posee cuatro tareas fundamentales: planificación, diseño, desarrollo y pruebas. Esta metodología está basada en la simplicidad durante el desarrollo, la comunicación entre las partes implicadas (clientes y desarrolladores) y la retroalimentación para poder reutilizar el código desarrollado. En su concepción establece entregas frecuentes con posibilidad de refactorización continua, permitiendo mejorar el diseño cada vez que se añade una funcionalidad (Hiebaum, 2015).

Metodología de desarrollo de *software* Proceso Unificado Ágil (AUP)

El Proceso Unificado Ágil o *Agile Unified Process* (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (*test driven development* - TDD), Modelado Ágil, Gestión de Cambios Ágil y Refactorización de Base de Datos para mejorar la productividad. AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Para ello, se crean y mantienen listas identificando los riesgos desde etapas iníciales del proyecto. Especialmente relevante en este sentido es el desarrollo de prototipos ejecutables durante la base de elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos clave del producto y que determinan los riesgos técnicos (Vela & Reyes, 2017).

AUP-UCI

La metodología que se va a utilizar en el desarrollo del sistema es AUP-UCI ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de *software* tiene entre sus objetivos aumentar la calidad del *software* que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad.

En la metodología AUP-UCI existen cuatro escenarios para la disciplina de requisitos , para el desarrollo del sistema de generación de reporte para análisis de tendencia de calidad se trabajará en el escenario cuatro el cual se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una historia de usuario no debe poseer demasiada información.

Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de *software*, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Se redujo a 1 la cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11. Por todo lo anteriormente planteado se decide implementar la metodología AUP-UCI cumpliendo así con la estandarización del proceso de desarrollo de *software* (Rodríguez Sánchez, 2015).

1.4. Lenguajes y herramientas para el modelado de la solución

1.4.1. Lenguajes de programación (Lado del Cliente)

JavaScript

JavaScript se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web. Los navegadores modernos interpretan el código *JavaScript* integrado en las páginas web (Guchat, 2012). Para el desarrollo del sistema se emplea el lenguaje *JavaScript* en su versión 5.0 ya que al estar alojado en el ordenador del

usuario los efectos son rápidos y dinámicos. Por ser un lenguaje de programación permite toda la potencia de la programación como uso de variables, condicionales, bucles, etc. También podemos citar algún inconveniente: por ejemplo, si el usuario tiene desactivado *JavaScript* en su navegador, no se mostrarán los efectos.

CSS

CSS en su versión 3.0 son las siglas de *Cascading Style Sheets* - Hojas de Estilo en Cascada - que es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación. El lenguaje CSS se basa en una serie de reglas que rigen el estilo de los elementos en los documentos estructurados y que forman la sintaxis de las hojas de estilo. Cada regla consiste en un selector y una declaración, esta última va entre corchetes y consiste en una propiedad o atributo y un valor separados por dos puntos (Luján Mora, 2013).

En el desarrollo del sistema se utilizará CSS ya que guarda el estilo completamente por separado del contenido siendo posible, por ejemplo, almacenar todos los estilos de presentación para una web de 10.000 páginas en un sólo archivo de CSS. Permite un mejor control en la presentación de un sitio web que los elementos de HTML, agilizando su actualización. Aumenta la accesibilidad de los usuarios gracias a que pueden especificar su propia hoja de estilo, permitiéndoles modificar el formato de un sitio web según sus necesidades, de manera que personas con deficiencias visuales puedan configurar su propia hoja de estilo para aumentar el tamaño del texto. El ahorro global en el ancho de banda es notable, ya que la hoja de estilo se almacena en cache después de la primera solicitud y se puede volver a usar para cada página del sitio, no se tiene que descargar con cada página *web*.

Bootstrap

Es una herramienta que permite crear interfaces de usuario transparentes, además, *bootstrap* dispone de numerosas herramientas necesarias para diseñar cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías (Maldonado & Mauricio, 2017).

Se hará uso de *bootstrap* en su versión 4.0 por permitir crear interfaces que sean adaptables a cualquier navegador, incluido internet explorer usando *HTML Shim*, de igual manera a equipos de escritorio, *tablets* y móviles a distintas escalas, permitiendo ser usado de forma muy flexible para desarrollo *web* con excelentes resultados. Permite agregar imágenes responsive, es decir, con solo insertar la imagen cuya

clase sea *"img-responsive"* las imágenes se acoplarán al tamaño. Se integra perfectamente con las principales librerías *javascript*, por ejemplo, *jquery*.

HTML

Es un lenguaje de marcado de hipertexto especializado en crear páginas *web*, para ello se utilizan unas series de etiquetas. Todo documento creado con html tiene una estructura claramente definida. Siempre se comienza con la etiqueta <html>, que es la que comprende a toda la página *web*. Tiene dos secciones básicas bien diferenciadas: la cabecera y el cuerpo que se corresponde con las etiquetas <head> y <body> respectivamente (Hugo & Rmos, 2015).

Se utilizará HTML en su versión 5 por su compatibilidad con la mayoría de los navegadores, multiplataforma, facilidad de aprendizaje (debido al reducido número de etiquetas que usa), factibilidad de programación para dispositivos móviles, computación en la nube y funcionamiento en red, que lo hacen conveniente para el proyecto que se desarrolla.

1.4.2. Lenguaje de programación (Lado del servidor)

PHP

PHP en su versión 7.0 *Hypertext Pre-processor*, fue desarrollado puntualmente para diseñar páginas *web* dinámicas programando scripts del lado del servidor. El lenguaje PHP siempre va incrustado dentro del HTML y generalmente se le relaciona con el uso de servidores linux. PHP se caracteriza por ser un lenguaje gratuito y multiplataforma. Además de su posibilidad de acceso a muchos tipos de bases de datos, también es importante destacar su capacidad de crear páginas dinámicas, así como la posibilidad de separar el diseño del contenido de una web (Beati, 2015).

Se empleará PHP para el desarrollo del sistema en su versión 7.0 porque se ejecuta en cualquier plataforma utilizando el mismo código fuente. La sintaxis de PHP es similar a la del C, por esto cualquiera con experiencia en lenguajes del estilo C podrá entender rápidamente PHP. Es completamente expandible y modificable. Está compuesto de un sistema principal, un conjunto de módulos y una variedad de extensiones de código. Muchas interfaces distintas para cada tipo de servidor. PHP actualmente se puede ejecutar bajo *Apache*, IIS, *AOLServer*, *Roxen* yTHTTPD. Otra alternativa es configurarlo como un módulo. Permite la interacción con gran cantidad de motores de bases de datos tales como MySQL, MS SQL,

Oracle, Informix, *PostgreSQL*, etc. PHP es de código abierto esto significa que no depende de ninguna compañía comercial y que no requiere de licencias.

1.4.3. Lenguaje de modelado

UML

El Lenguaje de Modelamiento Unificado (UML - *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de *software*. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reusables. Es un lenguaje de modelado formado por símbolos y es utilizado por muchas metodologías (Romero, 2016). Para el desarrollo de la presente investigación se utilizará UML en su versión 2.1 por las funcionalidades anteriormente expuestas.

1.4.4. Herramienta de modelado

Visual paradigm

Es una herramienta *CASE* con varias opciones para el modelado con diagramas UML2 y también es compatible con los requisitos de SysML esquemas y diagramas entidad-relación. La herramienta tiene un buen ambiente de trabajo, lo que facilita la visualización y manipulación del proyecto de modelado. Es una herramienta de trabajo y también es compatible con cambios específicos en el código fuente de algunos lenguajes de programación como C ++ y Java (Sánchez & Sánchez, 2018).

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML, ideal para ingenieros de *software*, analistas de sistemas y arquitectos de sistemas, que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Para la presente investigación, se utiliza esta herramienta en su versión 8.0.

1.4.5. Herramienta de desarrollo

NetBeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. *NetBeans* IDE es un producto libre y gratuito sin restricciones de uso, es de código abierto de gran éxito con una gran base

de usuarios, una comunidad en constante crecimiento y con cerca de 100 socios en todo el mundo. La plataforma *NetBeans* permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de *software* llamados módulos. Un módulo es un archivo *Java* que contiene clases de *java* escritas para interactuar con las APIs de *NetBeans* y un archivo especial (*manifest file*) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma *NetBeans* pueden ser extendidas fácilmente por otros desarrolladores de *software* (Ayala Narea, Sangoquiza, & Estefanía, 2017). Por lo anteriormente plateado se decide utilizar *NetBeans* en su versión 8.1.

Symfony

Es un *framework* diseñado para optimizar el desarrollo de las aplicaciones *web* basado en el patrón Modelo Vista Controlador. El cual aporta una serie de herramientas y clases ya programadas, encaminadas a reducir el tiempo de desarrollo de una aplicación *web*. Otra ventaja es que automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación, por lo tanto, reduce el tiempo de desarrollo de cualquier sitio *web*, que en definitiva es lo persiguido en el campo del desarrollo de aplicaciones (Valle Dávila, 2017).

Se empleará *Symfony* en su versión 3.4 para el desarrollo del sistema por ser fácil de instalar y configurar en la mayoría de plataformas..*Symfony* ha sido desarrollado con la idea de ofrecer aplicaciones de alto rendimiento, es más rápido que otros frameworks usando la mitad de la memoria**.** Es flexible ya que se adapta a casi cualquier necesidad, cuenta con bases de datos del motor independiente, permitiendo instalar únicamente las piezas requeridas para el proyecto en vez de todo el *framework*. Es un sistema rápido y que consume poca memoria, *Symfony* ha sido desarrollado con la idea de ofrecer aplicaciones de alto rendimiento.

1.4.6. Servidor *web*

Se basa en el procesamiento de una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales con el cliente y generando una respuesta en cualquier lenguaje. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa (Villada, 2015).

Servidor web Apache

Apache es un poderoso servidor web, cuyo nombre proviene de la frase inglesa "*a patchy server*" y es completamente libre, ya que es un *software Open Source* y con licencia GPL. Una de las ventajas más grandes de Apache, es que es un servidor web multiplataforma, es decir, puede trabajar con diferentes sistemas operativos y mantener su excelente rendimiento (Passadore, 2017).

Entre las principales características de Apache, se encuentran las siguientes:

1. Soporte de seguridad SSL y TLS.
2. Puede realizar autentificación de datos utilizando SGDB.
3. Puede dar soporte a diferentes lenguajes, como Perl, PHP, *Python*.

Se usará Apache en su versión 2.4 porque realiza servicio a páginas web, ya sean estáticas o dinámicas. Este servidor se integra a la perfección con otras aplicaciones, creando el famoso paquete *XAMP* con Perl, *Python*, *MySQL* y PHP, junto a cualquier sistema operativo, que por lo general es *Linux*, *Windows* o *Mac OS*.

1.4.7. Servidor de Bases de Datos

MySQL

MySQL Database Server es un sistema gestor de bases de datos de código fuente abierto. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar. La extensiva reutilización del código dentro del *software* y una aproximación minimalística para producir características funcionalmente ricas, ha dado lugar a un sistema de administración de base de datos incomparable en velocidad, compactación, estabilidad y facilidad de despliegue (Arias, 2017). Por lo anteriormente planteado se utilizará MySQL en su versión 5.7.24.

1.4.8. Herramientas para pruebas de *software*

Apache JMeter

Apache JMeter fue inicialmente diseñado para pruebas de estrés en aplicaciones web, hoy en día, su arquitectura ha evolucionado no sólo para llevar acabo pruebas en componentes habilitados en *Internet* (HTTP), sino además en bases de datos , programas en perl ,

requisiciones FTP y prácticamente cualquier otro medio. Además, posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de requisiciones que permiten diagnosticar el comportamiento de una aplicación en condiciones de producción. En este sentido, simula todas las funcionalidades de un Navegador ("*Browser*"), o de cualquier otro cliente, siendo capaz de manipular resultados en determinada requisición y reutilizarlos para ser empleados en una nueva secuencia. (Patel, B., Parikh, J., & Shah, R., 2014). Por lo anteriormente planteado se decide utilizar *Apache JMeter* en su versión 3.1.

Acunetix

Acunetix Web Vulnerability Scanner es una herramienta que será capaz de escanear sitios web en busca de posibles fallos de seguridad que puedan poner en peligro la integridad de la página publicada en Internet. Esta aplicación ejecuta una serie de pruebas, totalmente configurables por el usuario, para identificar las vulnerabilidades tanto en la programación de la página como en la configuración del servidor (Daud, N. I., Bakar, K. A. A., & Hasan, M. S. M., 2014). Por lo anteriormente planteado se decide utilizar Acunetix en su versión 8.0.

1.5. Conclusiones Parciales

En este capítulo se abordaron los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, en tal sentido se puede arribar a las siguientes conclusiones:

1. La definición de los principales conceptos relacionados con el desarrollo del sistema de gestión de reportes para análisis de tendencia de calidad y las relaciones entre estos, permitió alcanzar una mayor comprensión de la propuesta de solución.
2. Con el estudio de los sistemas basados en la generación de reportes se adquirió una serie de conocimientos que son útiles para el desarrollo de la propuesta de solución de la presente investigación, aunque estos no reúnen los requisitos para ser incluidos completamente en la misma.
3. El análisis de las tecnologías seleccionadas, permitió un acercamiento a los elementos del marco de trabajo y sus características, para hacer más fácil su uso durante el desarrollo de la propuesta de solución planteada.

