Universidad de las Ciencias Informáticas

Facultad 8

"Herramienta para generar las estadísticas cuantitativas de las No Conformidades"

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor:

Eneida García Valón.

Tutores:

Ing. Asnier E. Góngora Rodríguez.

Ing. Heney Díaz Pérez.

Ciudad de La Habana, Cuba.
3 de junio del 2010.

Declaración de Autoría

Declaro que soy la única autora del trabajo “Herramienta para generar las estadísticas cuantitativas de las No Conformidades” y se autoriza a la Facultad 8 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste se firma el presente trabajo a los ____ días del mes de _____ del año _____.

Autores: _____
Eneida García Valón

Tutores: _____
Ing. Asnier E. Góngora Rodríguez.

_____
Ing. Heney Díaz Pérez.

Agradecimientos

A mis padres Eneida Valón Savigne y Mario García Quilarte por su inmenso apoyo y dedicación.

A toda mi familia por confiar siempre en mí, y brindarme todo su apoyo.

A mi novio Flabio Garrido por estar conmigo en todo momento y brindarme siempre su apoyo.

A nuestros profesores de la UCI que han influido en nuestra preparación como profesional.

A mis tutores los ingenieros Asnier E. Góngora Rodríguez y Heney Díaz Pérez por brindarme todo su apoyo, comprensión, y siempre estar ahí conmigo en todo momento.

A la Revolución por crear esta universidad y brindarme la posibilidad de poder estudiar en la misma, abriéndome las puertas hacia el futuro.

Dedicatoria

A todas las personas que contribuyeron a que este sueño se hiciera realidad, en especial: A mi madre Eneida Valón y mi padre Mario García por siempre estar presente cuando los necesito, por brindarme su amor, cariño, dedicación y esfuerzo todos estos años. Por ambos confiar siempre en mí, por ambos ser mi razón de ser y por lo mucho que los quiero.

A mi abuela Catalina Savigne por ser la luz que me ha guiado siempre en la vida y me ha dado fuerzas para continuar.

A mi tía Nereida Duarte por confiar todos estos años en mí, y brindarme su apoyo y cariño incondicional.

Resumen

La investigación está asociada al Laboratorio Industrial de Pruebas de Software (LIPS) en la Universidad de las Ciencias Informáticas (UCI). En la misma se pretende dar respuesta a la siguiente interrogante: ¿Cómo contribuir a automatizar el proceso para llevar a cabo el control estadístico cuantitativo de las No Conformidades de las pruebas de liberación? Para el logro de esta investigación se ha implementado una herramienta para automatizar el proceso del control estadístico de las No Conformidades diseñando una aplicación web que permita tramitar todos los elementos según sus clasificaciones. Esta aplicación le permitirá, al jefe del laboratorio, a los especialistas de calidad, al administrador y al personal autorizado, interactuar a través de la red con la información almacenada en la aplicación, así como hacer reportes de seguimiento para llevar un control estadístico de todo lo referente a las pruebas de calidad de software, como también gestionar el estado de los proyectos durante las pruebas de liberación.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Introducción

A nivel mundial el desarrollo de software es cada vez más creciente y con este los clientes se vuelven cada vez más exigentes con la calidad de los mismos. Existen considerables números de organizaciones que cuentan con laboratorios altamente calificados, los cuales realizan pruebas de software y prestan diferentes servicios en función de lograr validaciones y certificaciones confiables.

Entre ellos se encuentran la Universidad de Carleton, Ottawa, Canadá; el laboratorio InQalab en Barcelona, España y el del Instituto Nacional de Ingeniería Industrial, en Argentina. Debido a esto, la calidad de software se ha convertido en el mundo globalizado de hoy, en una necesidad para permanecer en el mercado. Para que el software llegue a manos de los clientes con la calidad que se requiere, debe ser revisado por un laboratorio de pruebas. En ellos se gestionan las No Conformidades encontradas, las cuales son controladas estadísticamente de diferentes formas. Un ejemplo de ello lo constituye uno de los laboratorios de España donde se tiene un sistema llamado Omega 4, el cual supone un cambio radical en las posibilidades de gestión de los sistemas de laboratorio y el mismo se ha ido realizando bajo los estándares de calidad. Además cuenta con un certificado de No Conformidad para los centros de Barcelona y Suiza con respecto al diseño, desarrollo, producción, distribución y servicio de software para sistemas de información de laboratorios, áreas de trabajos y gestión de datos. También utiliza el Certificado de Conformidad en contenido de Innovación Tecnológica, ejemplo la AIDT (Agencia de Acreditación en Investigación, Desarrollo e Innovación Tecnológica), dependiente de la entidad nacional de acreditación española. (Roche, 2008 - 2009.)
En nuestro país se cuenta con una creciente Industria de Software, la cual se ha venido desarrollando en los últimos años con el objetivo de producir software para las empresas y entidades nacionales con el fin de informatizar la sociedad cubana y otras entidades internacionales. Con los nuevos retos que han surgido, las empresas y organizaciones se han visto en la necesidad de brindar una respuesta rápida a los clientes que se han vuelto más exigentes, no sólo en el costo, sino en la funcionalidad, integridad, portabilidad, confiabilidad y eficiencia del producto. También se les realizan pruebas internas a diversos productos, los cuales son desarrollados por algunos proyectos encargados de producir software en las universidades del país, un ejemplo de ello lo constituye la Universidad Central de las Villas Martha Abreu y la empresa DESOFT (Empresa encargada del desarrollo del software en Cuba). Estas pruebas se realizan para controlar estadísticamente las No Conformidades registradas, pero tiene como inconveniente que dichos registros se realizan de forma manual, y en ocasiones ni siquiera se lleva a cabo el control.

La UCI, surgida hace 7 años, es un ejemplo vigente de empresa que se ha venido dedicando por años al desarrollo y producción de software. En la misma radica el centro CALISOFT, encargado de prestar varios servicios a la universidad y al país, entre ellos se encuentran las pruebas de liberación que se les realizan a todos los productos de la universidad en el Laboratorio Industrial de Pruebas de Software (LIPS), que posteriormente serán comercializados tanto en nuestro país como a nivel internacional. El control estadístico de las No Conformidades en el Laboratorio Industrial de Pruebas de Software se hace de forma manual. El cual no tiene bien definido el proceso para este control estadístico, ya que se hace mediante un Excel. Además no es eficiente a la hora de controlar estadísticamente estas No Conformidades en cantidades exactas.

En CALISOFT, se realizan las pruebas de liberación a los diferentes tipos de software que se desarrollan en la Universidad de las Ciencias Informáticas. En el mismo no se tienen organizadas las clasificaciones de las No Conformidades por proyecto, iteración y revisiones, para luego realizar los reportes correspondientes. Por otro lado los directivos de la universidad no tienen cómo argumentar de una forma más rápida y eficiente, acerca de cuáles son los errores más comunes encontrados en los proyectos mediante las pruebas de liberación en el LIPS, para después tomar decisiones en el desarrollo de los proyectos en la universidad. Además el reporte de los estados de los proyectos que están liberándose se realiza de forma manual, una vez a la semana y de una forma poco eficiente.

Mediante el desarrollo de esta investigación se determinó como **problema a resolver**: ¿Cómo contribuir al control estadístico de las No Conformidades de las pruebas de liberación en el LIPS?

Para dar respuesta a esta interrogante, se decidió llevar a cabo una herramienta informática que permita automatizar y agilizar el control estadístico de las No Conformidades de las pruebas de liberación, permitiendo así tener controladas a nivel central todas las No Conformidades encontradas en todos los proyectos que se liberen, de forma organizada según su clasificación e impacto en el proyecto, así como, el estado de los proyectos después de ser sometidos a las pruebas de liberación.

Por tanto, el **objeto de estudio** es: La gestión de las No Conformidades de las pruebas de liberación.

El **campo de acción** está determinado por: El control estadístico de las No Conformidades de las pruebas de liberación.

Este trabajo tiene como **objetivo general:** Desarrollar una herramienta para obtener estadísticas de las No Conformidades de las pruebas de liberación.

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

1. Realizar un estudio del estado del arte sobre el control estadístico de las No Conformidades de las pruebas de liberación.
2. Analizar la herramienta a utilizar para generar estadísticas de las No Conformidades de las pruebas de liberación.
3. Diseñar la herramienta para generar estadísticas de las No Conformidades de las pruebas de liberación.
4. Implementar la herramienta para generar estadísticas de las No Conformidades de las pruebas de liberación.

Se definen las siguientes **tareas** para dar cumplimiento de forma exitosa a los objetivos planteados:

1. Investigación del proceso de pruebas de liberación.
2. Estudio de herramientas que realizan estadísticas de las No Conformidades.
3. Estudio de la gestión de las No Conformidades.
4. Entrevistas con los especialistas del grupo de calidad de la UCI.
5. Estudio de los documentos:
   5.1 Plantilla de reporte semanal del estado del proyecto.
   5.2 Plantilla de reporte mensual de las clasificaciones de las No Conformidades.
   5.3 Plantilla del control estadístico de las clasificaciones de las No Conformidades.
6. Estudio de las metodologías de desarrollo de software para realizar el análisis y el diseño de la aplicación.
7. Análisis y diseño de la aplicación.
8. Estudio de las tecnologías y herramientas a utilizar en la implementación del sistema.
9. Implementación y pruebas a la herramienta desarrollada.

Con las tareas definidas anteriormente se obtendrán los siguientes **resultados**:

1. Llevar a cabo una herramienta para generar las estadísticas de las No Conformidades de las pruebas de liberación.
2. Lograr con ella la rapidez en la búsqueda de información acerca del control estadístico de las No Conformidades y el estado de los proyectos obtenidos mediante las pruebas de liberación.
3. Obtener la rapidez a la hora de gestionar las estadísticas de las No Conformidades que se encuentran registradas en un documento Excel.

La estructura capitular está definida de la siguiente forma:

En el **Capítulo 1 Fundamentación Teórica**: Se refleja una breve explicación y definición del flujo actual de los procesos, su análisis crítico, así como las tecnologías y tendencias actuales que interactúan con el objeto de estudio.

En el **Capítulo 2 Características del Sistema**: Se refleja un análisis detallado del flujo actual de los procesos, así como los trabajadores que desarrollan dichos procesos. Se define además, el objeto de automatización, la información manejada y la propuesta del sistema. Se identifican los requisitos funcionales y no funcionales, los casos de uso con sus descripciones, los diagramas de actividades y el modelo de objeto.

En el **Capítulo 3 Análisis y Diseño del Sistema**: Se analizan los casos de uso del sistema para diseñar las clases que se implementarán, se representan los diagramas de colaboración del diseño, el diagrama de las clases diseñadas con sus relaciones, los principios utilizados para el diseño de dichas clases, el diagrama de clases persistentes, el diagrama entidad relación y la descripción de las tablas de la base de datos.

En el **Capítulo 4 Implementación y Prueba**: Se reflejan los diferentes diagramas correspondientes a este flujo de trabajo, en el cual se implementa la propuesta realizada en el análisis y diseño, además se representa el diagrama de despliegue, los de componentes y el modelo de prueba correspondiente a la aplicación desarrollada.

El diseño teórico ayuda a concretar el trabajo, pues en el mismo quedan definidos todos los elementos necesarios para buscar la respuesta al problema de investigación, pero también es de vital importancia el diseño metodológico para la ejecución de una investigación, pues define el tiempo necesario para su realización, el costo de la misma y la de los resultados, permitiendo así definir la estrategia de investigación más adecuada para cumplir los objetivos propuestos.

Los métodos de investigación científica que se realizan en esta investigación son:

Métodos Teóricos:

- Analítico – Sintético: se analizan documentos, sitios web, artículos electrónicos y libros, buscando las características de los diferentes modelos y herramientas existentes para la gestión estadística de las No Conformidades, permitiendo la extracción de los elementos más importantes que se relacionan con la gestión estadística de las No Conformidades en los proyectos productivos.

- Análisis Histórico Lógico: se realiza el análisis histórico lógico de los procesos para la gestión estadística de las No Conformidades que actualmente se llevan a cabo en los diferentes proyectos productivos en el Laboratorio Industrial de Pruebas de Software.

Métodos Empíricos:

Entrevista:

- Se realizan entrevistas a varios especialistas del laboratorio Industrial de Pruebas de Software, para conocer cómo se desarrolla el proceso de gestión estadística de las No Conformidades en los proyectos productivos (métodos y herramientas).
- Además se auxilian de miembros del grupo de investigación de calidad en la Universidad de las Ciencias Informáticas.

Capítulo 1 Fundamentación Teórica.

Introducción

En este capítulo se dará una breve explicación y definición del flujo actual de los procesos, su análisis crítico, así como las tecnologías y tendencias actuales interactuadas con el objeto de estudio. Se definirá la metodología a utilizar, el lenguaje de modelado, así como las herramientas que servirán de apoyo para construir la solución a la problemática planteada. Además se definirá el tipo de servidor a utilizar, el gestor de base de datos, el lenguaje Web y sobre qué arquitectura llevar a cabo el desarrollo de la investigación.

1.1 Pruebas de software.

Es un proceso que corre paralelo al proceso de desarrollo de software y se realiza por el convencimiento de que todo software debe ser revisado con el objetivo de que tenga el nivel de calidad requerido. Este debe llevarse a cabo de manera sistemática, y debe realizarse sobre métricas bien definidas. Este proceso debe comenzar en la fase de requerimientos y terminar con la finalización de la aplicación con los casos de prueba al sistema ejecutable en la fase de construcción según la metodología escogida para el diseño de la aplicación. En el proceso de pruebas se definen varios métodos, técnicas y tipos de pruebas, las cuales se mencionarán a continuación. (Grimán, y otros)

1.1.1Técnicas de pruebas.

Técnica de Caja Blanca: Basada en los requerimientos, en el diseño y en el código fuente.

Entrada: Estándar de programación, archivos fuentes y documentos de diseño.

Salidas: Documentos con defectos encontrados. Estadísticas acerca del seguimiento al estándar, así como de algunas características del sistema, tales como: el tamaño (puntos de función), la complejidad (ciclomática, estructural), la modularidad (cohesión, acoplamiento) y la legibilidad.

Técnica de Caja Negra: Es basada en los requerimientos y verifica si el sistema realmente hace lo que debe hacer, es decir, se llevan a cabo sobre la interfaz del software. Es completamente indiferente el comportamiento interno y la estructura del programa.

Entrada: Archivos ejecutables, documentos de requerimientos.

Salida: Documentos con defectos encontrados. Métricas de la aplicación de casos de pruebas. (Grimán, y otros)

1.1.2 Niveles de Pruebas.

Prueba de Unidad: Es realizada por el desarrollador y se centra en el módulo. Utilizando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. La prueba de unidad hace uso intensivo de las técnicas de prueba de caja blanca.

Pruebas de Integración: Es la que verifica si el conjunto de componentes que conforma al sistema completo interactúa correctamente. Se recomienda que esta fase sea llevada a cabo por una entidad distinta al grupo de desarrollo por cuestiones de eficiencia y objetividad. Tiene como objetivo obtener los módulos probados en la prueba de unidad y construir una estructura del programa, que esté de acuerdo con lo que dicta el diseño.

Existen dos formas de integración:

Integración no incremental: Se combinan todos los módulos por anticipado y se prueba todo el programa en conjunto.

Integración incremental: El programa se construye y se prueba en pequeños segmentos.

En la prueba de integración el foco de atención es el diseño y la construcción de la arquitectura del software.

Las técnicas que más prevalecen son las de diseño de casos de prueba de caja negra, aunque se pueden llevar a cabo unas pocas pruebas de caja blanca.

Prueba del Sistema: Verifica que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total. La prueba del sistema está constituida por una serie de pruebas diferentes, cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora.

Prueba de Validación: Proporciona una seguridad final de que el software satisface todos los requerimientos funcionales y de rendimiento. Además, valida los requerimientos establecidos comparándolos con el sistema que ha sido construido. Durante la validación se usan exclusivamente técnicas de prueba de caja negra. (Profesor: López.)

1.1.3 Tipos de Pruebas.

Prueba de Recuperación: Fuerza un fallo del software y verifica que la recuperación se lleva a cabo apropiadamente.

Prueba de Seguridad: Prueba centrada en asegurar que los datos o sistemas que son objetos de prueba, son accedidos sólo por los actores que tienen permiso para hacerlo. Esta prueba es implementada y ejecutada contra varios objetos de prueba.

Prueba de Resistencia: Enfrenta a los programas a situaciones anormales.

Prueba de Rendimiento: Prueba el rendimiento del software en tiempo de ejecución.

Prueba de Instalación: Se centra en asegurar que el sistema de software desarrollado se puede instalar en diferentes configuraciones, hardware y software, bajo condiciones y excepciones, por ejemplo con espacio de disco insuficiente o continuas interrupciones.

Prueba función: Es la prueba centrada en validar las funciones que son objeto de prueba como lo que deben ser, ofreciendo los servicios, métodos o casos de usos requeridos. Esta prueba es implementada y ejecutada contra diferentes objetos de pruebas, incluyendo unidades, unidades integradas, aplicaciones y sistemas.

Se le aplican a todas o solo un subconjunto de pruebas de función. Puede implicar la re-ejecución de cualquier tipo de prueba. Normalmente, esta prueba se lleva a cabo durante cada iteración, ejecutando otra vez la prueba de la iteración anterior.

Prueba de Volumen: Prueba centrada en verificar las habilidades de los objetos de prueba para manejar grandes cantidades de datos, tanto en entrada como en salida o residente en la base de datos. Puede incluir un procedimiento que indique el uso de consultas que devuelvan todo el contenido de la base de datos, o cuando la cantidad de datos de entrada excede a la cantidad establecida de cada campo.

Prueba Usabilidad: Prueba encaminada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.

Prueba de Integridad: Enfocada a la valoración de la robustez (resistencia a fallos).

Prueba de Estructura: Es la prueba enfocada a la valoración y adherencia a su diseño y formación. Este tipo de prueba es hecho a las aplicaciones Web asegurando que todos los enlaces están conectados, el contenido deseado es mostrado y no hay contenido huérfano.

Prueba de Stress: Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible).

Prueba de Performance Profile: Enfocadas a monitorear el tiempo en flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar los cuellos de botellas y los procesos ineficientes.

Prueba de Carga: Es la prueba usada para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema está bajo prueba permanece constante. La variación en carga es simular a la carga de trabajo promedio y con picos que ocurren dentro de tolerancias operacionales normales.

Prueba de Configuración: Es la prueba enfocada en asegurar que funciona en diferentes configuraciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema.

Prueba regresiva: Es la prueba que verifica que un error ha sido corregido y no continúe ocurriendo. (Pressman, y otros, 1990)

1.2 Estado Actual.

Actualmente en el Laboratorio Industrial de Pruebas de Software se ha llevado a cabo el proceso de pruebas de liberación de la siguiente forma. Primeramente se realiza la solicitud de prueba, luego se realiza el análisis a dicha solicitud para definir:

- El grupo de muestreo (GM): Definir muestras y artefactos.
- El grupo de diseño de pruebas (GDP): Lista de chequeo (LCH) y Casos de pruebas (CP). (Análisis y necesidades). Define tipos de pruebas.
- El grupo de herramientas (GH): Define herramientas y tipo de pruebas.
- El grupo de gestión de prueba (GGP): Preparar las condiciones para la gestión de las pruebas. Definir cantidad de probadores por máquina.

Después de analizar lo anteriormente mencionado se elabora el pre-plan de prueba, luego se lleva a cabo la reunión de inicio donde no deben faltar los siguientes invitados:

1. Jefe Proyecto.
2. Asesor facultad, invitados: Vicedecano de producción y Decanos.

En esta reunión de inicio se obtienen los siguientes artefactos.

- Queda definido el plan de pruebas.
- Se firma la minuta de la reunión.
- Se definen los protocolos de comunicación.

Después de realizada esta actividad se llevan a cabo las pruebas exploratorias (PE) donde se tienen en cuenta los criterios de criticidad y pasan a realizarse las pruebas funcionales (PFx), si el software cumple con la calidad requerida se libera el producto, de lo contrario se abortan las pruebas y se realiza una reunión con el jefe del proyecto para mitigar los errores encontrados, volviéndose a realizar la reunión de inicio y consigo el flujo nuevamente. Ver figura 1.

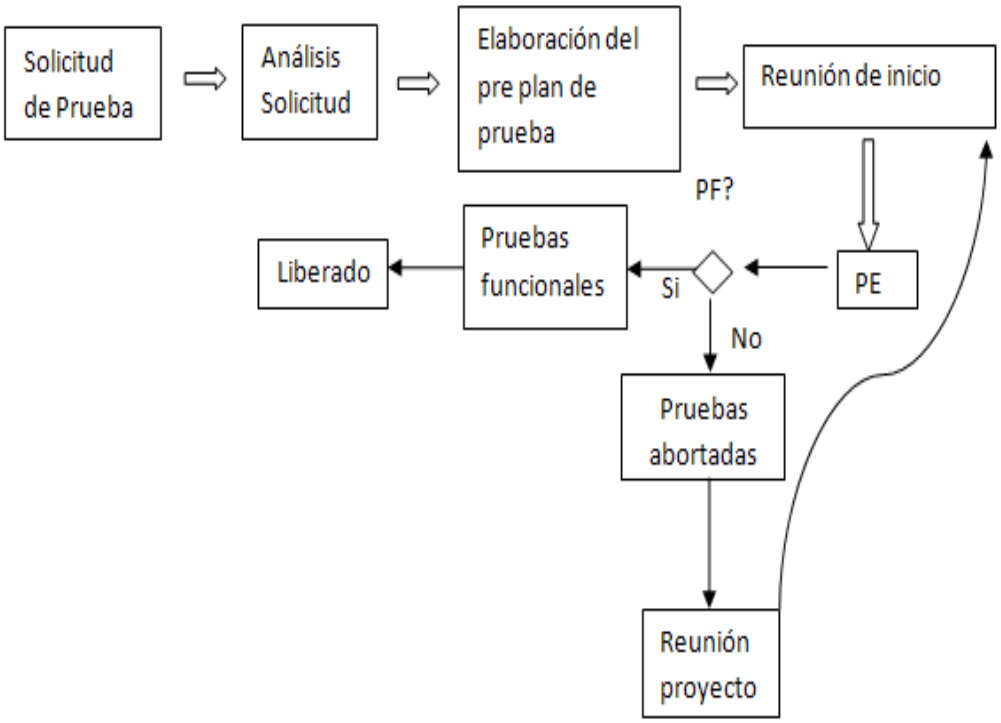


Figura 1: Proceso de pruebas de liberación.

1.3 Clasificación de las No Conformidades en las pruebas de liberación.

La descripción de las clasificaciones de las No Conformidades ayudará a la acelerada gestión de los proyectos y a que el proceso de pruebas de liberación se realice en un tiempo reducido y planificado, sin atentar contra el cronograma de desarrollo y los compromisos con el cliente. Se podrá realizar una evaluación más profunda del software a revisar, se tendrán mejores estadísticas de cuáles son las No Conformidades más comunes en los diferentes tipos de software y se le podrá dar un mejor tratamiento a la No Conformidad en el proceso de liberación del software. (Mares Amézquita, 2007)

1.3.1 ¿Qué es una No Conformidad?