CAPÍTULO 2. DESCRIPCIÓN DEL SISTEMA DE GESTIÓN DE REPORTES PARA ANÁLISIS DE TENDENCIAS DE CALIDAD

Introducción

Una de las prioridades cuando se desea desarrollar un *software*, es establecer un entendimiento entre el cliente y el equipo de trabajo en relación con los objetivos a lograr (Molina, et al., 2014), realizando un correcto análisis y diseño de dicho sistema. El objetivo de este capítulo es presentar los resultados que se obtuvieron una vez cumplidas las fases de análisis y diseño que propone la metodología AUP-UCI. Se detallan las características del sistema de gestión de reportes para análisis de tendencia de calidad, se especifican los requisitos funcionales y no funcionales del mismo y se presentan los artefactos generados para dar solución al problema planteado en la investigación.

2.1. Marco conceptual

El marco conceptual es una representación gráfica que suele presentarse en forma de árbol o de esquema y que aporta una visión de conjunto de un estudio determinado. En otras palabras, es una guía de trabajo que permite comprender los apartados de una investigación y cómo se relacionan entre sí (Daros, 2017).

Para lograr un mejor entendimiento de los procesos que requieren informatización, se realizó un marco conceptual (ver Ilustración 1), con un total de tres clases y dos relaciones, el cual recoge y describe los conceptos más importantes dentro del contexto del sistema de gestión de reportes, así como las relaciones entre ellos.

Como parte del marco conceptual se aprecia el **asesor de calidad,** como actor principal del proceso. El asesor recopila la información manual la cual puede ser insertada en varios ***Excel***, un *Excel* ofrece varios diagramas que corresponden a la información anteriormente insertada, luego con dichos diagramas se genera el **reporte** para análisis de tendencia en el cual se presenta la calidad de los proyectos.

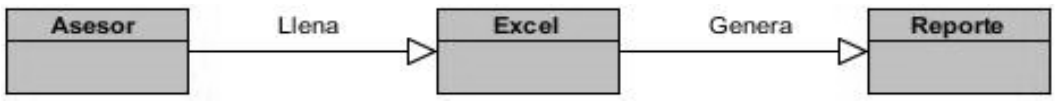


Ilustración 1: Marco conceptual. Fuente: Elaboración Propia

2.2. Técnicas de obtención de requisitos

Las técnicas de obtención de requerimientos son aquellas que permiten comprender el dominio del sistema, buscar y recolectar información para definir sus límites y restricciones, e identificar a las personas interesadas en el sistema. El resultado brindará una colección y clasificación de los requerimientos del sistema, mediante la participación de los clientes y usuarios (Fortis Urbano, 2015).

Las técnicas aplicadas en esta investigación son:

1. Entrevistas: Se llevó a cabo una conversación con el cliente con el objetivo de entender el dominio del problema y sus necesidades. Esta se basó en un formato de preguntas y respuestas, buscando obtener criterios sobre el proceso de generación de reportes actual con vista a mejorarlo.
2. Observación del flujo actual de los procesos: Esta técnica ayuda a la obtención de información de los procesos actuales y los principales aspectos en los que se evalúa cada proyecto. La aplicación de esta técnica aportó grandes beneficios para la elaboración de los requisitos funcionales que conformaron la propuesta de solución.

2.3. Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe realizar, es decir, define qué es lo que el sistema debe hacer, así como las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas, para producir salidas (Ovalle, Salazar, & Duque, 2014). A continuación, se muestra en la tabla 2 los requisitos funcionales obtenidos, de donde se deriva las funcionalidades que contendrá el sistema propuesto.

Tabla 2 Requisitos funcionales del sistema. Fuente: Elaboración Propia.

Requisitos Funcionales		**Complejidad**
RF: 1	Autenticar usuario.	Media
RF: 2-5	Gestionar reporte (editar, eliminar, mostrar, generar).	Alta

RF: 6-9	Gestionar el listado de proyectos (insertar, eliminar, mostrar, editar).	Media
RF: 10-13	Gestionar el listado de indicadores (insertar, eliminar, mostrar, editar)	Alta
RF: 14-17	Gestionar la cantidad actividades (insertar, editar, eliminar, mostrar).	Media
RF: 18-21	Gestionar el tiempo en que se cumplieron todas las actividades (insertar, editar, eliminar, mostrar).	Media
RF: 22-25	Gestionar el estado del plan de evaluaciones (insertar, editar, eliminar, mostrar).	Alta
RF: 26-29	Gestionar la tendencia de las no conformidades (insertar, editar, eliminar, mostrar).	Alta
RF: 30-33	Gestionar la adherencia a procesos y productos de trabajo (insertar, editar, eliminar, mostrar).	Alta
RF: 34-37	Gestionar el análisis de las causas (insertar, editar, eliminar, mostrar).	Media
RF: 38-41	Gestionar el estado de las no conformidades de las evaluaciones (insertar, editar, eliminar, mostrar).	Alta
RF: 42-45	Gestionar la apreciación de la utilidad de los procesos y productos de trabajo (insertar, eliminar, editar, mostrar).	Alta
RF: 46-49	Gestionar el estado de las expectativas de la organización(insertar, editar, eliminar, mostrar).	Baja
RF: 50-53	Gestionar el estado del plan de liberaciones (insertar,	Baja

	editar, eliminar, mostrar).	
RF: 54-57	Gestionar las no conformidades por tipo de error según su clasificación(insertar, editar, eliminar, mostrar).	Media
RF: 58-61	Gestionar las iteraciones según artefacto a liberar (insertar, editar, eliminar, mostrar).	Alta
RF: 62-65	Gestionar las no conformidades por iteraciones por tipo de error según su clasificación(insertar, editar, eliminar, mostrar).	Media
RF: 66-69	Gestionar el estado de las no conformidades por iteraciones (insertar, editar, eliminar, mostrar).	Media
RF: 70	Exportar el reporte a pdf.	Media

2.4. Requisitos no funcionales

Los requisitos no funcionales son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del mismo (Martínez & Martínez, 2014). A continuación, se muestra en la tabla 3 los requisitos no funcionales obtenidos, de donde se derivan las funcionalidades que contendrá el sistema propuesto.

Tabla 3 Requisitos no funcionales del sistema. Fuente: Elaboración Propia.

Requisitos no funcionales	
Usabilidad	El sistema debe presentar una interfaz que permita la fácil interacción con el usuario y llegar de manera rápida y efectiva a la información buscada. Debe ser una interfaz agradable que posibilite a los usuarios sin experiencia

	una rápida adaptación.
Seguridad	La seguridad de la base de datos es a nivel de roles, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos. La seguridad del sistema se garantiza mediante roles, mediante usuarios y contraseña definidos en el sistema.
Confiabilidad	El sistema debe ser tolerante a fallos y mostrar solo la información necesaria para orientar al usuario.
Portabilidad	La propuesta de desarrollo debe ser capaz de ejecutarse en los navegadores empleados en la UCI. Su interfaz debe ser adaptable a cualquier dispositivo, ya sea una computadora, tableta o teléfono celular. Debe disponer de un ordenador con sistema operativo GNU/Linux y cuyas propiedades mínimas sean un procesador Intel Dual Core con velocidad de 2.10 GHz, 2 GB de memoria RAM y 500 MB de disco duro.
Eficiencia	El sistema debe permitir que los usuarios interactúen con él de manera concurrente. El tiempo de demora de una petición al servidor debe ser menor de cinco (5) segundos aproximadamente.
Funcionalidad	La aplicación debe gestionar y requerir información de usuarios para su uso. La información manejada por el sistema debe estar protegida de accesos no autorizados. Ante los errores que puedan ocasionarse en el sistemas no se deben mostrar detalles de información que puedan comprometer su seguridad e integridad.
Mantenibilidad	Se permite realizar modificaciones posteriores para adaptar mejoras al sistema o en caso que cambien las necesidades de los clientes.

	El *software* debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

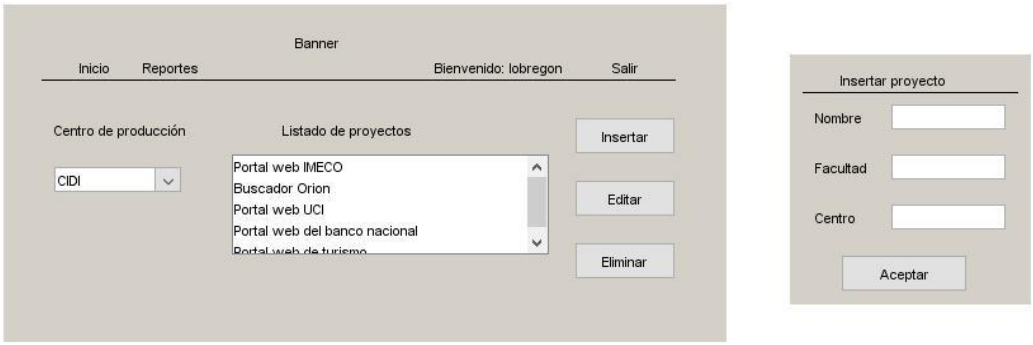
2.5. Historia de Usuario

La historia de usuario (HU) sirve para registrar los requerimientos de los clientes y son utilizadas para crear las pruebas de aceptación y realizar la estimación de cada una de las iteraciones durante la fase de planificación. Describen brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales en base a lo que se estima necesario para el sistema. El tratamiento de las HU es dinámico y flexible. Cada una es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas, así se asume en (Carrizo & Rojas, 2016). Para la presente investigación se generaron un total de dieciocho HU, a continuación, solo se muestra una de ellas en la tabla 4, perteneciente a los RF 4, especificado en el epígrafe 2.3.

Tabla 4 HU_3 Gestionar listado de proyectos. Fuente: Elaboración Propia

Historia de usuario	
Número: 7-11.	**Usuario:** asesor de calidad.
Nombre historia: Gestionar el listado de proyectos (insertar, eliminar, mostrar, editar).	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alta.
Tiempo estimado: 120 horas.	**Iteración asignada:** 9.
Programador responsable: Lissandra Acosta Obregón	
Descripción: El usuario puede visualizar el listado de proyectos.	
Observaciones: Se debe mostrar el nombre del proyecto. Se puede editar, eliminar o insertar un proyecto.	

Prototipo de interfaz:



2.6. Validación de requisitos

La validación es la actividad de la ingeniería de requisitos que permite demostrar que los requerimientos definidos en el sistema son los que realmente el cliente necesita y quiere. La validación verifica que los requisitos estén completos y sean consistentes y además que definan lo que el usuario espera del producto final (Terstine, 2015).

Existen diversas técnicas para la validación de requisitos entre los más comunes se encuentran la revisión de requisitos, la realización de prototipos de interfaz y la generación de casos de prueba. Para realizar la validación de requisitos de la presente investigación se utilizó la realización de prototipos.

2.7. Descripción de la arquitectura

Para seleccionar la arquitectura a utilizar por el sistema se tuvo en cuenta que la misma sea la apropiada para las aplicaciones web, así como que sea adaptable a las tecnologías proporcionadas por la plataforma PHP. Se consideró que el estilo arquitectónico que cumple con las condiciones planteadas es el cliente servidor haciendo uso del patrón modelo vista controlador. También se trató de lograr que todo el desarrollo esté soportado sobre *software* libre, para que la herramienta no herede ningún tipo de limitación en lo relativo a la soberanía tecnológica.

2.8. Patrón Arquitectónico

Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de *software*, que consta de subsistemas, sus responsabilidades e interrelaciones. No es una arquitectura como tal, sino un concepto que captura elementos esenciales de una arquitectura de *software*. El patrón de desarrollo Modelo-Vista-Controlador (MVC), separa los datos de una

aplicación, la interfaz de usuario y la lógica de negocio en tres capas conceptuales (Camarena, Trueba, Martínez, & López, 2016).

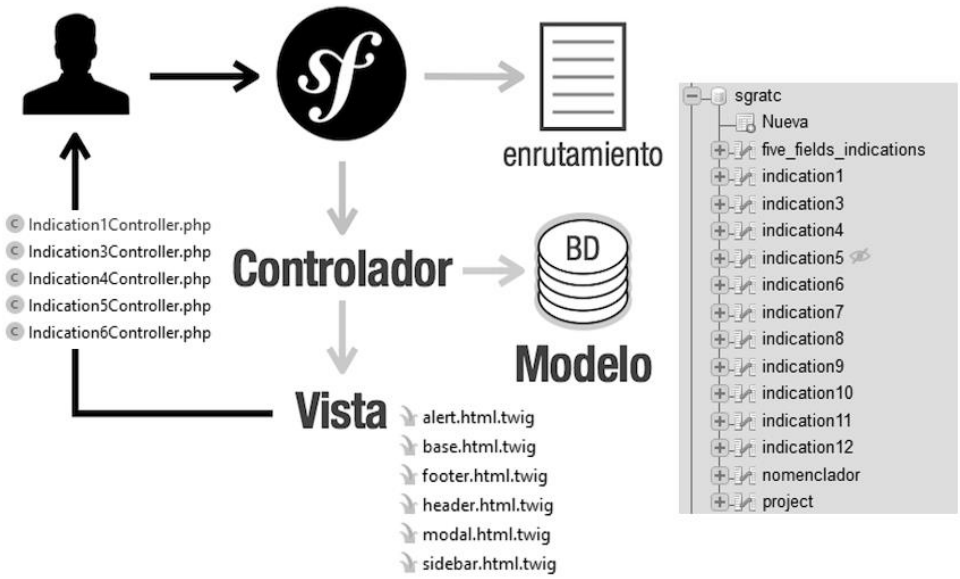


Ilustración 2 Patrón Modelo Vista Controlador (MVC). Fuente: Elaboración Propia.