Según lo planteado en la Norma UNE-EN ISO 9001:2000. “Es el incumplimiento de algunos de los requisitos”. (Mares Amézquita, 2007)

¿Qué significa ISO?

El Comité Europeo de Normalización (CEN), a través de su centro de gestión y la ayuda de grupos de trabajo, transpone a rango europeo las normas ISO internacionales, de modo que resultan normas en ISO. Las normas en ISO existen en tres versiones oficiales: alemán, francés e inglés. (Fernández Pereda)

¿Qué significa UNE-EN-ISO?

Las normas UNE-EN- ISO son el resultado de la versión en español de las normas en ISO. Cualquier versión en un idioma diferente al inglés, francés o alemán debe ser realizada por un miembro del Comité Europeo de Normalización (CEN).

En España el organismo nacional de normalización miembro de la red ISO es AENOR (Asociación Española de Normalización). AENOR es quien realiza la versión en español de las normas en ISO, resultando normas UNE-EN ISO. De ahí que las normas españolas se codifiquen de esta manera.

En España existen unas 17.000 normas de calidad que permiten evaluar todas las áreas de actividad de las empresas, sus productos o servicios y sus sistemas. De las 17.000 normas (UNE), unas 10.000 han sido directamente adaptadas de las propias normas europeas (EN). Algunas de estas normas son obligatorias, otras voluntarias. Las más conocidas en la actualidad (por prestigio y resonancia social) son las llamadas normas: UNE-EN-ISO 9000 y UNE-EN-ISO 14000. Ellas pertenecen a la categoría de voluntarias.

Se puede apreciar que una No Conformidad es un fallo en el Sistema de gestión de la Calidad que puede producirse por varias razones, no alcanzar el nivel de aceptación establecido en un determinado indicador y errores en la documentación del Sistema. Se trata de una desviación entre lo que hay escrito (lo que hemos dicho que vamos a hacer) y lo que ha ocurrido (lo que hemos hecho). Este fallo queda registrado en un informe y se establecen las acciones preventivas y correctivas necesarias para arreglar lo que no funcione y evitar que vuelva a ocurrir. (Fernández Pereda)

Las mismas se clasifican de acuerdo al nivel de importancia en:

1. **Las No Conformidades Significativas**: Son aquellas que afectan la calidad del producto o servicio de manera visible, impidiendo o no el cumplimiento de algún requisito.

2. **Las No Conformidades No Significativas**: Son aquellas que resultan menos visibles, que no atentan el cumplimiento de algún requisito.

3. **Las Recomendaciones**: Son aquellas que quedan en función de la apreciación del probador para oportunidades de mejoras del producto o servicio.

Las No Conformidades se clasifican además en función de sus características de acuerdo al tipo de artefacto como:

1. **No Conformidades de Documentación**:

- Formato.
- Error técnico.
- Otros errores.
- Correspondencia con otra documentación.

2. **No Conformidades de Aplicación**:

- Validación.
- Opciones que no funcionan.
- Errores de interfaz.
- Funcionalidad.
- Excepciones.
- Correspondencia de lo implementado con lo documentado.

3. **No Conformidades comunes (para ambos artefactos)**:

- Ortografía.
- Redacción.
- Errores de idioma.

1.4 Control Estadístico de las No Conformidades.

Actualmente en el Laboratorio Industrial de Pruebas de Software se lleva a cabo este proceso de la siguiente manera. En el proceso de pruebas de liberación se generan las No Conformidades clasificadas, estas se recogen de forma cuantitativa en un Excel dentro del expediente de las pruebas de liberación de cada proyecto. Luego se le realiza semanalmente un reporte a la dirección de la universidad sobre el estado en el que se encuentra el proyecto mediante las pruebas de liberación y mensualmente se realiza dicho control estadístico cuantitativo mediante un Excel, donde se registran estas No Conformidades encontradas para así obtener los proyectos que contienen mayor o menor cantidad de errores comunes o los proyectos que contienen estas cantidades de errores pero por iteraciones, así como otros reportes de interés para la universidad. Después de ser liberado el producto, se almacenan estos errores encontrados en un histórico (es un documento donde se registran o se guardan todos los errores encontrados), de forma manual. Ver figura 2.

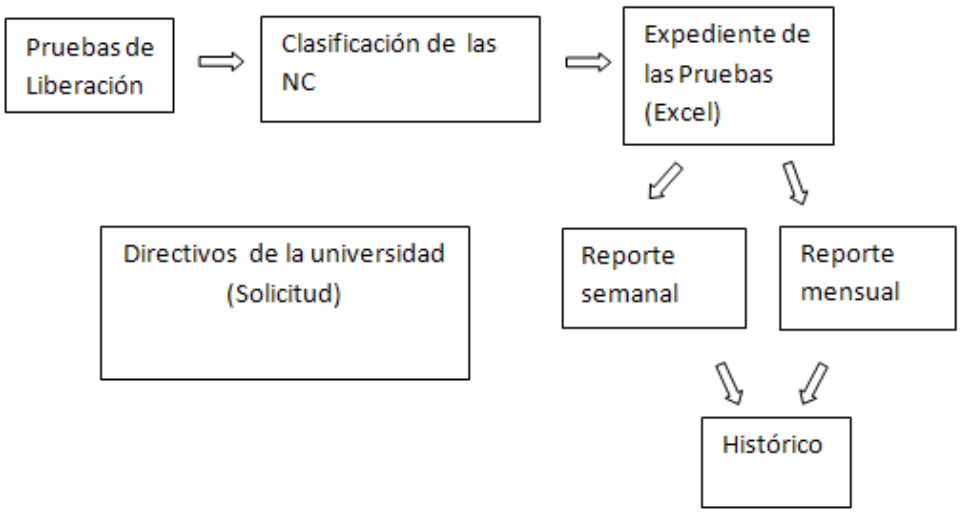


Figura 2: Control Estadístico.

1.5 Aplicaciones que han sido desarrolladas para el Control Estadístico de las No Conformidades.

En el mundo existe un considerable número de organizaciones que cuentan con Laboratorios de Calidad altamente calificados y con herramientas que gestionan las No Conformidades, pero cada uno de estos hace su gestión de forma diferente. Estas herramientas son capaces de generar el control estadístico de las No Conformidades mediante la clasificación, el tipo, y el estado de las mismas. Estas herramientas son necesarias para poder llevar a cabo el control estadístico de las No Conformidades.

1.5.1 Sistema para el Control Estadístico de las No Conformidades.

Dentro de los sistemas que generan el control estadístico de las No Conformidades se encuentra el sistema “Omega 4” desarrollado en los laboratorios de España, el mismo cuenta con potentes herramientas para diversas funciones y dentro de ellas “Orientación hacia la Gestión”, se tiene la herramienta TQM (Total Quality Management), la cual fue creada para la gestión de un modelo de calidad, donde incluyen gestión documental, incidencias, indicadores, auditorias, No Conformidades y mejoras continuas. Es una herramienta específicamente desarrollada para la implantación y seguimiento de modelos de calidad en los laboratorios de análisis clínicos. La herramienta incluye las siguientes funcionalidades: gestión documental de forma periódica, gestión de incidencias y acciones correctivas, gestión del mantenimiento del equipamiento analítico, No Conformidades, auditorías internas y externas, indicadores de calidad automáticos y manuales, implanta medidas de mejoras, lanza mensajes de forma pre-activa, personaliza a todos los usuarios implicados cuando es necesaria su participación en el proceso y gestiona de forma automática los flujos de información y documentación entre los colaboradores del laboratorio. Además la misma es privatizada. (Roche, 2008 - 2009.)

Se tiene además la herramienta TRAC para llevar a cabo el proceso de gestión de las No Conformidades. La misma es utilizada por algunos proyectos pertenecientes a la universidad, ejemplo de ellos se tiene el Cuerpo de Investigaciones Científicas, Penales y Criminalística (CICPC), donde dicha herramienta es utilizada para la gestión de las No Conformidades. Esta herramienta le facilita al probador del proyecto, quien es el encargado de realizar las pruebas de liberación al producto, el poder llevar el control de las No Conformidades según sus diferentes tipos de clasificaciones y necesidades. Con la diferencia de no poder generar un reporte sobre el control estadístico de las No Conformidades, además no permite devolver al mismo tiempo los errores encontrados en cantidades exactas en más

de un proyecto que se esté liberando en ese momento, que es lo que realmente se necesita en el Laboratorio Industrial de Pruebas de Software. Presenta la dificultad de no poder guardar la documentación del trabajo realizado si la computadora presentara algún problema en un momento determinado.

1.6 Tendencias y Tecnologías actuales. Metodología y herramientas utilizadas.

Según las necesidades y las posibilidades del entorno donde se aplicará la solución propuesta, teniendo en cuenta que ya existe una arquitectura definida por el Laboratorio Industrial de Pruebas de Software, se tomó la decisión de trabajar con esa misma metodología de desarrollo y herramientas utilizadas en el laboratorio para así brindar un mejor servicio. Para que la herramienta a desarrollar se pueda integrar a otras que se están implementando.

1.6.1 Metodología utilizada.

El Proceso Unificado de Desarrollo (RUP – Rational Unified Process) es un marco genérico de trabajo que puede especializarse para una gran variedad de sistemas de software, así como para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de actitud y diversos proyectos con diferentes tamaños. El Proceso Unificado posee rasgos fundamentales como: dirigido por Casos de Uso, Centrado en la Arquitectura e Iterativo e Incremental. Se utiliza para proyectos de corta como de larga duración. Brinda una mejor definición sobre el flujo de los procesos para el desarrollo del software. Tiene como objetivo asegurar la producción de software dentro de los plazos y presupuestos predecibles. (Grupo Soluciones, 2007)

1.6.1.1 Lenguaje de Modelado (UML).

Lenguaje Unificado de Modelado por sus siglas en inglés (UML – Unified Modeling Language) es el lenguaje de modelado de sistemas de software orientado a objetos más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, construir y documentar un sistema de software. También es utilizado para modelar todas las especificaciones de un sistema de software. UML se integra con Eclipse, Borland, JBuilder, Oracle para soportar las fases de implementación en el desarrollo de software. El mismo tiene los siguientes rasgos:

- Soporte de Planificación de Caso de Uso.
- Soporte de Análisis Textual.
- Modelo de apoyo entre los diagramas.

- Soporte para la importación de proyecto desde Rational Rose.
- Generación de reportes HTML.
- Generación de reportes PDF. (Zapata, y otros, 2005)

1.6.2 Herramienta CASE (Visual Paradigm).

Se selecciona Visual Paradigm como herramienta CASE (Computer - Aided Systems Engineering) ya que utiliza “UML” como lenguaje de modelado. Esta herramienta es además multiplataforma, provee soporte para la generación de código PHP y para la realización de ingeniería inversa para Java, reduciendo el tiempo de desarrollo del software, también soporta los últimos estándares de anotaciones de Java y UML. Es de licencia gratuita y comercial con versión free. Soporta aplicaciones Web. Genera código para Java y exportación como HTML. Es fácil de instalar y actualizar. (2007)

1.6.3 Dreamweaver

Dreamweaver es la herramienta de diseño de páginas web más avanzada. Los usuarios que lo manejen aunque sean expertos programadores en HTML, siempre se encontrarán en este programa razones para utilizarlo, sobre todo en lo que a productividad se refiere. Cumple además el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, muy fáciles de usar, tales como:

- Hojas de estilos y capas.
- Java Script para crear efectos e interactividades.
- Inserción de archivos multimedia.