Modelo: representa la información con la que trabaja la aplicación, es decir su lógica de negocio. Gestiona los datos solicitados por el controlador. Esto se evidencia cuando se necesita mostrar la fuente de datos.

Vista: convierte el modelo en una página web que facilita al usuario interactuar con ella. Contiene los elementos asociados a la presentación de los datos y la información visual de los elementos que conforman el reporte, encontrándose el código asociado a las interfaces, que se encargan de visualizar la información que el controlador devuelve, dicha información se encuentra en los *twigs*.

Controlador: es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista. La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista), lo que permite un mantenimiento más sencillo de las aplicaciones. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para las peticiones (HTTP, consola de comandos, *email*).

En el sistema de gestión de reportes para análisis de tendencia de calidad el patrón Modelo Vista Controlador (MVC) se aplica de la siguiente manera (Patiño Mora, 2016):

1. El cliente se conecta con el sistema y este consulta la tabla de enrutamiento, que es donde se encuentran todas las rutas de los controladores que tiene el sistema.
2. A través de esta ruta se conecta con el controlador.
3. El controlador realiza la petición al modelo y este a su vez se conecta con la base de datos para obtener la información.
4. Esta información es mostrada al usuario mediante la vista.

2.9. Modelado del diseño

Patrones de diseño

Un patrón es un esquema o micro-arquitectura que supone una solución a problemas, cada patrón es adecuado para ser adaptado a un cierto tipo de probleama. Para que una solución sea considerada un patrón debe poseer ciertas características, una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Cosmin, 2016).

2.9.1. Patrones Generales de *Software* para la Asignación de Responsabilidades (GRASP)

Según Visconti (2012) los patrones GRASP están basados en la asignación de responsabilidades a objetos. Es una buena práctica para el desarrollo eficaz de la Programación Orientada a Objetos (POO).

Experto: es un patrón que se usa para asignar responsabilidades; es un principio básico que se utiliza en el diseño orientado a objetos. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento (Visconti, M., & Astudillo, H., 2012).

En el sistema, este patrón se utiliza, por ejemplo, para mostrar el análisis de las causas de cada proyecto. La clase entidad *Indication3*, es la que tiene la información necesaria para brindar esta información, por tanto, es la clase experta en información.

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar

un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (Visconti, M., & Astudillo, H., 2012).

Este patrón se puede evidenciar en la clase *Indication1Controller e*specíficamente en la función *manageAction* cuando se crea la instancia de la clase *nomencladorService* a partir de los datos de usuario contenidos en ella.

Controlador: Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. El mismo asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente el "sistema" global (Visconti, M., & Astudillo, H., 2012).

El empleo de este patrón se evidencia cuando el formulario de autenticación envía los datos a la clase *SecurityController* y esta se encarga de enviarlos a la clase usuario para realizar el procesamiento de autentificación a partir de la base de datos.

Bajo acoplamiento: Es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases (Visconti, M., & Astudillo, H., 2012).

La utilización de este patrón se evidencia en la poca dependencia entre las clases del sistema, de tal forma que si ocurrieran cambios en alguna de las clases no se afectan otras. Por ejemplo *SecurityController* depende de las clases *SoapAuthenticador*.php y *WebserviceUserProvider*.php para la realización de la autenticación ya que esta cuenta de dos partes, autenticación por el servicio soap y la existencia del usuario en la lista de usuarios permitidos.

Alta cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Las mismas poseen un número relativamente pequeño de responsabilidades, definiendo así que cada clase realice solo las funcionalidades para las cuales fueron creadas, generando un bajo acoplamiento y fomentando la reutilización (Visconti, M., & Astudillo, H., 2012).

Esto se evidencia en la clase *WebserviceUser.*php perteneciente al modelo user la cual cuenta con la información estrechamente relacionada con los usuarios del sistema.

2.9.2. Patrones GoF

Los patrones GoF (*Gang of Four*, en español Pandilla de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Guerrero, Suárez, & Gutiérrez, 2013). Los patrones GOF utilizados fueron:

***Decorator* (Decorador):** responde a la necesidad de añadir dinámicamente funcionalidad a un objeto. Esto nos permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera (Tedeschi, N., Creacionales, P., Estructurales, P.,, 2015). Se evidencia en el sistema a través de los archivos alert.html. twig, header.html. twig, footer.html. twig que contienen diseño y funcionalidades comunes para todo el sistema.

Fachada: la controladora *Indication1Controller* representa el cliente a la cual la modelo indication1 le encapsula un conjunto de complejidades del negocio que la controladora no conoce, constituyendo la modelo una fachada de la controladora.

2.9.3. Diagrama de clases del diseño

Los diagramas de clase (DC) pueden usarse cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases. Con el uso de estereotipos *web* se describe gráficamente las especificaciones del modelo, la vista y la plantilla de las historias de usuario descritas en el epígrafe 2.3. Estas representaciones contienen información acerca de las clases, asociaciones, atributos, métodos y dependencias (Quevedo & Suárez, 2015).

Para la presente investigación se generaron un total de diecinueve DC, a continuación, se muestra en la Ilustración 3 uno de ellos, perteneciente a la historia de usuario número cuatro, especificado en el epígrafe 2.3.

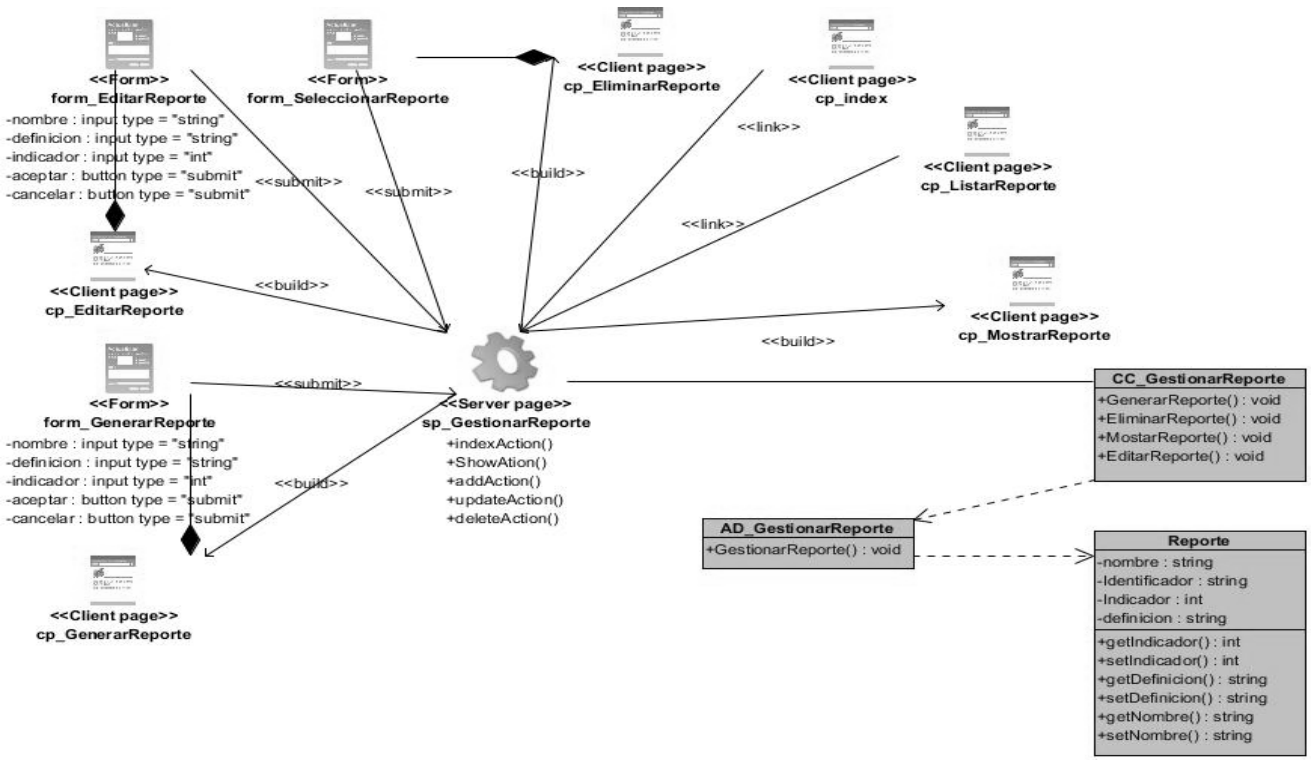


Ilustración 3 Diagrama de clases del diseño de gestionar reporte. Fuente: Elaboración Propia.

2.10. Diagrama de secuencia

Los diagramas de secuencia (DS) en el UML se usan principalmente para modelar las interacciones entre los actores y los objetos en un sistema, así como las interacciones entre los objetos en sí (Mazalu, 2013). Para la presente investigación se generaron un total de ocho DS, relacionados con los requisitos de prioridad alta especificados en el sub-epígrafe 2.3.1. A continuación, se muestra en la Ilustración 4 uno de ellos, perteneciente al RF 2-6, especificados en el sub-epígrafe 2.3.1.

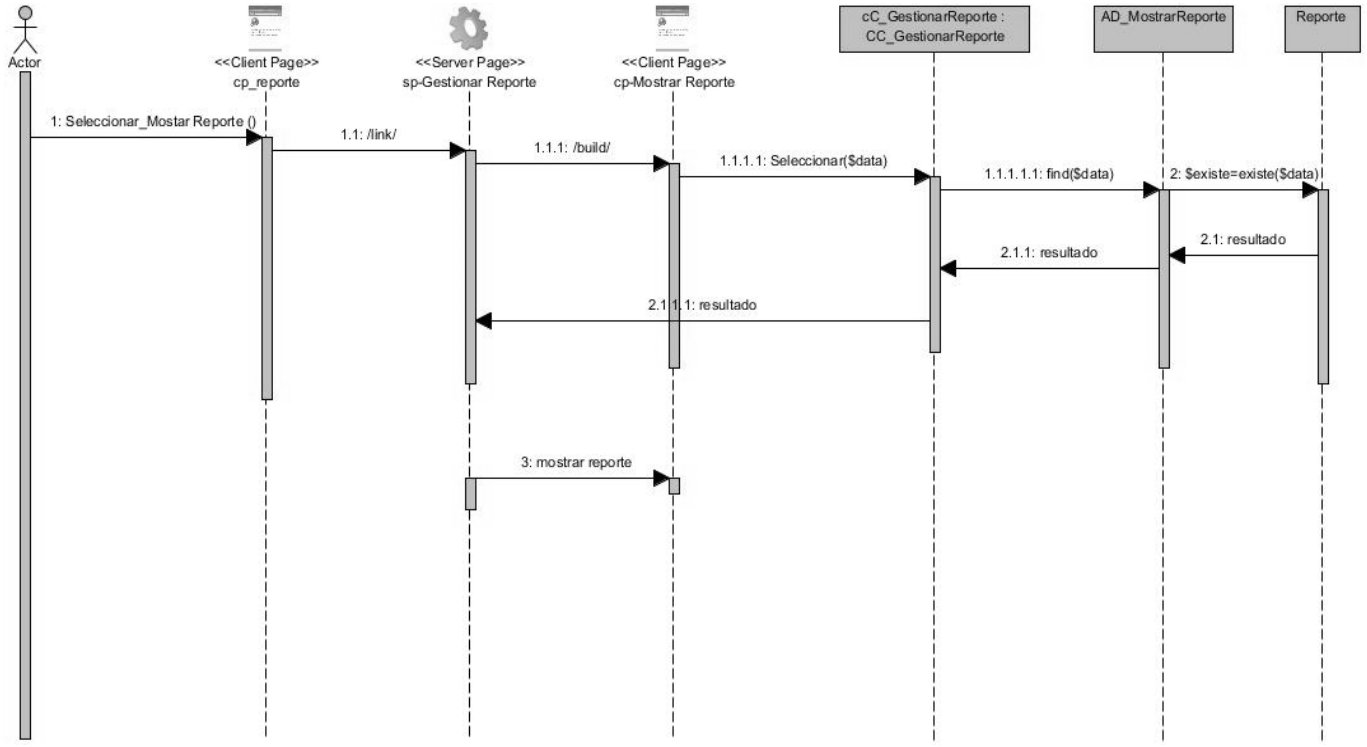


Ilustración 4 Diagrama de secuencia de mostrar reporte. Fuente: Elaboración Propia.

2.11. Modelo de base de datos

Este modelo está compuesto por las entidades (tablas) de la base de datos de MySQL que garantiza la persistencia y manejo de los datos desde las clases controladoras del sistema, que fueron representadas en los epígrafes anteriores. El modelo de datos generado, como se observa en la Ilustración 5, está compuesto por trece relaciones y catorce tablas, las cuales se explican a continuación, para un mejor entendimiento del modelo (de arriba abajo y de izquierda a derecha).

Indication1: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 1.

Indication3: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 3.

Indication4: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 4.

Indication5: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 5.

Indication6: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 6.

Indication7: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 7.

Indication8: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 8.

Indication9: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 9.

Indication10: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 10.

Indication11: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 11.

Indication12: almacena el resultado de lo anteriormente insertado en el formulario perteneciente a la indicación 12.

Five_fields_indications: almacena algunos de los atributos de los que heredaran la tabla indication1 e indication3

Nomencaldor: Almacena los nomencladores y la descripción de las actividades por las que será evaluado el proyecto.

Project: almacena los datos pertinentes a cada proyecto a evaluar.

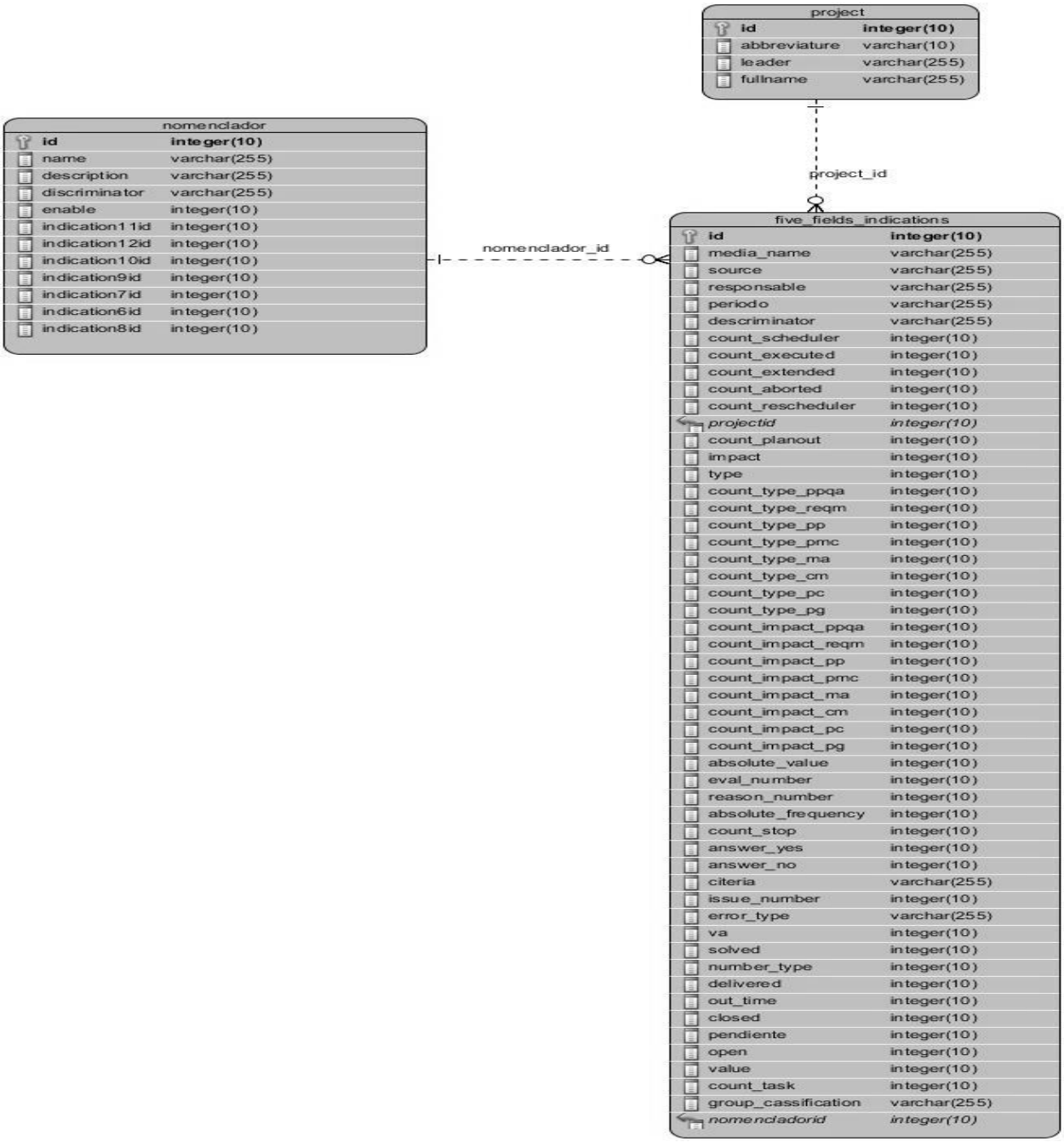


Ilustración 5 Modelo de datos. Fuente: Elaboración Propia.

2.12. Modelo de despliegue

Un modelo de despliegue consiste en una representación estructural de la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del *software* en los destinos de despliegue;

definiendo a los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo (Campoverde, Hernández, & Mazón, 2015).

A continuación, se muestra el diagrama de despliegue en la Ilustración 7 propuesto para el sistema de gestión de reportes para análisis de tendencia de calidad. El mismo, muestra la disposición física de los nodos que componen el sistema y el reparto de los componentes en dichos nodos.

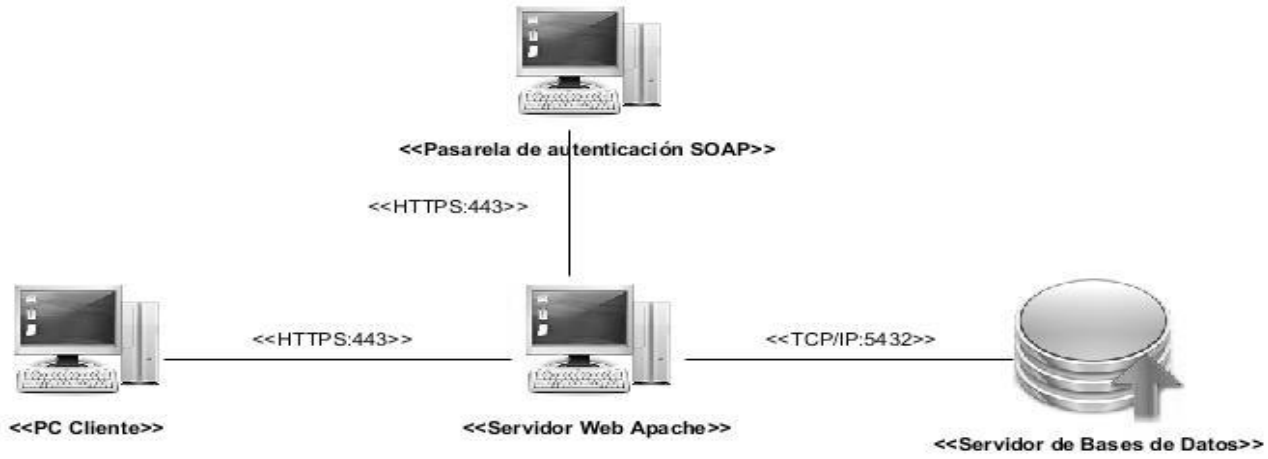


Ilustración 6 Diagrama de despliegue. Fuente: Elaboración Propia.

Este diagrama se considera importante para lograr un despliegue exitoso de la aplicación. En él, se definen las estaciones de trabajo (Dispositivo Cliente: computadora personal, tableta, teléfono celular) que el usuario utilizará para conectarse, vía HTTPS, con el servidor de aplicaciones web (Apache) en el que se encuentra alojado el sistema. Este servidor debe disponer de un ordenador con sistema operativo GNU/Linux y cuyas propiedades mínimas sean un procesador *Intel Dual Core* con velocidad de 2.10 GHz, 2 GB de memoria RAM y 500 MB de disco duro.

2.13. Conclusiones parciales

Después de realizado el análisis y diseño de la propuesta de solución y haber generado los diferentes artefactos que dispone la metodología AUP-UCI, se puede concluir lo siguiente:

1. El análisis del marco conceptual, permitió identificar los principales requisitos funcionales y no funcionales del sistema de gestión de reporte para análisis de tendencia de calidad, los cuales fueron reunidos y categorizados por historias de usuarios.
2. La identificación de los patrones de diseño y el estilo arquitectónico del sistema evidenció que la solución propuesta cuenta con un alto grado de resistencia ante posibles modificaciones.

3. El diseño de los diagramas de clases y de secuencia, permitió una mejor visión en cuanto a estructura física y lógica del sistema de gestión de reportes para análisis de tendencia de calidad.
4. La generación de los elementos requeridos por el modelo de desarrollo, evidencian la solución propuesta, lo cual contituye una ayuda para su posterior mantenimiento (actualización o adición de funcionalidades).

CAPÍTULO 3. VALIDACIÓN DEL SISTEMA DE GESTIÓN DE REPORTES PARA ANÁLISIS DE TENDENCIAS DE CALIDAD

Introducción

Antes de escribir una línea de código, es fundamental haber comprendido bien el problema que se pretende resolver y haber aplicado principios básicos de diseño que permitan construir un sistema de calidad. La misma se mide mediante modelos y métricas aplicados al producto y al proceso de creación, diseño, desarrollo y mantenimiento de *software*, por lo cual es de suma importancia realizar un correcto proceso de verificación y validación de *software*. Según Puello (2013), el objetivo del proceso de verificación y validación es establecer la seguridad de que el sistema software está "hecho para un propósito". Esto significa que el sistema debe ser lo suficientemente bueno para su uso pretendido. El nivel de confianza requerido depende del propósito del sistema, las expectativas de los usuarios del sistema y el entorno de mercado actual del sistema.

3.1. Diagrama de componentes

El diagrama de componentes muestra los componentes de un sistema de *software* conectados por las relaciones de dependencias lógicas entre cada uno de ellos. Provee una vista arquitectónica de alto nivel del sistema, ayudando a los desarrolladores a visualizar el camino de la implementación. Cada componente representa una unidad del código (fuente, binario o ejecutable), que permite mostrar las dependencias en tiempo de compilación y ejecución. La realización del diagrama posibilita tomar decisiones respecto a las tareas de implementación y los requisitos (Aguilar, J., & Dos Santos, R. , 2015).

A continuación, la Ilustración 8 muestra el diagrama de componentes del sistema de gestión de reportes para análisis y tendencias de calidad; cuya organización se encuentra acorde con el patrón arquitectónico MVC propuesto por el framework *symfony* y descrita en el capítulo anterior de este trabajo.

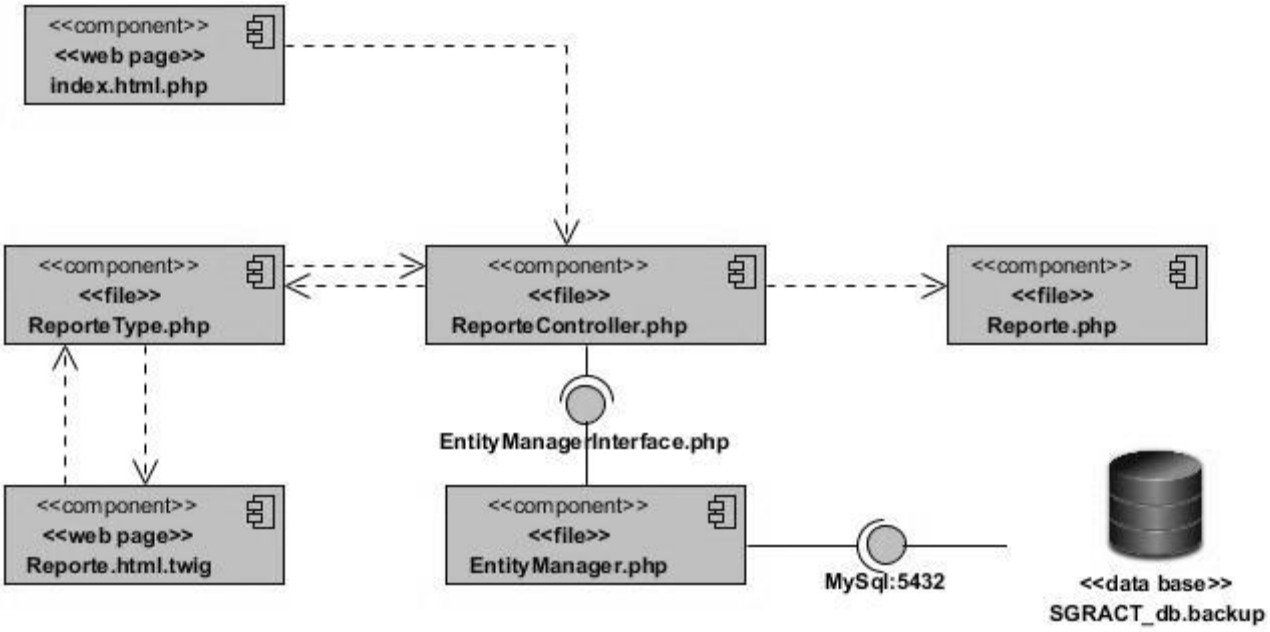


Ilustración 7 Diagrama de componentes. Fuente: Elaboración Propia.

3.2. Estándares de codificación

3.2.1. Indentación, llaves de apertura y cierre, y tamaño de las líneas

Usar una indentación sin tabulaciones, con un equivalente a 4 espacios. El uso de las llaves "{}" es seguido del nombre del método. La longitud de las líneas de código es aproximadamente de 75-80 caracteres. Para mantener la legibilidad del código.

```
public function indexAction()
{
    return $this->render( view: '@App/Admin/login');
}
```

Ilustración 8 Ejemplo de identación, llaves de apertura y cierre, y tamaño de las líneas. Fuente: Elaboración Propia.