Además es un programa que se actualiza con componentes, fabricado por Macromedia, también para la realización de otras acciones más avanzadas. Este es compatible con las principales tecnologías de servidor, por ejemplo, ColdFusion, PHP, ASP, ASP.NET y JSP, para que los desarrolladores que no son muy expertos, puedan dar vida a sus diseños. (Álvarez, 2002)

1.6.4 Lenguaje de Programación (PHP).

El mismo es un lenguaje del lado del servidor, que es utilizado básicamente para generar páginas dinámicas y que facilita de forma sencilla el acceso a diferentes bases de datos. Es utilizado además para generar imágenes, generar PDF. Los programadores Web desarrollan PHP introduciendo nuevas

funciones en las versiones mejoradas del lenguaje. Las mismas poseen muchas ventajas, algunas de ellas son:

- Es libre y abierto (Código fuente disponible y es gratuito).
- Es multiplataforma: inicialmente fue diseñado para entornos UNIX por lo que ofrece más prestaciones en este sistema operativo, pero es perfectamente compatible con Windows.
- Además presenta una compatibilidad total con Apache. (García, 1998 – 2004 (última modificación en el 2004.))

1.6.5 Servidor (Servidor Apache).

El servidor Apache es un software que está estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Estos módulos se pueden clasificar en tres categorías:

- **Módulos Base**: Módulo con las funciones básicas del Apache.
- **Módulos Multiproceso**: Estos son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones.
- **Módulos Adicionales**: Cualquier otro módulo que le añada una funcionalidad al servidor. Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código. El resto de las funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software. El servidor HTTP Apache es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en el HTTP 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que originalmente Apache consistía solamente en un conjunto de parches a aplicar al servidor. Era, en inglés, a patchy server (un servidor "parcheado"). Apache presenta entre otras características, mensajes de error altamente configurables, bases de datos de

autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. Presenta las siguientes características:

1. Funciona sobre muchas plataformas (ejemplo de ellas: Unix, Linux, Vms, Win32, OS2).

2. Módulos cargados dinámicamente.

2.1 CGI, Perl (ejemplo: Formularios, diccionarios en línea, entre otros) Ejemplo CGI.

2.1.1 Ejemplo Perl (Php3 + Bases de datos).

3. SSL: transacciones seguras.

4. Alto desempeño. (McCool, 1995)

1.6.6 Gestor Base Datos (PostgreSQL)

¿Qué es PostgreSQL?

PostgreSQL es un avanzado sistema de bases de datos relacionales basado en Open Source. Esto quiere decir que el código fuente del programa está disponible a cualquier persona libre de cargos directos, permitiendo a cualquiera colaborar con el desarrollo del proyecto o modificar el sistema para ajustarlo a sus necesidades.

Un sistema de base de datos relacionales es un sistema que permite la manipulación de acuerdo con las reglas del álgebra relacional. Los datos se almacenan en tablas de columnas y renglones. Con el uso de llaves, esas tablas se pueden relacionar unas con otras.

Ideas Básicas acerca del funcionamiento.

En la jerarquía de bases de datos, PostgreSQL usa el modelo cliente/servidor. Una sesión en PostgreSQL consiste en ejecución de los siguientes procesos:

- El servidor, que maneja archivos de bases de datos, acepta conexiones a las aplicaciones cliente y realiza acciones en la base de datos. El programa servidor de bases de datos se conoce como postmaster.
- La aplicación cliente, necesita realizar operaciones en la base de datos. Las aplicaciones cliente pueden ser de la más diversa naturaleza: pueden ser aplicaciones de texto en una consola, aplicaciones gráficas, un servidor web que accede a la base de datos para mostrar una página, o herramientas especializadas de mantenimiento de bases de datos.

Como es habitual en las aplicaciones cliente/servidor, el cliente y el servidor pueden estar en diferentes máquinas. En este caso, estos se comunican sobre una conexión de red TCP/IP.

El servidor PostgreSQL puede manejar múltiples conexiones concurrentes de los clientes. Para esto inicia un nuevo proceso (”fork”) para cada conexión llamado backend. Con esto, el cliente y el nuevo proceso del servidor se comunican sin la intervención del proceso original del postmaster. Así, el postmaster está siempre ejecutándose, esperando por conexiones de parte de los clientes.

¿Qué es una base de datos relacional?

Una base de datos relacional desde el punto de vista del usuario es una colección de tablas interrelacionadas que permiten almacenar información para que esta pueda ser utilizada posteriormente, y se basa en el modelo de datos relacional para la manipulación de las tablas, el que a su vez se basa en elementos de la teoría de conjuntos para establecer las relaciones.

Una lista breve de características y técnicas que PostgreSQL ofrece:

- Integridad referencial.
- Replicación (soluciones comerciales y no comerciales) que permiten la duplicación de bases de datos maestras en múltiples sitios de réplica, por ejemplo:
    1. Reglas.
    2. Vistas.
    3. Secuencias.
    4. Herencia.
    5. Una API abierta.
    6. Procedimientos almacenados.
    7. Índices parciales y funcionales.
    8. Autentificación Kerberos nativa.
    9. Herramientas para generar SQL portable para compartir con otros sistemas compatibles con SQL.
    10. Sistema de tipos de datos extensible para proveer tipos de datos definidos por el usuario, y rápido desarrollo de nuevos tipos.
    11. Funciones de compatibilidad para ayudar en la transición desde otros sistemas menos compatibles con SQL.

PostgreSQL ofrece muchas ventajas con respecto a otros sistemas de bases de datos:

1. PostgreSQL se caracteriza por ser un sistema estable, de alto rendimiento, gran flexibilidad ya que funciona en la mayoría de los sistemas Unix, además tiene características que permiten extender fácilmente el sistema.
2. PostgreSQL puede ser integrada al ambiente Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes.
3. Permite desarrollar o migrar aplicaciones desde Access, Visual Basic, FoxPro, Visual FoxPro, C/C++ Visual C/C++, Delphi, etc., para que utilicen a PostgreSQL como servidor de BD.
4. Por lo anterior PostgreSQL se convierte en una gran alternativa al momento de decidirse por un sistema de bases de datos.

Instalación ilimitada.

Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. PostgreSQL, aún no ha podido ser demandado por violar acuerdos de licencia, puesto que no existe un costo asociado a la licencia del software.

Esto tiene varias ventajas adicionales:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar el cumplimiento de la licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.

Ahorros considerables en costos de operación.

Este software ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características, estabilidad y rendimiento.

Estabilidad y confiabilidad legendarias.

En contraste a muchos sistemas de bases de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.

Extensible.

El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.

Multiplataforma.

PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows está actualmente en estado beta de pruebas.

Diseñado para ambientes de alto volumen.

PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mucha mejor respuesta en ambientes de grandes volúmenes. Los principales proveedores de sistemas de bases de datos comerciales usan también esta tecnología, por las mismas razones.

Herramientas gráficas de diseño y administración de bases de datos.

Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin y pgAccess). (Grupo de Trabajo, 2002. (última modificación: 2005 (2010).))

1.6.7 Herramienta Integradora (WAMP 5).

W.A.M.P. son las siglas de Windows + Apache + MySQL + PHP, es decir, un instalador paso a paso que incluye en un sólo paquete estas tres tecnologías para montar su propio servidor bajo Windows. El software que se instala con WAMP5 contiene los siguientes servidores y programas: Apache 1.3.31. Es el servidor de páginas web más extendido del mercado. Aunque la última versión de este servidor es Apache 2, se instala una versión anterior que resulta más estable. Existe un Add-on que permite sustituir la versión 1.3.31 de Apache por la última versión, ejemplo:

- PHP5: El motor renovado del lenguaje.
- MySQL: La base de datos más extendida para utilizar con PHP.
- PHPmyadmin: Un software que permite administrar una base de datos a través de una interfaz web.
- SQLitemanager: Un sistema para administrar una base de datos a partir de sentencias SQL.

1.6.8 Arquitectura (Cliente-Servidor).

Con la proliferación de ordenadores personales de bajo coste en el mercado, los recursos de sistemas de información existentes en cualquier organización se pueden distribuir entre ordenadores de diferentes tipos: ordenadores personales de gama baja, media y alta, estaciones de trabajo, miniordenadores o incluso grandes ordenadores. El concepto de cliente/servidor proporciona una forma eficiente de utilizar todos estos recursos de máquina de tal forma que la seguridad y fiabilidad que proporcionan los entornos mainframe se traspasa a la red de área local. A esto hay que añadir la ventaja de la potencia y simplicidad de los ordenadores personales.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario. Los clientes realizan generalmente funciones como:

- Manejo de la interfaz de usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.

Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo. Entre las principales características de la arquitectura cliente/servidor se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.

- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Ventajas

1. Aumento de la productividad:

1.1 Los usuarios pueden utilizar herramientas que le son familiares, como hojas de cálculo y herramientas de acceso a bases de datos.

1.2 Mediante la integración de las aplicaciones cliente/servidor con las aplicaciones personales de uso habitual, los usuarios pueden construir soluciones particularizadas que se ajusten a sus necesidades cambiantes.

1.3 Una interfaz gráfica de usuario consistente reduce el tiempo de aprendizaje de las aplicaciones.

1.4 Menores costes de operación.

1.5 Permiten un mejor aprovechamiento de los sistemas existentes, protegiendo la inversión. Por ejemplo, la compartición de servidores (habitualmente caros) y dispositivos periféricos (como impresoras) entre máquinas clientes permite un mejor rendimiento del conjunto.

1.6 Proporcionan un mejor acceso a los datos. La interfaz de usuario ofrece una forma homogénea de ver el sistema, independientemente de los cambios o actualizaciones que se produzcan en él y de la ubicación de la información.

1.7 El movimiento de funciones desde un ordenador central hacia servidores o clientes locales origina el desplazamiento de los costes de ese proceso hacia máquinas más pequeñas y por tanto, más baratas.

2. Mejora en el rendimiento de la red:

2.1 Las arquitecturas cliente/servidor eliminan la necesidad de mover grandes bloques de información por la red hacia los ordenadores personales o estaciones de trabajo para su proceso.

2.2 Los servidores controlan los datos, procesan peticiones y después transfieren sólo los datos requeridos a la máquina cliente. Luego, la máquina cliente presenta los datos al usuario mediante interfaces amigables.

2.3 Esto reduce el tráfico de la red, lo que facilita que pueda soportar un mayor número de usuarios. Tanto el cliente como el servidor pueden escalarse para ajustarse a las necesidades de las aplicaciones.

2.4 En una arquitectura como esta, los clientes y los servidores son independientes los unos de los otros con lo que pueden renovarse para aumentar sus funciones y capacidad de forma independiente, sin afectar al resto del sistema.

2.5 Permite además centralizar el control de sistemas que estaban descentralizados, como por ejemplo la gestión de los ordenadores personales que antes estuvieran aislados. (Góngora Rodríguez, y otros, 2007)

1.6.9 Symfony

El framework Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. Symfony está desarrollado completamente con PHP5, por lo cual es completamente orientado a objetos. Ha sido probado en numerosos proyectos reales y se utiliza en sitios Web de comercio electrónico de primer nivel. Vale destacar que Symfony es compatible con la mayoría de los sistemas gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Además implementa la mayoría de las mejores prácticas y patrones de diseño para la programación Web.

Principales características:

1. Fácil de instalar y configurar en la mayoría de plataformas.
2. Independiente del sistema gestor de bases de datos.
3. Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
4. Sigue la mayoría de mejores prácticas y patrones de diseño para la Web.
5. Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.

6. Fácil de extender, lo que permite su integración con las librerías de otros fabricantes. Por todas estas características ya mencionadas y el estudio realizado, se decidió seleccionar a Symfony, a pesar de que Zend también tiene sus cualidades como Framework, pero posee un inconveniente, no es libre. (Zaninotto Potencier, y otros)

Conclusiones del Capítulo 1.