3.2.2. Convención de nomenclatura

Variables: se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

```php
namespace AppBundle\Entity;


use ...

/**
 * Estado del Plan de Evaluaciones
 * Class Indication1
 * @package AppBundle\Entity
 * @ORM\Entity(repositoryClass="AppBundle\Entity\Repository\Indication1Repository")
 */
class Indication1 extends FiveFieldsIndications
{

    /**
     * Tipo de evaluacion
     * @var integer
     * @ORM\ManyToOne(targetEntity="Nomenclador")
     * @Assert\NotNull(message="app.validations.nevaluacion", groups={"indication1"})
     */
```

Ilustración 9 Ejemplo de convención de nomenclatura en variables. Fuente: Elaboración Propia.

Clases: siempre comienzan con mayúscula, en caso de nombre compuesto la primera letra con mayúscula.

```php
namespace AppBundle\Entity;


use ...

/**
 * Estado del Plan de Evaluaciones
 * Class Indication1
 * @package AppBundle\Entity
 * @ORM\Entity(repositoryClass="AppBundle\Entity\Repository\Indication1Repository")
 */
class Indication1 extends FiveFieldsIndications
{

    /**
     * Tipo de evaluacion
     * @var integer
     * @ORM\ManyToOne(targetEntity="Nomenclador")
     * @Assert\NotNull(message="app.validations.nevaluacion", groups={"indication1"})
     */
```

Ilustración 10 Ejemplo de convención de nomenclatura en clases. Fuente: Elaboración Propia.

Funciones: se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula terminando con la palabra Action. Los parámetros son separados por espacio luego de la coma que los separa.

```php
public function indexAction()
{
    return $this->render( view: '@App/Admin/login');
}
```

Ilustración 11 Ejemplo de convención de nomenclatura en funciones. Fuente: Elaboración Propia.

Ficheros: Todo siempre en minúscula y en caso de nombres compuestos se usa el carácter Subrayado” _”.

Vistas: Debe ser intuitivo y relacionado con el formulario y/o vista que representa.

Modelos: con la misma clase que representa.

Librerías: con la misma clase que representa.

Controladoras: con la misma la clase que representa, terminando con la palabra Controller.

3.3. Estrategia de validación de software

Durante la etapa de implementación, pueden cometerse algunos errores y pueden pasarse por alto algunos elementos que son importantes para el correcto funcionamiento del sistema. Por tal motivo es esencial llevar a cabo la fase de validación, en la cual a través de varios tipos y métodos de pruebas de *software* (estrategia de pruebas), se pretende comprobar el cumplimiento de las especificaciones del diseño y de la codificación, identificar los posibles errores cometidos y validar la solución propuesta en los capítulos anteriores (Carrizo & Rojas, 2016). En este epígrafe se muestran los resultados de la estrategia de prueba diseñada para el sistema de gestión de reportes para análisis de tendencia de calidad (ver tabla 5) en función de garantizar y validar su calidad.

Tabla 5 Validación de la propuesta de solución. Fuente: Elaboración Propia.

Tipo de prueba	**Método (técnica) de prueba**	**Validación**
Funcional	Casos de prueba(Caja Negra)	Valida las funcionalidades diseñadas para el sistema
Usabilidad	Listas de chequeo	Valida, a partir de sus características, la capacidad del sistema de cumplir con el propósito para el que fue diseñado
Seguridad	*Software Acunetix*	Valida la confidencialidad, integridad y disponibilidad de los datos en el sistema
Aceptación	Carta de aceptación del cliente	Valida la funcionalidad del sistema a partir de la aceptación determinada por el usuario (cliente) del mismo
Carga y estrés	*Software Apache JMeter*	Valida el comportamiento del

		sistema con distintos niveles de usuarios concurrentes y el consumo excesivo de sus recursos

3.3.1. Pruebas funcionales

Se denominan pruebas funcionales, a las pruebas de *software* que tienen por objetivo probar que los sistemas desarrollados cumplan con las funciones específicas para los cuales han sido creados. A este tipo de pruebas se les denomina también pruebas de comportamiento o pruebas de caja negra, pues los probadores o analistas de pruebas, no enfocan su atención a cómo se generan las respuestas del sistema. Básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas (Escobar-Sánchez & Fuertes-Díaz, 2015).

Con el objetivo de realizar este tipo de pruebas al sistema de gestión de reportes para análisis de tendencia de calidad, se diseñó un conjunto de casos de pruebas, nueve en total, referentes a varias de las historias de usuarios obtenidas en la fase de diseño del capítulo anterior, pertenecientes a requisitos funcionales de prioridad alta, también especificados en dicho capítulo. A continuación, se muestra uno de los casos de prueba mencionados. En las celdas de la tabla del caso de prueba (ver tabla 5) se pueden encontrar los valores V, para datos válidos, I, para datos inválidos, y N/A, para datos a los que no es necesario proporcionarles un valor.

Tabla 6 Descripción de las variables de caso de prueba 1: RF4_Gestionar el listado de indicadores. Fuente: Elaboración Propia

Variable	**Nombre de campo**	**Clasificación**	**Valor nulo**	**Descripción**
1	Nombre	Campo texto	No	Permite solo letras
2	Descripción	Campo texto	No	Permite solo letras

3	Tipo	Campo selección múltiple	No	Puede tomar valores
4	Activo	Casilla de verificación	No	Este campo puede estar activado o desactivado

Tabla 7 Caso de prueba RF4_Gestionar el listado de indicadores. Fuente: Elaboración Propia.

Caso de prueba 1: Gestionar el listado de indicadores							
Condiciones de ejecución: El usuario debe tener los permisos necesarios							
Escenario	**Descripción**	**1**	**2**	**3**	**4**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crear el nomenclador	Interfaz con el formulario para llenar los datos del nomenclador, si todos son correctos, se agrega el nomenclador al sistema	V	V	V	V	Agrega el nomenclador y muestra mensaje de notificación	1.Seleccionar en configuraciones la opción nomencladores. 2.Seleccionar la opción adicionar. 3.Llenar los campos correspondientes
		RNA	Esto es una prueba	Nivel	Activado		

EC 1.2 Crear el nomenclador con campos vacíos	Interfaz con el formulario para llenar los datos del ejercicio, si existe algún campo vacío, se muestra un mensaje pidiendo llenar el campo	I	I	I	V	Comprueba si los campos están vacíos, si lo están, muestra un mensaje que solicita llenarlos	en el formulario y seleccionar la opción aceptar.
					Desactivado		
EC 1.3 Crear nomenclador con el mismo nombre a otro existente	Interfaz con el formulario para llenar los datos del ejercicio, si existe algún ejercicio con el mismo nombre, se muestra un mensaje de error	I	V	V	V	Comprueba si existe algún nomenclador con el mismo nombre, si existe muestra un mensaje de notificación	
		RNA	Esto es una prueba	nivel	Activador		

Como resultado final de las pruebas funcionales, se obtuvo, en una primera iteración, un total de diez no conformidades, divididas en tres de ortografía, dos de redacción, una de funcionalidad y cuatro de validación. De estas, se resolvieron todas las no conformidades. En una segunda iteración no se identificaron nuevas no conformidades. La siguiente gráfica, muestra los resultados antes descritos:

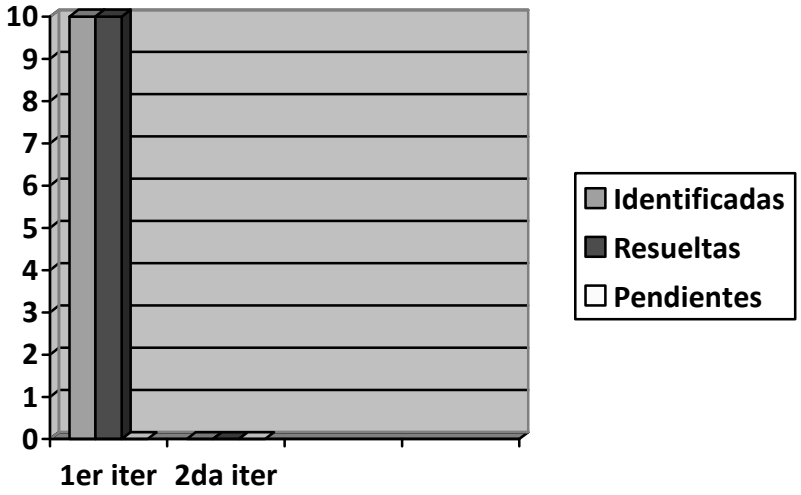


Ilustración 12 Resultados de las pruebas funcionales. Fuente: Elaboración Propia.

La no conformidad de funcionalidad, estuvo relacionada con eliminar un nomenclador ya que no se permitía eliminarlo si estaba relacionado con algun proyecto. La misma fue resuelta al realizar modificaciones en la bases de datos eliminando la relación entre el nomenclador y el proyecto estableciendo la misma a través de los indicadores.

Las no conformidades de validación fueron encontradas en el formulario para crear un nomenclador, donde se permitía introducir caracteres extraños en el nombre de un nomenclador. La solución para los caracteres extraños fue validar el nombre del ejercicio mediante una expresión regular que permite solo el uso de caracteres alfa-numéricos, las tildes, las diéresis, espacios, puntos y el guión bajo. El sistema muestra un mensaje de error si se intenta realizar esta acción de manera incorrecta.

3.3.2. Pruebas de seguridad

Las actividades de prueba de seguridad se llevan a cabo principalmente para demostrar que un sistema cumpla con sus requisitos de seguridad y para identificar y minimizar el número de vulnerabilidades de seguridad en el *software* antes de que el sistema entre en producción. (Ramírez, 2016). Además, el objetivo de las pruebas de seguridad es asegurar que el *software* que está siendo probado es sólido y funciona de una manera aceptable incluso en la presencia de un ataque malicioso. (Londoño-Palacio, N., Hidalgo-Martínez, P., Páez-Mo, 2014).

Para la realización de este tipo de prueba, se empleó la herramienta *Acunetix Web Vulnerability Scanner*. En una primera iteración, se obtuvo un total de quince no conformidades, divididas en seis de nivel alto, seis de nivel medio y tres de nivel bajo. De las de nivel alto destacó los campos de contraseña con autocompletamiento activado. De las de nivel medio, destacó el uso del protocolo no seguro para el envío de datos. Las de nivel bajo estuvieron relacionadas con problemas para la protección contra ataques de fuerza bruta a la página de autenticación, así como directorios que pueden ser accesibles directamente sin pasar la autenticación y la protección de las cookies y las sesiones en el navegador. Todas estas deficiencias fueron corregidas en la primera iteración, y para una segunda, no se identificó ninguna nueva, por lo cual se obtuvo finalmente una herramienta que cumple con los requisitos de seguridad definidos para la misma. Los resultados antes descritos, se muestran a continuación en la siguiente gráfica:

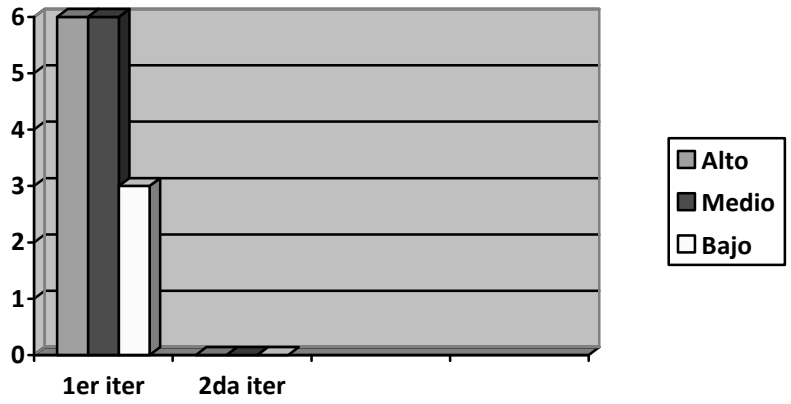


Ilustración 13 Resultados de la prueba de seguridad. Fuente: Elaboración Propia.

3.3.3. Pruebas de usabilidad

Según diversos estándares de la Ingeniería de *Software*, se puede definir la usabilidad como el grado en el que un producto puede ser utilizado por usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción, en un determinado contexto de uso. Como se puede apreciar, la usabilidad de un sistema está ligada a usuarios, necesidades y condiciones específicas (Enriquez, J. G., & Casas, S. I. , 2014). De manera general, el término usabilidad se emplea para referirse a la capacidad que posee un producto de ser utilizado por los usuarios de forma fácil, eficiente y con satisfacción, en un determinado contexto de uso.

Para la realización de las pruebas de usabilidad, se hace uso de la “Lista de Chequeo de Usabilidad para sitios web”, desarrollada por los especialistas del grupo de Seguridad del Departamento de Evaluación de Productos de *Software* (DEPSW), perteneciente al Centro Nacional de Calidad de *Software* (CALISOFT). A continuación, se muestran los resultados de dichas pruebas.

La tabla 8 refleja las categorías en las que se dividen los indicadores, cuántos indicadores existen por cada categoría, de ellos, cuántos se aplican a la evaluación del sistema de gestión de reportes para análisis de tendencia de calidad (Proceden), y de los evaluados, cuántos están implementados correcta e incorrectamente en el sistema.

Tabla 8 Resultados de las pruebas de usabilidad. Fuente: Elaboración Propia.