En este capítulo se definió todo el marco teórico, así como darle seguimiento a los procesos claves definidos durante las pruebas que se le realizarán a un determinado producto, desarrollado en la universidad en el Laboratorio Industrial de Pruebas de Software. En el LIPS se tiene definida una arquitectura, la cual utiliza como metodología de desarrollo RUP dando una mejor definición de los procesos para el desarrollo de software, sin importar el tamaño que este posea. Se utiliza además como lenguaje para modelar los procesos definidos para el desarrollo del producto UML. Se utiliza también la herramienta case Visual Paradigm para llevar a cabo el modelado de esos procesos, brindando soporte para la utilización de códigos PHP; permitiendo el completo desarrollo de aquellas páginas dinámicas que conforman los módulos que les darán vida a la herramienta a desarrollar .

Capítulo 2: Características del Sistema.

Introducción

En este capítulo se dará una breve descripción de los roles que desarrollan dichos procesos los cuales son (el estado de los proyectos y el control estadístico de las No Conformidades), así como de otras personas que puedan estar involucradas. Se definirá además la información manejada y la propuesta del sistema. Se identificarán los requisitos funcionales y no funcionales, los casos de uso con sus descripciones, los diagramas de actividades y el modelo de objeto.

2.1 Modelo del negocio.

Se utilizará cómo flujo de trabajo el Modelamiento del Negocio, técnica para reflejar cómo se lleva a cabo el negocio según la especificación de los requisitos definidos para darle soporte al negocio. También define los modelos previos para el desarrollo de un software, así como comprender la estructura y la dinámica de la organización en la cual se va a implementar el sistema, comprender los problemas actuales de la organización e identificar las mejoras potenciales, asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización, además de derivar los requerimientos del sistema que va a soportar la organización. Permite obtener una visión de la organización que le permita definir los procesos, roles y responsabilidades de la organización en los modelos de casos de uso del negocio y de objetos para lograr los propósitos anteriores.

La descripción del negocio propuesto en detalle tendrá entre sus actividades principales la identificación de los procesos del negocio, así como la redacción y especificación de los casos de uso, la identificación de los roles y entidades que ejecutan las realizaciones de los casos de uso del negocio y detallar la definición de las entidades y las responsabilidades de los roles del negocio.

Por tal motivo se seleccionó el modelo del negocio y no el modelo del dominio puesto que es necesario hacer un modelo completo del negocio para de esta forma entender mejor lo que se pretende realizar. Además se especifica con claridad las responsabilidades de las personas que ejecutan las actividades, ya que esto no lo permite el modelo del dominio.

2.1.1 Actor del negocio.

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos, con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados. Además todo lo que interacciona con el ambiente del negocio se modela con actores. Cada actor humano expresa un rol, no una persona específica. El mismo modela algo fuera del negocio, se involucra con un caso de uso, al menos como regla, también tiene una descripción y un nombre que explica su rol en relación al negocio.

Tabla 1: Descripción de los actores.

Actor	Descripción
Directivos de la universidad	Le solicitan al jefe del laboratorio de pruebas, el estado de los proyectos que están siendo revisados hasta ese momento. Ellos pueden ser los decanos de cualquier facultad, vicedecanos, el rector, vicerrector, etc.
Solicitante	Asesor de calidad de la facultad, vicedecanos de producción o toda aquella persona que solicite la revisión del software.

2.1.2 Trabajadores del negocio.

Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

Tabla 2: Descripción de los trabajadores.

Trabajador	Descripción
Jefe del Laboratorio	Encargado de elaborar un reporte general sobre el estado de todos los proyectos que se están revisando, para luego enviárselo a los directivos de la universidad. Planifica los recursos del laboratorio, repartir el trabajo, supervisar la gestión de las No Conformidades y realiza los reportes de seguimiento de los recursos.
Especialista de Calidad	Encargado de recoger en un reporte el estado de los proyectos que estén revisando en ese momento, para luego enviárselo al jefe del laboratorio. Encargado de insertar las pruebas a realizar y las métricas.
Probador	Persona que realiza las pruebas de software.
Desarrollador	Ve las No Conformidades emitidas y las responde.

2.1.3 Diagrama de Casos de Uso del negocio.

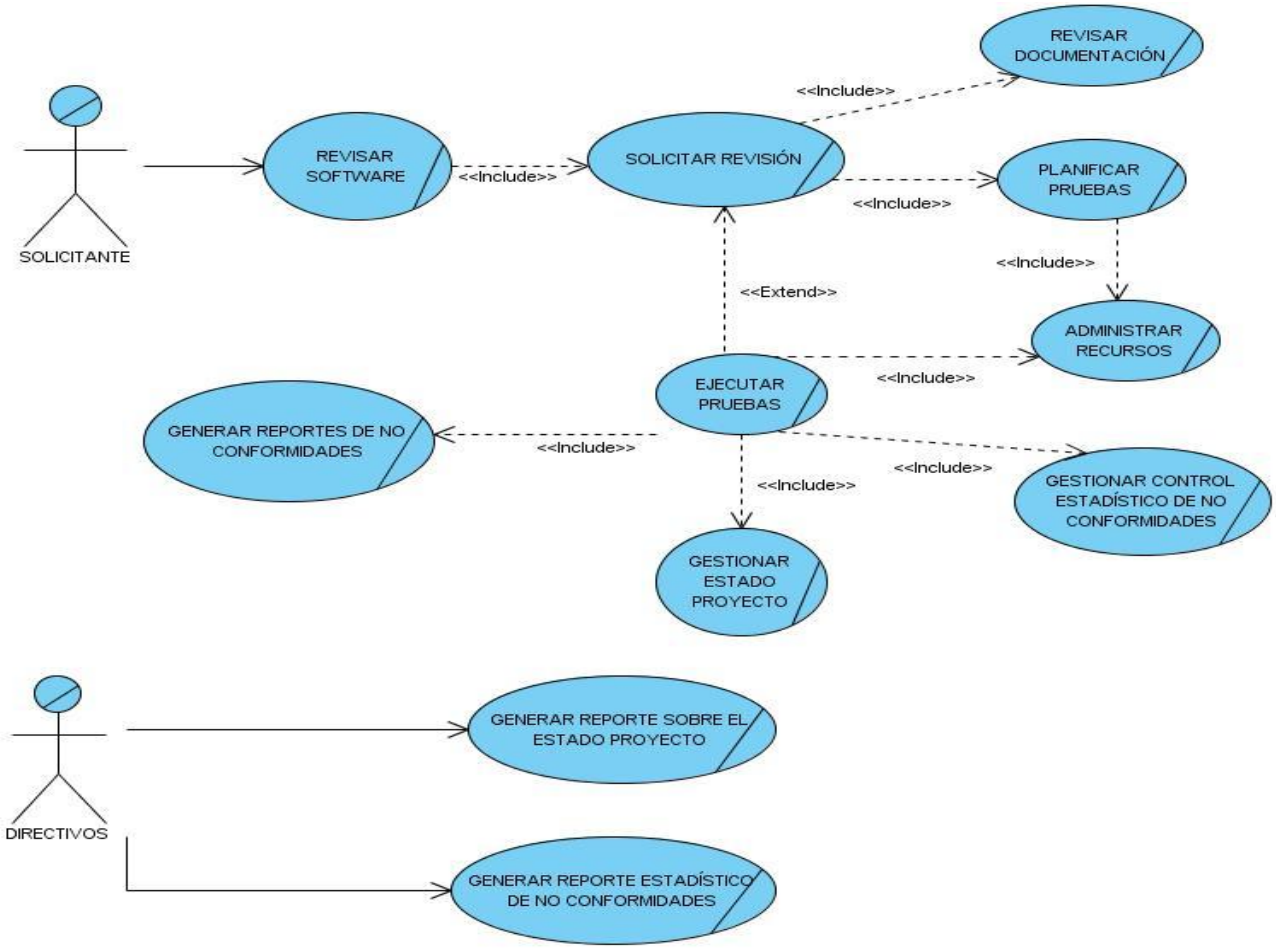


Figura 3: Diagrama de Casos de Uso del negocio.

2.1.4 Identificar los Casos de Uso del negocio más significativos.

- Generar Reporte del Estado de los Proyectos.
- Generar Reporte del Control Estadístico de las No Conformidades.
- Gestionar el Control Estadístico de las No Conformidades.
- Gestionar Estado de los Proyectos.

2.1.5 Descripción del negocio: CU1Generar Reporte del Estado de los Proyectos.

Los directivos de la Universidad le solicitan al jefe del laboratorio de pruebas el estado en el que se encuentran los proyectos que se están revisando en ese momento. Luego el jefe del laboratorio le solicita a los especialistas un reporte del estado actual en el que se encuentran los

proyectos hasta ese momento. Los especialistas recogen en un reporte el estado de cada uno de los proyectos que se estén revisando en ese momento, para luego enviárselo al jefe del laboratorio. El mismo elabora un reporte general, de todos los proyectos que se están revisando hasta ese momento para mandárselo a los directivos de la universidad cada semana.

2.1.6 Descripción del negocio: CU2 Generar Reporte del Control Estadístico de las No Conformidades.

Los directivos de la Universidad le solicitan al jefe del laboratorio de pruebas el reporte estadístico de las No Conformidades de aquellos proyectos que se estén revisando en ese momento. Luego el jefe del laboratorio, le solicita a los especialistas que le hagan llegar los reportes estadísticos de las No Conformidades encontradas en los proyectos que se hayan revisado, para determinar la cantidad de No Conformidades encontradas en los mismos por iteraciones según sus clasificaciones mediante la revisión. Después los especialistas recogen en un reporte de forma manual la cantidad de estas No Conformidades encontradas en cada uno de los proyectos que se revisaron, para luego enviárselo al jefe del laboratorio. El mismo elabora un reporte de forma general, de todos los proyectos que se revisaron hasta ese momento, para luego enviárselo a los directivos de la Universidad. Esto se realiza de forma mensual, aunque hay que tener los reportes al día porque los directivos de la Universidad pueden pedirlo en cualquier momento para tener un control de cómo se encuentra el estado del proyecto según las No Conformidades encontradas por su clasificación y en la iteración que se obtuvo. Adquiriéndose como resultado el proyecto que mayor cantidad de No Conformidades tuvo hasta ese entonces.

2.1.7 Descripción del negocio: CU3 Gestionar el Control Estadístico de las No Conformidades.

Durante el proceso normal de pruebas, cada vez que se termina una iteración, los especialistas de calidad al frente de las pruebas de cada proyecto deben de recoger los datos por cada artefacto que estén liberando, así como la cantidad de No Conformidades encontradas según la clasificación del artefacto y la cantidad de No Conformidades según la clasificación de las mismas. Esto se almacena en un Excel en el expediente de las pruebas de liberación de cada proyecto para después realizar los diferentes reportes que se soliciten por parte de los directivos de la universidad o para mejorar el proceso de pruebas de liberación para los proyectos a revisar.

2.1.8 Descripción del negocio: CU4 Gestionar Estado de los Proyectos.

Durante el proceso de pruebas los diferentes artefactos a revisar pasan por diferentes estados, los especialistas de calidad al frente de las pruebas, deben actualizar el estado de los proyectos que están revisando cada vez que los artefactos pasen de una iteración a otra, para así mantener informados a todos los involucrados en las pruebas de liberación y a los miembros de los proyectos que se estén revisando.

2.1.9 Diagrama de Actividades: "Generar Reporte del Estado de los Proyectos".

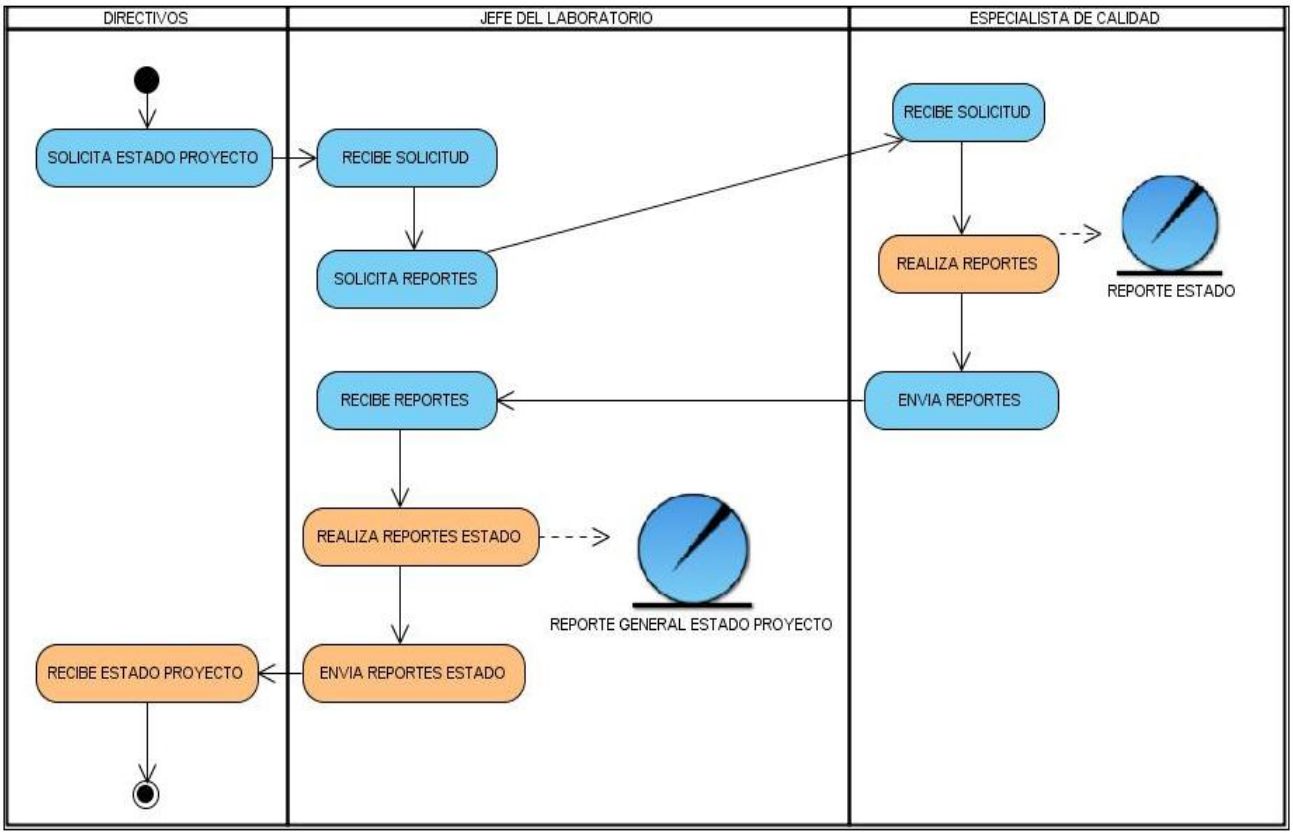


Figura 4: Diagrama de Actividades: "Generar Reporte del Estado de los Proyectos"

2.1.10 Diagrama de Actividades: "Generar Reporte del Control Estadístico de las No Conformidades".

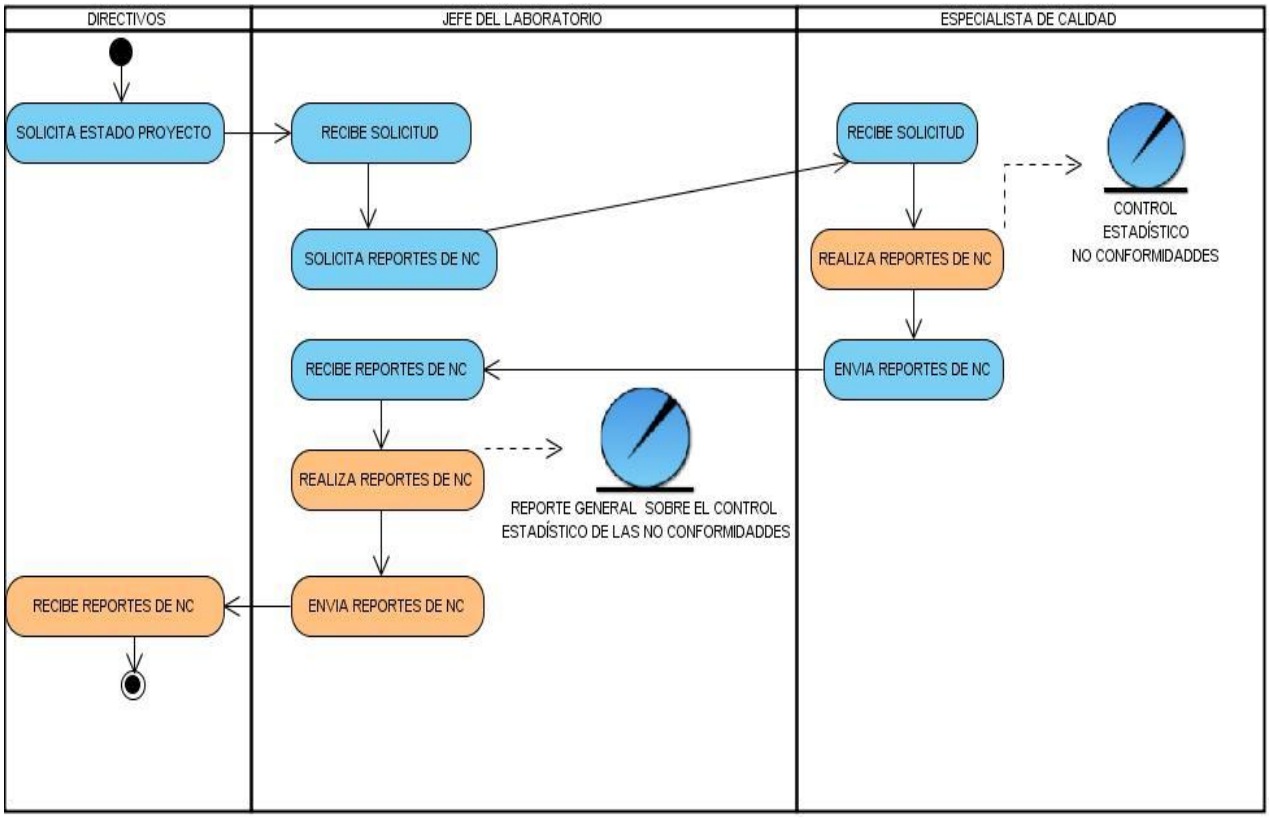


Figura 5: Diagrama de Actividades: "Generar Reporte del Control Estadístico de las No Conformidades".

2.1.11 Diagrama de Actividades: "Gestionar el Control Estadístico de las No Conformidades".

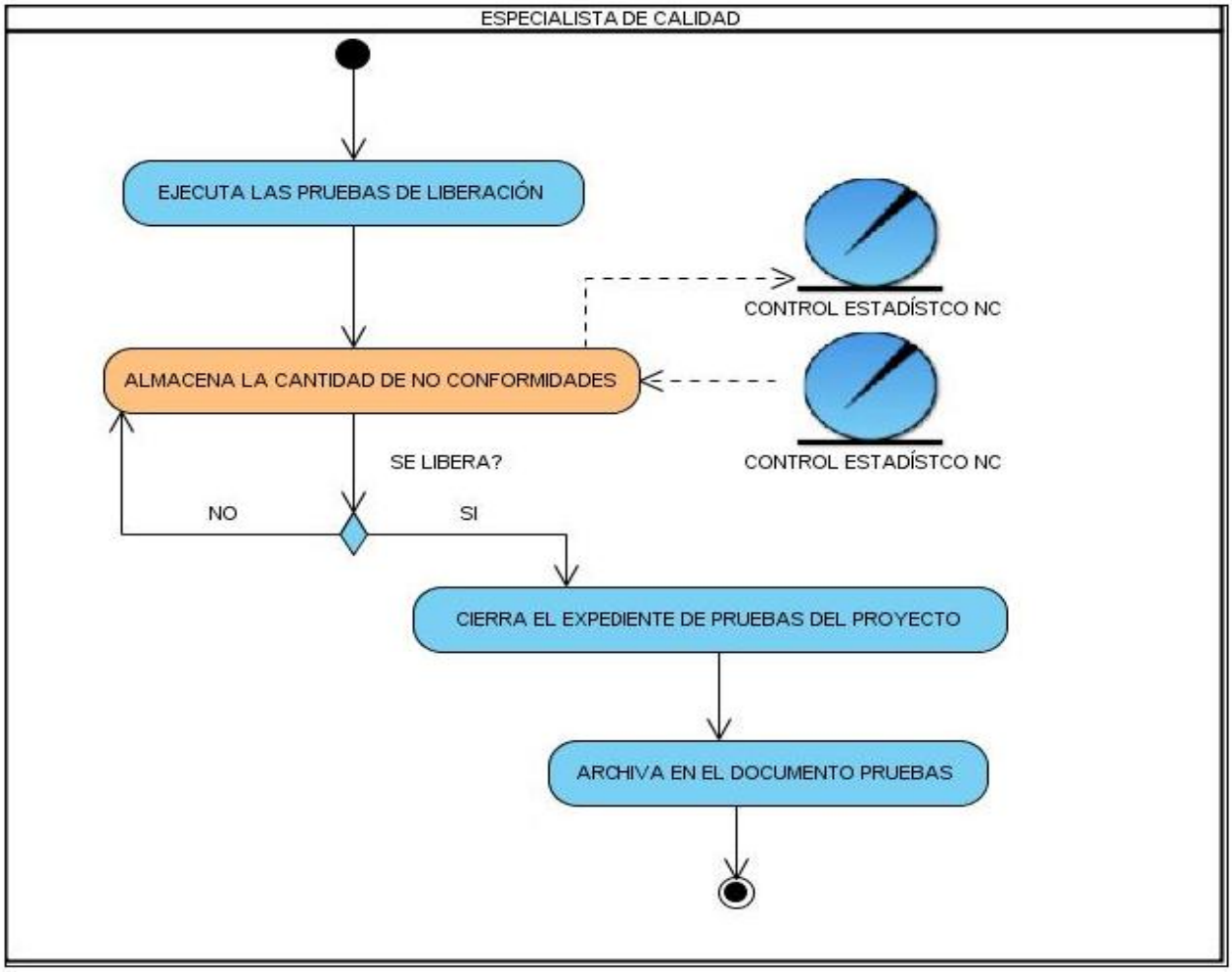


Figura 6: Diagrama de Actividades: "Gestionar el Control Estadístico de las No Conformidades".