Categoría de los indicadores	**Indicadores**	**Proceden**	**Correctos**	**Incorrectos**
Visibilidad del sistema	17	15	13	2
Lenguaje común entre el sistema y el usuario	12	6	6	0
Libertad y control por parte del usuario	29	18	16	2
Consistencia y estándares	33	21	20	1
Estética y diseño minimastista	17	11	11	0
Prevención de errores	8	6	6	0

Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores	11	7	7	0
Ayuda y documentación	11	3	3	0
Flexibilidad y eficiencia	6	6	6	0
Total	144	93	88	5

Como se observa en la tabla, de los 144 parámetros originales de la lista de chequeo, solo proceden 93. En una primera iteración se evaluaron como correctos ochenta y ocho parámetros, identificando cinco no conformidades, para un 94,6 % de usabilidad. Los principales problemas estuvieron relacionados con la visibilidad del sistema, la libertad y control por parte del usuario y la consistencia y estándares. Las no conformidades se revisaron y de las cinco encontradas, fueron solucionadas todas, obteniendo un sistema con un 100 % de usabilidad.

3.3.4. Pruebas de aceptación

Según Mera (2016), dentro del tipo de pruebas que existe para validar el *software*, una de las principales es la prueba de aceptación (*user's tests*). Dicha prueba se diseña por el propio equipo de desarrollo en base a los requisitos funcionales especificados en la fase de análisis para cubrir todo ese espectro. Es ejecutada por el propio usuario final, no por todos evidentemente, pero sí por una cantidad de usuarios finales significativo que den validez y conformidad al producto que se les está entregado en base a lo que se acordó inicialmente.

Para realizar la prueba de aceptación, se entregó la aplicación al cliente, el cual emitió su criterio a través de una carta de aceptación, a partir de sus consideraciones respecto a las ventajas que ofrece el sistema y las necesidades que resuelve. Para respaldar dicho criterio, se solicitó la colaboración de

un grupo de expertos de calidad de *software*. El instrumento utilizado es una encuesta (ver Anexo 2) a través de la cual se pueda validar el nivel de aceptación del sistema de gestión de reportes en los usuarios finales. La encuesta se realizó a cuatro asesores de calidad los cuales cuentan con un promedio de cuatro años de experiencia ocupando dicho cargo.

La tabla 9 resume el resultado de los juicios emitidos por los encuestados, de acuerdo a los siguientes parámetros:

1. Tratamiento de los aspectos teóricos de la calidad de *software*.

2. Nivel de apoyo que brinda al asesor el uso del sistema de gestión de reporte para la orientación y evaluación de *software*.

3. Contribución del sistema de gestión de reportes al proceso de toma de decisiones en el manejo de información de tendencia.

4. Presentación de una interfaz agradable e intuitiva para el usuario.

5. Usabilidad del sistema.

Para el procesamiento y análisis de la información obtenida se analizaron las respuestas de cada uno de los parámetros que aparecen en la encuesta. De esta manera se presentan los resultados teniendo en cuenta que los niveles empleados para la valoración fueron: **MA**: Muy adecuado, **A**: Adecuado, **PA**: Poco adecuado y **NA**: No adecuado. En la tabla 9, N se refiere a la cantidad de encuestados que emitieron una valoración determinada y % al porciento que representa con respecto al total de encuestados.

Tabla 9 Resultado de la encuesta realizadas. Fuente: Elaboración Propia.

Parámetros evaluados	**Niveles de valoración**									
	MA		**A**		**PA**		**NA**		**Total**	
	N	**%**	**N**	**%**	**N**	**%**	**N**	**%**	**N**	**%**
1	4	100	0	0	0	0	0	0	4	100
2	2	50	2	50	0	0	0	0	4	100

3	2	50	2	50	0	0	0	0	4	100
4	3	75	1	25	0	0	0	0	4	100
5	4	100	0	0	0	0	0	0	4	100

Se observa, al analizar los resultados de la encuesta se consta que el 100% de los encuestados considera muy adecuado el tratamiento de los aspectos teóricos de la calidad de *software*. El nivel de apoyo que brinda el sistema al asesor de calidad, así como la contribución del sistema de gestión de reportes al proceso de toma de decisiones en el manejo de información de tendencia fueron valorados de muy adecuado por el 50% de los encuentados, el otro 50% considera que estos aspectos son afrontados de manera adecuada. Respecto a la presentación de una interfaz agradable e intuitiva para el usuario, solo el 75% de los encuestados considera que es muy adecuada, sin embargo, el 100% de ellos considera muy adecuada su usabilidad.

Teniendo en cuenta los criterios, así como otras consideraciones expresadas por los asesores de calidad en la encuesta, se puede comprobar que la solución implementada tiene un nivel satisfactorio de aceptación para ellos. Los aspectos evaluados, los cuales están en concordancia con el objetivo general de la investigación, fueron valorados todos entre los niveles de adecuado y muy adecuado. Esto demuestra el correcto cumplimiento de este objetivo, desde el punto de vista de los expertos. Se obtuvo además un conjunto de recomendaciones y valoraciones que aportan mejoras a la propuesta de solución, las cuales se tienen en cuenta para el resultado final de la investigación, así como futuras profundizaciones sobre la misma.

3.3.5. Pruebas de carga y estrés

La prueba de carga y estrés se refiere a la práctica de comprobar el comportamiento de una aplicación mediante cargas o entradas pesadas. Las mismas se diseñan para asegurar que el sistema pueda procesar su carga esperada. Estas se ocupan tanto de demostrar que el sistema satisface sus requerimientos, como de descubrir problemas y defectos en el sistema (Moncayo, L. A. S., Barragán, M. S., Golondrino, G., 2016).

Para la realización de esta prueba se utilizó la herramienta Apache JMeter v. 2.10. Las pruebas se realizaron desde un ordenador con memoria RAM de 8.00 GB Microprocesador: Intel(R) Core(TM) i3-

7100U CPU @2.40GHz 2.40GHz y sistema operativo: *Windows* v.10. A continuación, se describen las variables que miden el resultado de las pruebas de carga y estrés realizadas al sistema:

Muestra: Cantidad de peticiones realizadas para cada URL.

Media: Tiempo promedio en milisegundos en el que se obtienen los resultados.

Mediana: Tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.

Min: Tiempo mínimo que demora un hilo en acceder a una página.

Max: Tiempo máximo que demora un hilo en acceder a una página.

Línea 90 %: Máximo tiempo utilizado por el 90 % de la muestra, al resto de la misma le llevó más tiempo.

% Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.

Rendimiento (Rend): El rendimiento se mide en cantidad de solicitudes por segundo.

Kb/s: Velocidad de carga de las páginas.

Como se muestra en la siguiente tabla, se simularon las peticiones realizadas al sistema por un total de cincuenta usuarios simultáneamente, los cuales realizan hasta cinco peticiones por segundo. Se obtuvieron los siguientes resultados:

Tabla 10 Resultados de la prueba de carga y estrés. Fuente: Elaboración Propia.

Usuarios	Muestras	Media	Mediana	Min	Max	Linea 90%	%Error	Rend	Kb/s
50	200	5760	5677	410	27770	7151	0.00%	7.9	41.6

Las pruebas realizadas muestran que el sistema es capaz de responder a doscientas peticiones de cincuenta usuarios conectados simultáneamente en un tiempo promedio de 5760 milisegundos (5.7 segundos aproximadamente) con 0 % de error, esto evidencia que el sistema puede procesar la carga esperada.

3.4. Interfaces principales del sistema de gestión de reportes para análisis de tendencias de calidad

Una vez desarrollado el sistema, es posible visualizar las pantallas principales del mismo, donde se observa el resultado obtenido durante la implementación de las historias de usuarios descritas en el epígrafe 2.5.

Ilustración 14 Interfaz de autenticación Fuente: Elaboración Propia.

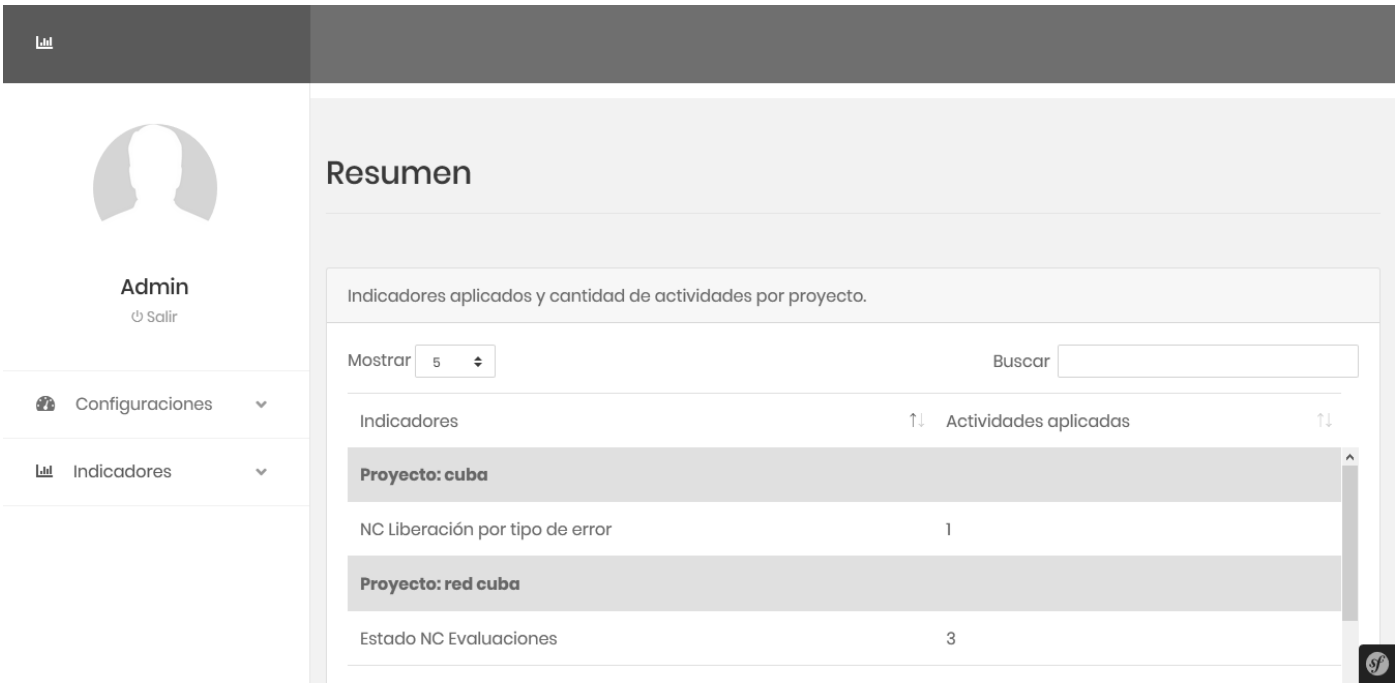


Ilustración 16 Interfaz de autenticación Fuente: Elaboración Propia.

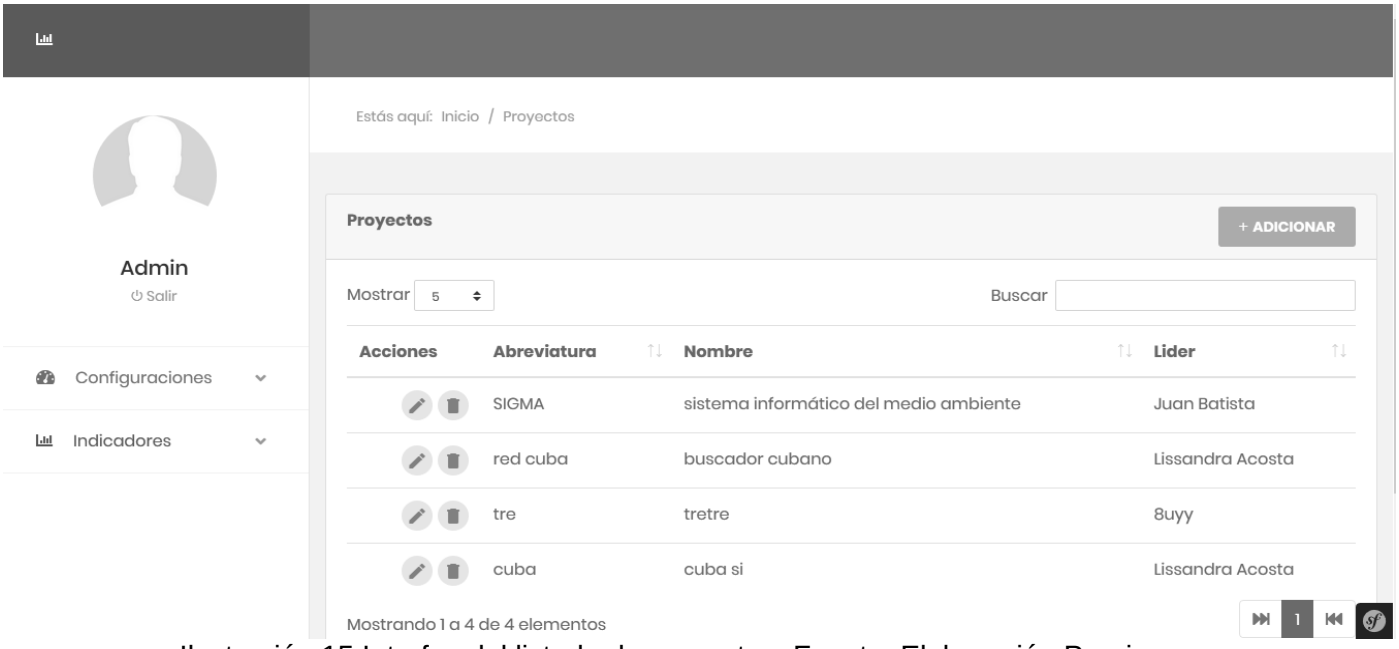


Ilustración 15 Interfaz del listado de proyectos. Fuente: Elaboración Propia.