2.1.12 Diagrama de Actividades: “Gestionar Estado del Proyecto”.

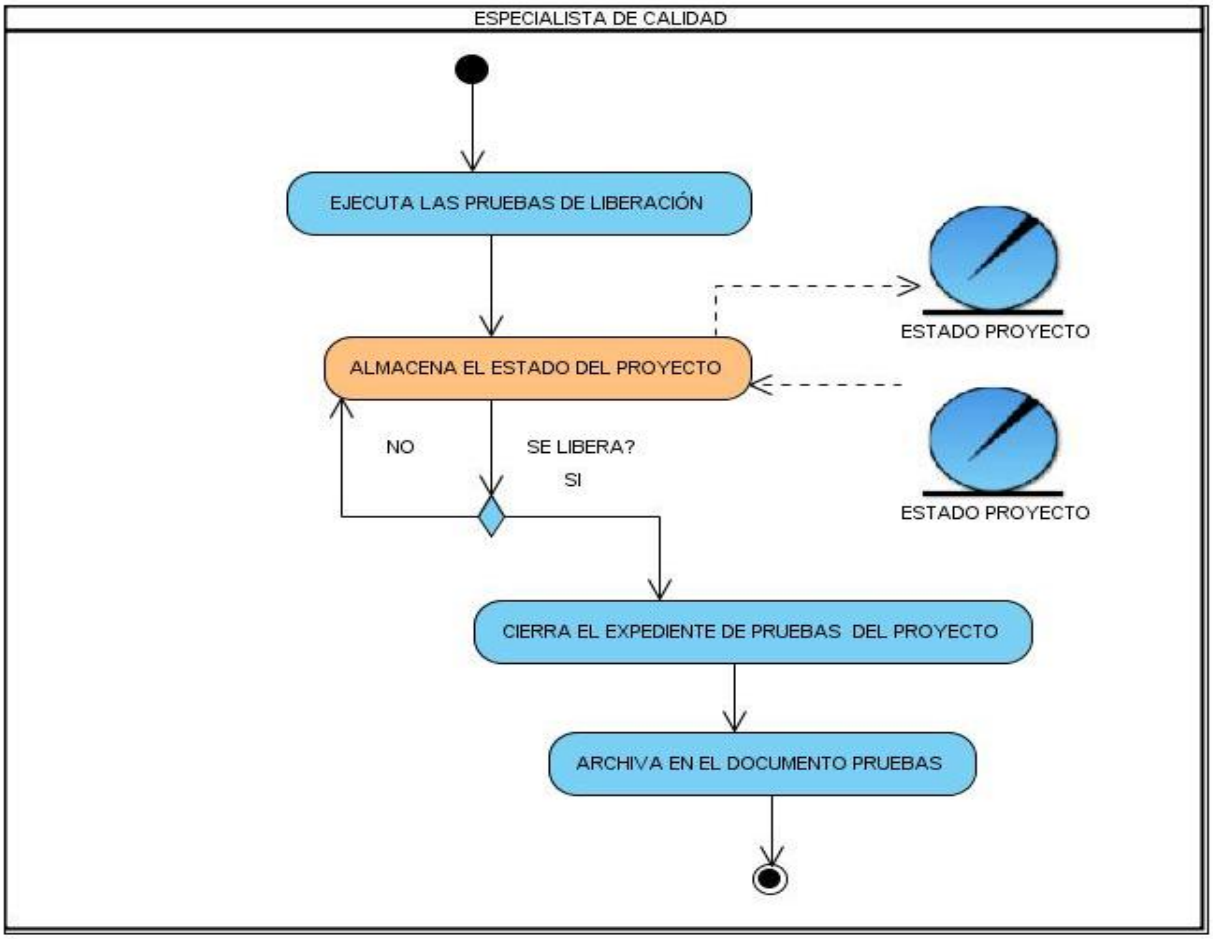


Figura 7: Diagrama de Actividades: “Gestionar Estado del Proyecto”.

2.1.13 Diagrama de Clases del Modelo de Objetos.

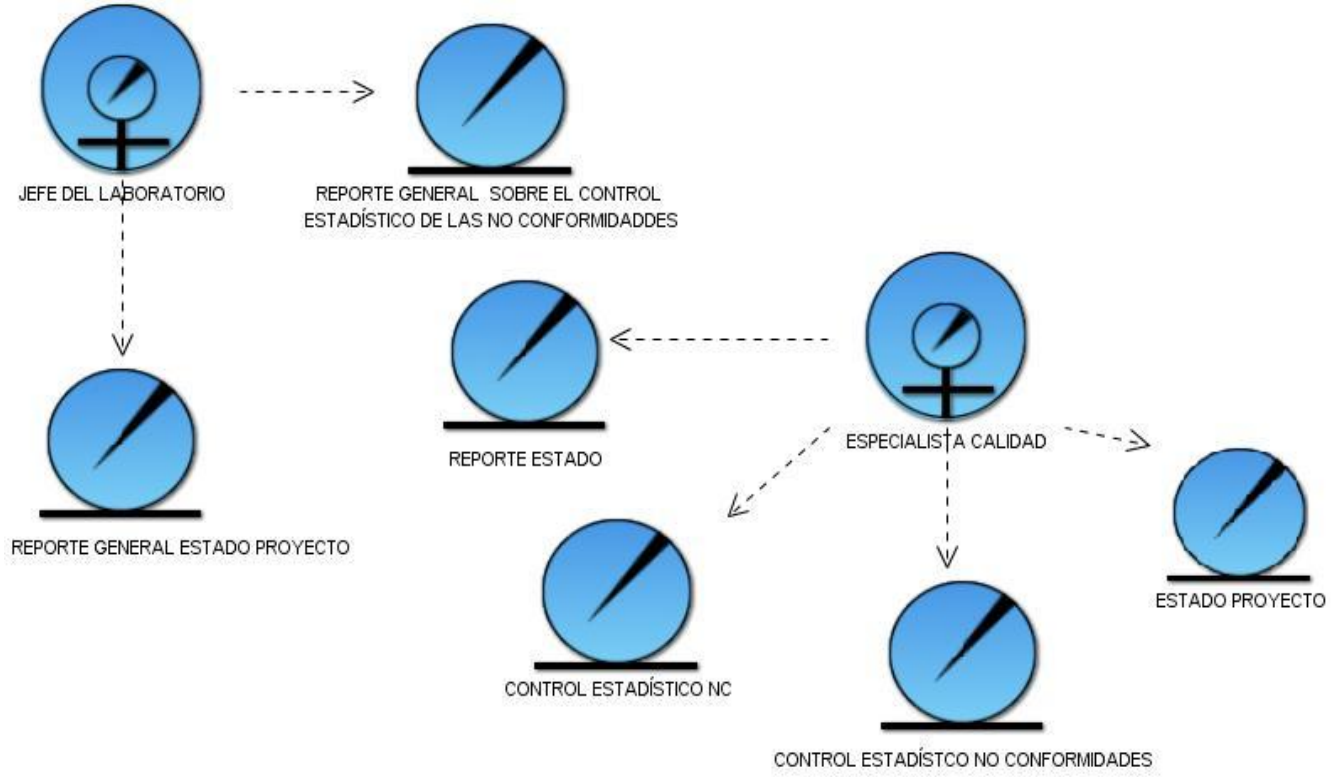


Figura 8: Diagrama de Clases del Modelo de Objetos.

Después de definir los **Casos de Uso del Negocio** se determinaron los requisitos funcionales y no funcionales**:**

2.2 Definición de Requisitos Funcionales.

Los requerimientos funcionales son aquellos requisitos que, desde el punto de vista de las necesidades del usuario, son capacidades o condiciones que debe cumplir el sistema y que están fuertemente ligados a las opciones del programa. Para cumplir con los objetivos propuestos se prevé que el sistema tenga las siguientes funcionalidades:

RF 1: Gestionar Proyecto

1.1 Adicionar proyecto

1.1.1 Nombre del proyecto

1.1.2 Nombre de los módulos

1.1.3 Iteraciones

1.1.4 Revisiones

1.2 Modificar proyecto

1.3 Eliminar proyecto

1.4 Visualizar proyecto

RF 2: Gestionar Clasificación de las No Conformidades

2.1 Adicionar cantidad clasificación de las No Conformidades

2.1.1 Nombre de la clasificación

2.1.2 Iteración

2.1.3 Revisión

2.2 Modificar cantidad clasificación de las No Conformidades

2.3 Eliminar cantidad clasificación de las No Conformidades

2.4 Visualizar cantidad clasificación de las No Conformidades

RF 3: Generar Control Estadístico Cuantitativo de las No Conformidades

3.1 Por tipo de proyecto

3.2 Por tipo de artefacto

3.3 Cantidad de no conformidades por iteraciones

3.4 Cantidad de no conformidades por su clasificación

RF 4: Imprimir los Reportes del Control Estadístico Cuantitativo de las No Conformidades

RF 5: Gestionar Estado de los Proyectos

5.1 Seleccionar estado de los proyectos

5.2 Seleccionar el estado de los módulos

5.3 Insertar artefacto

RF 6: Configuración

6.1 Adicionar nombre de las clasificaciones de las No Conformidades

6.2 Modificar nombre de las clasificaciones de las No Conformidades

6.3 Eliminar nombre de las clasificaciones de las No Conformidades

6.4 Visualizar nombre de las clasificaciones de las No Conformidades

6.5 Adicionar tipo de proyecto

6.6 Modificar tipo de proyecto

6.7 Eliminar tipo de proyecto

6.8 Visualizar tipo de proyecto

6.9 Adicionar tipo de artefacto

6.10 Modificar tipo de artefacto

6.11 Eliminar tipo de artefacto

6.12 Visualizar tipo de artefacto

6.13 Adicionar estado del proyecto

6.14 Modificar estado del proyecto

6.15 Eliminar estado del proyecto

6.16 Visualizar estado del proyecto

RF 7: Autenticar Usuario

RF 8: Gestionar Roles

8.1 Adicionar roles

8.2 Modificar roles

8.3 Eliminar roles

8.4 Visualizar roles

RF 9: Gestionar Usuario

9.1 Adicionar usuario

9.2 Modificar usuario

9.3 Eliminar usuario

9.4 Visualizar Usuario

2.3 Definición de Requisitos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades son las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto, donde normalmente están vinculados a requisitos funcionales, es decir una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

1. Apariencia o interfaz externa

El sistema debe poseer una interfaz Web sencilla, lo más atractiva, amigable y clara posible para el usuario, además su funcionamiento debe ser de fácil comprensión.

2. Usabilidad

El sistema solo puede ser utilizado por aquellas personas que de una forma u otra se encuentren relacionadas con las pruebas de liberación de los proyectos productivos de la Universidad de las

Ciencias Informáticas, en el Laboratorio Industrial de Pruebas de Software. Por ejemplo el especialista de calidad y el jefe del laboratorio.

3. Rendimiento

Se van obtener buenos y esperados resultados con poco trabajo. Debido a que si se trata de una aplicación cliente/servidor la misma debe ser eficiente, con capacidad adecuada de procesamiento y cálculo, así como requiere de un tiempo de respuesta relativamente pequeño.

4. Portabilidad

El sistema debe ser multiplataforma.

5. Seguridad

Permitir mantener la seguridad del sistema. Para poder acceder al sistema el usuario deberá estar registrado en la aplicación. Los usuarios serán creados en dependencia del rol que desempeñen. Se debe garantizar que sólo posean acceso a la información con derecho a ver o manipular según el rol.

6. Software

El software debe poseer métodos de instalación rápidos, debe facilitar también la reinstalación en caso de contingencia. En el lado del Cliente debe existir un navegador que soporte Adobe Reader. En el lado del Servidor debe estar instalado PHP 5.0 o superior a este y cómo gestor de base de datos PostgreSQL.

7. Hardware

Para el desarrollo y ejecución de esta aplicación se necesita conexión a la red local, por lo que se requiere tarjeta de red. Además es necesario contar con una impresora para poder imprimir los diferentes tipos de reportes. Pentium IV, 239 GHz, 249 MB de RAM.

2.4 Actores del Sistema a Automatizar.

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato para actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema. Los actores del sistema se clasifican por:

- No son parte de él.
- Pueden intercambiar información con él.
- Pueden ser un recipiente pasivo de información.

- Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado.

Tabla 3: Descripción de actores del sistema a automatizar.

Actor	Descripción
Usuario del sistema	Rol que representa a los usuarios del sistema que se han autenticado y pueden acceder a los recursos que le son permitidos.
Jefe del Laboratorio	Encargado de elaborar los reportes de forma general sobre el estado de todos los proyectos que se están revisando, para luego enviárselos a los directivos de la universidad.
Especialista de calidad	Encargado de dirigir las pruebas de liberación que se le realicen a los proyectos, así como de actualizar el estado de los mismos y recoger la clasificación de las No Conformidades.
Administrador	Este rol constituye una generalización del Usuario del sistema. Encargado del mantenimiento del sistema así como de gestionar todo el proceso de permisos a los usuarios que acceden al mismo.