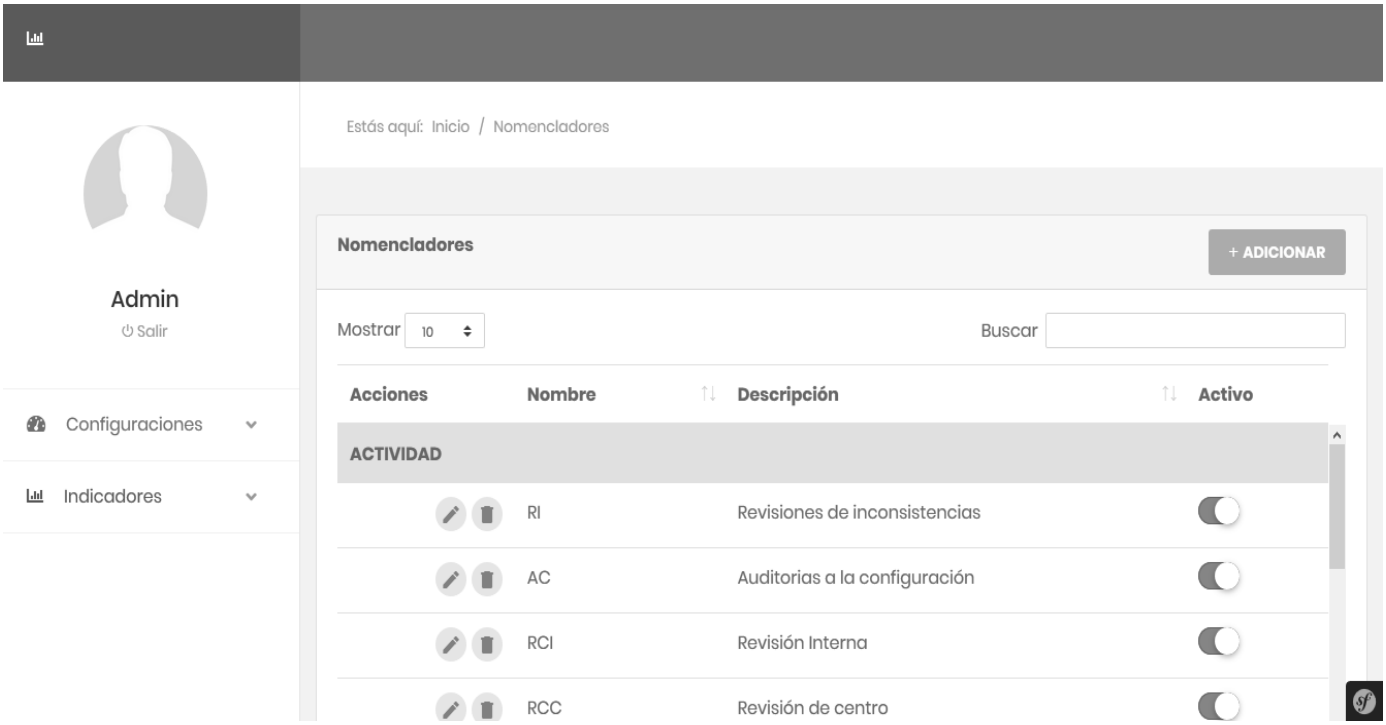


Ilustración 16 Interfaz del listado de nomencladores. Fuente: Elaboración Propia.

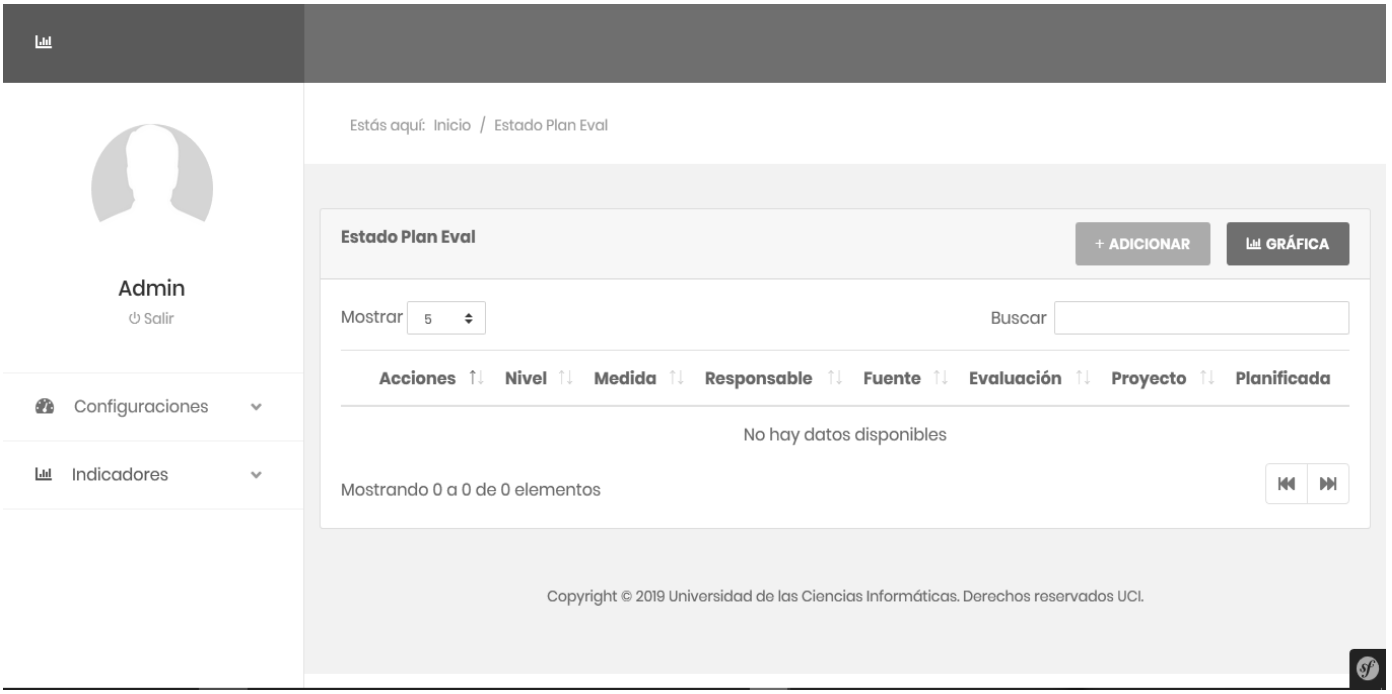


Ilustración 17 Interfaz de estado del plan de evaluaciones. Fuente: Elaboración Propia.

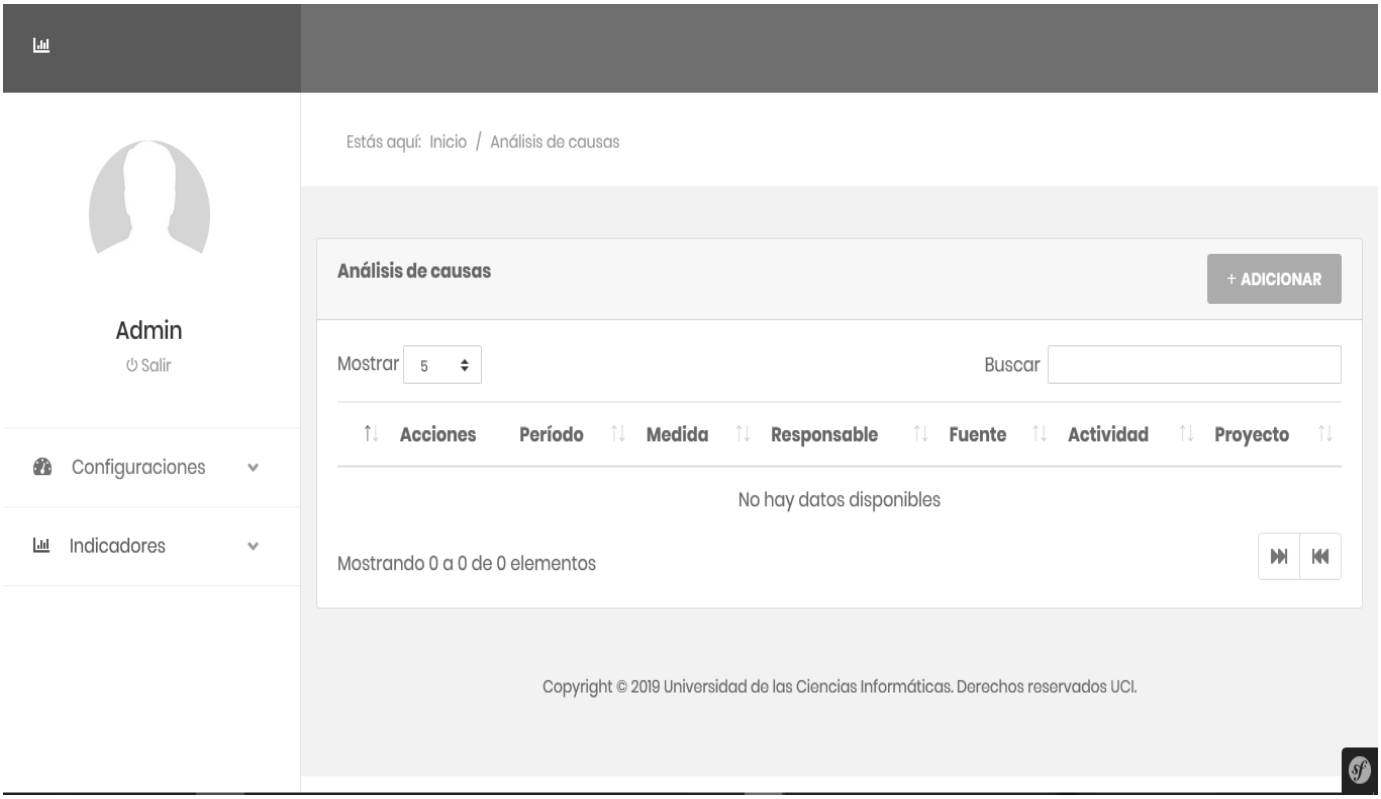


Ilustración 18 Interfaz de análisis de las causas. Fuente: Elaboración Propia.

3.5. Conclusiones parciales

En este capítulo se han abordado los elementos de la implementación del sistema de gestión de reportes para análisis de tendencia de calidad, así como las pruebas realizadas al mismo y los resultados obtenidos; lo cual permite arribar a las siguientes conclusiones:

- La elaboración del diagrama de componentes permite una mejor comprensión de la estructura de los componentes del sistema implementado.
- El correcto uso de los estándares de codificación cede que el código del sistema desarrollado fuera legible para lograr una fácil y mejor comprensión del mismo, la cual es de utilidad para el mantenimiento del sistema.

- La implementación del sistema permite la obtención de una aplicación funcional y completamente operativa.
- El proceso de validación de la propuesta de solución a través de la estrategia de pruebas especificada, arroja como resultado que el sistema implementado responde a los requerimientos definidos por el cliente.

CONCLUSIONES GENERALES

De manera general, la presente investigación concluyó con el desarrollo del sistema de gestión de reportes para análisis de tendencia de calidad, el cual sirve de apoyo al proceso de generación de informes de tendencia y contribuye a una correcta toma de deciciones en el manejo con información de tendencia.

Otros aspectos significativos que se pueden destacar son:

- El análisis de los elementos teóricos y técnicos que sustentaron la investigación permitió definir las herramientas, tecnologías y metodologías a utilizar para el desarrollo del sistema de gestión de reportes para análisis de tendencias de calidad.
- A partir de las funcionalidades descritas se realizó el análisis y diseño del sistema de gestión reportes para análisis de tendencias de calidad, proporcionando el punto de partida para ejecutar de forma organizada las tareas de implementación.
- La implementación del sistema a través de las herramientas y lenguajes seleccionados permitió obtener una aplicación web capaz de manejar datos referentes a la gestión de reportes de calidad basado en el análisis de tendencias de calidad.
- Las pruebas ejecutadas al software fueron un punto clave para detectar y corregir errores en el software, los resultados obtenidos al aplicar las técnicas cualitativas y cuantitativas de validación reafirmaron el correcto funcionamiento del sistema de gestión de reportes para análisis de tendencias de calidad.

RECOMENDACIONES

- Valorar la factibilidad de extender la solución a los demás centros de produccion de la UCI que comparten necesidades similares respecto al informe de tendencias.
- Aplicar una limpieza de datos a la base de datos utilizada por el sistema de gestión de reportes para análisis de tendencias de calidad con el objetivo de aplicar como técnica de descubrimiento de conocimiento en bases de datos la minería de datos, para clasificar, agrupar o analizar las tendencias de algún reporte.

Bibliografía

Aguilar, J., & Dos Santos, R. . (2015). Sistema Buscador de Contenidos Digitales de la Nube de Conocimiento del Proyecto Madre. *ReVeCom*.

Arias, M. (2017). *Aprende Programación Web con PHP y MySQL.* IT Campus Academy.

Ayala Narea, G., Sangoquiza, V., & Estefanía, M. (2017). *Generar un Geo-portal para el grupo de investigación en ciencias ambientales (GRICAM) de la Universidad Politécnica Salesiana.*

Beati, H. (2015). *PHP-Creador de páginas web dinámicas.*

Betancourt Agüero, Y. (2013). *Procedimiento para la evaluación de la calidad percibida del servicio educativo de pregrado en la Facultad de Ciencias Económicas del Centro Universitario de Las Tunas.*

CALISOFT. (2016). Obtenido de Quiénes Somos: https://www.calisoft.cu/#quienes_somos

Camarena , J., Trueba , A., Martínez , M., & López , M. (2016). *Automatización de la codificación del patrón modelo vista controlador (MVC) en proyectos orientados a la Web.* Ciencia Ergo Sum.

Campoverde, A., Hernández, D., & Mazón, B. (2015). Cloud computing con herramientas open-source para Internet de las cosas.

Carrizo, D., & Rojas, J. (2016). Clasificación de prácticas de educción de requisitos en desarrollos ágiles: un mapeo sistemático. *Revista chilena de ingeniería*.

Churo, Q., Rodrigo , B., Alpala, G., & Carla, V. (2010). *Sistema generador de reportes dinámicos para web, configurable para las plataformas de bases de datos más conocidas.*

Conde, C., & Gutiérrez, M. P. (2014). . La experiencia de la OCDE en la implementación de políticas de Gobierno Abierto. El desafío de la etapa de implementación. *Revista del CLAD Reforma y Democracia,*.

Cosmin, P. I. (2016). Patrones de Diseño. *MoleQla: Revista de Ciencias de la Universidad Pablo de Olavide*.

Daros, W. (2017). ¿ Qué es un marco teórico? *Enfoques*.

Daud, N. I., Bakar, K. A. A., & Hasan, M. S. M. (2014). A case study on web application vulnerability scanning tools. *Science and Information Conference (pp. 595-600). IEEE.*

easyRedmine. (2018). Obtenido de Easy Redmine 2018: https://www.opshub.com/integrations/hp-alm-qc-integration/

Enriquez, J. G., & Casas, S. I. . (2014). Usabilidad en aplicaciones móviles.

Escobar-Sánchez , M., & Fuertes-Díaz, W. (2015). Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2. .*Facultad de Ingeniería*.

Eurotherm. (2018). Obtenido de Software Dream Report: https://www.eurotherm.es/products/programmable-automation-controller-system/pac-software/dream-report

Fortis Urbano, J. (2015). *Base de datos para encontrar relaciones entre proyectos.*

Gimenez, C., Braun, G., & Fillottrani, P. (2016). *Una arquitectura cliente-servidor para modelado conceptual asistido por razonamiento automático.*

Gómez, S., & Roquet, J. V. (2012). *Metodología de investigación.* Mexico .

Guchat, J. (2012). *El gran libro de HTML5,CSS3 Y JAVASCRIPT.*

Guerrero, C., Suárez, J., & Gutiérrez, L. (2013). *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* Información tecnológica.

Hiebaum, J. (2015). *Programación Extrema XP.*

Hugo, V., & Rmos, S. (2015). *Aplicación de metodología web en el desarrollo de un módulo para la búsqueda de contenido en un sistema educativo.*

Kelly, M. (30 de 12 de 2010). *Introducción a IBM Rational Quality Manager*. Obtenido de IBM developerWorks: https://www.ibm.com/developerworks/ssa/rational/library/08/1230_kelly/index.html

Londoño-Palacio, N., Hidalgo-Martínez, P., Páez-Mo. (2014). Guía para la certificación de servicios diagnósticos de trastornos del sueño en Colombia.

Luján Mora, S. (2013). *Ejercicio:CSS conceptos básicos.*

Maldonado, T., & Mauricio, W. (2017). *Estudio de la integración de los framework bootstrap y primefaces para el desarrollo de aplicaciones web adaptativas con java server faces Aplicativo: Sistema de control de notas, para la unidad educativa mariano Suarez Veintimilla .*

Martinez Jacomino, A. (2017). *Aplicación web para el control del almaéen de la cervecería Manacas.*

Martínez, A., & Martínez, R. (2014). Guía a rational unified process.

Mayo, I. C., & Fernández, J. L. (2016). Los procesos de gestión de calidad. *Revista Iberoamericana sobre Calidad ,Eficacia y Cambio en la educación* .

Mazalu, R. C. (2013). Evaluación de accesibilidad del contenido web utilizando agentes. *In XVIII Congreso Argentino de Ciencias de la Computación.*

Mejoras del Proceso del Software. (2018). Obtenido de Actividades de Calidad: http://mejoras.prod.uci.cu/

Méndez, J. (2013). *Calidad, concepto y filosofías.*

Mera Paz, J. (2016). *Análisis del proceso de pruebas de calidad de software.*

Molina, J., Rubio, F., Pelechano, V., Vallecillo, A., Vara, J., & Vicente-Chicote, C. (2014). *Desarrollo de software dirigido por modelos: conceptos, métodos y herramientas. .*

Moncayo, L. A. S., Barragán, M. S., Golondrino, G. (2016). Video on demand service based on the inference of emotions user. *Sistemas & Telemática*.

Moraga , J., & Cartes-Velásquez, R. (2015). *Pautas de chequeos.*

opshub integration & Migration Solutions. (2014). Obtenido de HP Application Lifecycle Management (HP ALM)/HP Quality Center (HPQC) integrations using OpsHub Integration Manager: https://www.opshub.com/integrations/hp-alm-qc-integration/

Ovalle, D., Salazar, O., & Duque, N. (2014). Modelo de recomendación personalizada en cursos virtuales basado en computación ubicua y agentes inteligentes. *Información tecnológica*.

Oyarce, J., Orellana, C., & Flores, H. (2016). Reporte integrado: Nuevo paradigma en la información corporativa. *Horizontes Empresariales*.

Passadore, N. (2017). *Recomendaciones de artículos a través de PUBMED.*

Patel, B., Parikh, J., & Shah, R. (2014). A Review Paper on Comparison of SQL Performance Analyzer Tools: Apache JMeter and HP LoadRunner. *International Journal of Current Engineering and Technology.*

Patiño Mora, C. H. (2016). Desarrollo de una aplicación web que automatice el proceso de registro de matrículas y notas en la Escuela Fiscomisional Mercedes Navarrete utilizando la tecnología Symfony.

Pinto, M., Uribe Tirado, A., Gómez Díaz, R., & Cordón, J. (2011). La producción científica internacional sobre competencias informacionales e informáticas. *Información, Cultura y Sociedad*.

Piñero, P. Y. (11 de 2011). *ResearchGate*. Obtenido de gespro 11.05 un sistema para la dirección integrada de proyectos para la gestión de la producción: https://www.researchgate.net/publication/283459955_GESPRO_1105_UN_SISTEMA_PARA_LA_DIRECCION_INTEGRADA_DE_PROYECTOS_PARA_LA_GESTION_DE_LA_PRODUCCION

Puello, O. . (2013). Modelo de verificación y validación basado en CMMI. *Investigación e Innovación en Ingenierías*.

Quevedo, J., & Suárez, J. (2015). Sistema informático para el monitoreo y evaluación de la producción de alimentos y energía. *BiomaSoft*.

Ramírez, C., & Alférez, L. (2014). Modelo conceptual para determinar el impacto del merchandising visual en la toma de decisiones de compra en el punto de venta. *Revista científica Pensamiento y Gestión*.

Ramírez, L. G. (2016). Estrategia de validacion para aplicaciones móviles de salud.

Rodríguez Sánchez, T. (2015). *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana.

Romero, N. (2016). Propuesta de extensión UML para procesos de conceptualización de requisitos . *Revista Latinoamericana de Ingenieria de Software*.

Sánchez, L., & Sánchez, R. (2018). Sistema de gestión bibliotecaria ABCD. *Revista Publicando*.

Schwaber, K., & Sutherland, J. (2013). *La guía de scrum.*

Suárez Ibujes, M. O., & Tapia Zambrano, F. A. (2013). *Interaprendizaje de estadística básica .*

Tedeschi, N., Creacionales, P., Estructurales, P.,. (2015). ¿Qué es un Patrón de Diseño? *Retrieved marzo.*

Terstine, M. (2015). El Progreso de la Investigación en Ingeniería de Requisitos. *Revista Antioqueña de las Ciencias Computacionales.*

Valle Dávila, M. (2017). *Estudio del framework symfony 2 para el desarrollo de aplicaciones empresariales.*

Vela, S., & Reyes, R. (2017). Portal de datos abiertos de la Universidad Cental del Ecuador. *Investigación y desarrollo*.

Villada, R. (2015). *Instalación y configuración del software de servidor web.* Madrid,ES:IC Editorial.

Visconti, M., & Astudillo, H. (2012). Fundamentos de Ingeniería de Software.

Anexos

Anexo 1. Entrevista al cliente para conocer la necesidad del desarrollo de la propuesta de solución y definir los requisitos funcionales y no funcionales.

Estimado profesor o estudiante: Se necesita de su cooperación en una investigación para una tesis de pregrado. Por ello, sería de gran ayuda que respondiera lo siguiente:

1. ¿Cómo funciona el proceso de generación de reportes de tendencias de calidad en la actualidad?
2. ¿Cómo recopilar la información de cada proyecto?
3. ¿Qué otras funcionalidades considera que debe prensentar el sistema?
4. ¿Existe algún sistema en la Universidad que realice reportes de tendencia de calidad?
5. ¿Cuáles son sus características?

Anexo 2. Encuesta al profesor para evaluar el nivel de aceptación del sistema de gestión de reportes.

Estimado asesor: Se necesita de su cooperación en una investigación para una tesis de pregrado para evaluar el nivel de aceptación de una herramienta web, por ello sería de gran ayuda que respondiera lo siguiente, con la sinceridad y seriedad que el proceso necesita.

Respecto al sistema de gestión de reportes para análisis de tendencias de calidad, emita su valoración respecto a los siguientes parámetros: MA, si valora el parámetro como Muy Adecuado; A, si lo valora como Adecuado; PA, si considera que es Poco Adecuado; o NA si considera que es No Adecuado. Marcar con una X.

No.	**Parámetros**	**Valoración**			
1	Tratamiento de los aspectos teóricos de la calidad de software	**MA**	**A**	**PA**	**NA**
2	Nivel de apoyo que brinda al asesor el uso				

	del sistema de gestión de reporte para la orientación y evaluación de *software*.				
3	Contribución del sistema de gestión de reportes al proceso de toma de decisiones en el manejo de información de tendencia.				
4	Presentación de una interfaz agradable e intuitiva para el usuario				
5	Usabilidad del sistema.				

Anexo 3. Carta de aceptación

Estimados señores:

Por este medio informo el trabajo desarrollado por la estudiante Lissandra Acosta Obregón quien fue asesorada por: Yennifer Delgado Mesa y Paul Rodríguez Leyva, bajo el título “Sistema de gestión de reportes para análisis de tendencias de calidad”

Estoy satisfecha con la estructura y diseño de la aplicación, la cual contiene los formularios necesarios para generar el reporte final como:

- ✓ Proyectos
- ✓ Nomencladores
- ✓ Estado del plan de evaluaciones
- ✓ Tendencia de las no conformidades
- ✓ Adherencia a procesos y productos de trabajo
- ✓ Análisis de las causas
- ✓ Estado de las no conformidades de las evaluaciones
- ✓ Apreciación de la utilidad de los procesos y productos de trabajo
- ✓ Estado de las expectativas de la organización
- ✓ Estado del plan de liberaciones
- ✓ No conformidades por tipo de error según su clasificación
- ✓ Iteraciones según artefacto a liberar
- ✓ No conformidades por iteraciones por tipo de error según su clasificación
- ✓ Estado de las no conformidades por iteraciones

Dicho sistema fue diseñado con el fin de automatizar el proceso de generación de reportes para análisis de tendencias de calidad el cual cumple correctamente con todos los requisitos funcionales encontrados. Por lo anteriormente descrito y luego de haber realizado varias veces el flujo completo del proyecto se concluye que el sistema tiene un 100 % de usabilidad y contiene además la aceptación del cliente.

Anexo 4. Historias de Usuarios

Historia de usuario	
Número: 1.	**Usuario:** Asesor de calidad.
Nombre historia: Autenticar usuario.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Media.

Tiempo estimado: 8 horas	**Iteración asignada:** 3.
Programador responsable: Lissandra Acosta Obregón	
Descripción: Se necesitan adicionar los datos del usuario que se desea insertar sobre el asesor de calidad del centro.	
Observaciones: se debe insertar el usuario y contraseña	

Historia de usuario	
Número: 2.	**Usuario:** Asesor de calidad.
Nombre historia: Gestionar reporte (editar, eliminar, mostrar, generar).	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alta.
Tiempo estimado: 48 horas.	**Iteración asignada:** 5.
Programador responsable: Lissandra Acosta Obregón	
Descripción: Se necesita generar el reporte por el asesor de calidad del centrol.	
Observaciones: se debe mostrar es estado de los indicadores por los que será evaluado el proyecto.	