2.5 Descripción del Diagrama de los Casos de Uso del Sistema.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores. Cada caso de uso debe comunicarse con al menos un actor, si no aparece ningún actor que se comunique con un caso de uso esto indica error en el modelo de caso de uso o en los requisitos planteados. Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, así como el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

2.6 Diagrama de Casos de Uso del Sistema.

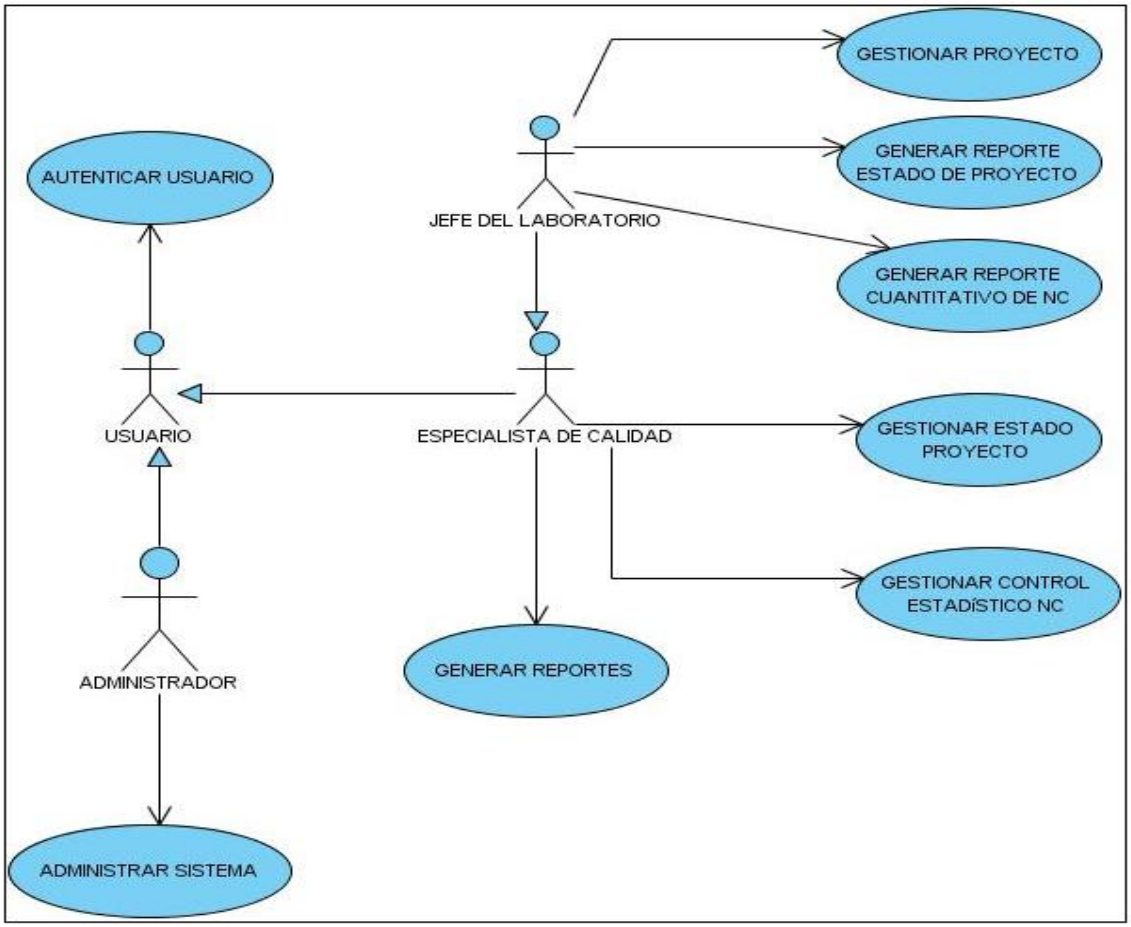


Figura 9: Diagrama de Casos de Uso del Sistema.

2.7 Descripción de los Casos de Uso del Sistema.

Tabla 4: Descripción del Caso de Uso Gestionar Proyecto.

Caso de Uso:	Gestionar Proyecto.
Actores:	Jefe del laboratorio.
Resumen:	Este caso de uso se inicia cuando el jefe del laboratorio distribuye entre los especialista de calidad los proyectos que serán llevados a pruebas antes de su liberación para obtener el estado de los mismos: - Insertar el estado de los proyectos. - Modificar el estado de los proyectos. - Visualizar el estado de los proyectos.

	- Eliminar el estado de los proyectos.
Precondiciones:	Deben de existir los proyectos para poder distribuirlos.
Referencias:	RF1.
Prioridad:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción de gestionar proyecto.	2. El sistema muestra la interfaz para brindar la posibilidad de permitirle adicionar el o los proyectos que se van a revisar.
3. El actor selecciona la opción para asignar los proyectos que le son otorgados a cada Especialista.	4. El sistema inserta el proyecto. Muestra un mensaje de información “Debe llenar todos los campos”.
6. El actor selecciona una de las siguientes opciones: - Si selecciona Eliminar, ver sección Eliminar Proyecto. - Si selecciona modificar, ver sección Modificar Proyecto. -Si selecciona Visualizar, ver sección Visualizar Proyecto.	5. El sistema brinda la posibilidad de asignar los proyectos a los diferentes especialistas para su revisión.
	7.El sistema brinda la posibilidad de: - Eliminar el proyecto. - Modificar el proyecto. - Visualizar el proyecto.
	8. El sistema brinda la posibilidad de enviar los proyectos a los diferentes especialistas. El sistema finaliza la ejecución del caso de uso.
Sección “Visualizar Proyecto”	
	1. El sistema muestra todos los proyectos que existen, finalizando así el caso de uso.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Sección "Modificar Proyecto"	
	1. El sistema muestra una opción para modificar el o los proyectos necesarios.
2. El actor selecciona el o los proyectos que se quieren modificar.	3. El sistema brinda la posibilidad de modificar el o los proyectos que se van a revisar en ese momento.
4. El actor modifica el o los proyectos seleccionados y presiona la opción Modificar.	5. El sistema modifica los datos, finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2. El sistema muestra un mensaje de información "Debe seleccionar el o los proyectos a modificar". Regresa al paso 4 del flujo alterno Modificar Proyecto.
Sección "Eliminar Proyecto"	
2. El actor selecciona el proyecto a eliminar y da clic en el botón eliminar.	1. El sistema muestra una opción para eliminar el o los proyectos en ese momento.
	3. El sistema elimina el proyecto. Finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
*Si el actor no desea eliminar el proyecto presiona la opción Cancelar.	
Prototipo de Interfaz	
Pos condiciones:	Quedan visualizados, modificados o eliminados los diferentes proyectos que se estén revisando en ese momento.

Tabla 5: Descripción del Caso de Uso Generar reportes del estado de los proyectos.

Caso de Uso:	Generar reportes del estado de los proyectos.
Actores:	Jefe del laboratorio.
Resumen:	El caso de uso inicia cuando el jefe del laboratorio desea realizar el reporte general del estado de los proyectos que se están revisando.
Precondiciones:	Deben de existir proyectos revisándose.
Referencias:	RF5.
Prioridad:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción de realizar el reporte general del estado de los proyectos que se estén revisando.	2. El sistema brinda la posibilidad de seleccionar los proyectos que se están revisando.
3. El actor selecciona los proyectos para realizar el reporte.	4. El sistema muestra el reporte del estado de los proyectos con la siguiente información: - Nombre del proyecto. - Especialista al frente de las pruebas. - Estado del proyecto. - Artefactos en revisión. - Observaciones.
	El sistema brinda la posibilidad de exportar el reporte. Finalizando así el caso de uso.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	***.** El sistema permite cancelar las opciones.
Prototipo de Interfaz	

Pos condiciones:	

Tabla 6: Descripción del Caso de Uso Generar reportes cuantitativos de las No Conformidades.

Caso de Uso:	Generar reportes cuantitativos de las No Conformidades.
Actores:	Jefe del laboratorio.
Resumen:	Este caso de uso se inicia cuando el jefe del laboratorio desea generar el reporte estadístico de las No Conformidades, el mismo se realiza una vez al mes.
Precondiciones:	Deben de existir proyectos revisándose y la cantidad de cada clasificación de las No Conformidades insertadas.
Referencias:	RF2, RF3, RF4.
Prioridad:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción de generar reportes del control estadístico de las No Conformidades.	2. El sistema brinda la posibilidad de seleccionar los proyectos a los que se le van a realizar el reporte.
3. El actor selecciona los proyectos que desea que se le haga el reporte.	4. El sistema brinda la posibilidad de seleccionar el tipo de reporte que desea realizar tales como: - Cantidad de No Conformidades por artefacto. - Cantidad de No Conformidades por iteración. - Artefacto con más No Conformidades según el tipo que se especifique.
5. El actor selecciona el tipo de reporte que desee.	6. El sistema muestra el reporte.
	7. El sistema brinda la posibilidad de exportar el reporte. Finalizando así el caso de uso.
Sección “”	
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
*Si el actor no desea eliminar el proyecto presiona la opción Cancelar.	
Prototipo de Interfaz	
Pos condiciones:	

Tabla 7: Descripción del Caso de Uso Gestionar estado de los proyectos.

Caso de Uso:	Gestionar estado de los proyectos.
Actores:	Especialista de Calidad.
Resumen:	Este caso de uso se inicia cuando el especialista de calidad al frente de las pruebas desea insertar o actualizar el estado en que se encuentran los proyectos que está revisando.
Precondiciones:	Deben de existir proyectos revisándose.
Referencias:	RF1, RF2, RF3, RF5.
Prioridad:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción de gestionar estado del proyecto.	2. El sistema muestra la posibilidad de seleccionar los proyectos que está revisando.
3. El actor selecciona el proyecto.	4. El sistema muestra el proyecto y brinda la posibilidad de seleccionar los artefactos del proyecto que se están revisando.
5. El actor selecciona el artefacto.	6. El sistema muestra el artefacto y brinda la posibilidad de cambiar el estado del proyecto y llenar las posibles observaciones que tenga la revisión del artefacto.

7. El actor selecciona el estado del proyecto y escribe las observaciones.	8. El sistema actualiza el estado del proyecto, la fecha y las observaciones y lo muestra finalizando así el caso de uso.
Sección “”	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
*Si el actor no desea realizar la opción presiona el botón Cancelar.	
Prototipo de Interfaz	
Pos condiciones:	Queda actualizado el estado del proyecto.

Tabla 8: Descripción del Caso de Uso Gestionar el control estadístico de las No Conformidades.

Caso de Uso:	Gestionar el control estadístico de las No Conformidades.
Actores:	Especialista de Calidad.
Resumen:	El caso de uso inicia cuando el especialista de calidad desea insertar la cantidad de No Conformidades según su clasificación e importancia por iteración en el proyecto que se encuentra al frente. También podrá modificar y eliminar la cantidad de estos errores encontrados.
Precondiciones:	
Referencias:	RF1, RF2, RF3, RF4.
Prioridad:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción de gestionar el control estadísticos de las No Conformidades.	2. El sistema brinda la posibilidad de seleccionar el proyecto, artefacto y la iteración.

3. El actor selecciona el proyecto, el artefacto y la iteración.	4. El sistema brinda la posibilidad de insertar la cantidad de No Conformidades según su clasificación e importancia por iteración.
5. El actor inserta la cantidad de No Conformidades según su clasificación e importancia por iteración.	6. El sistema verifica los datos y muestra los valores insertados. Además da la posibilidad de visualizar y modificar la cantidad de No Conformidades.
7. El actor selecciona una de las siguientes opciones: - Si selecciona visualizar la cantidad de No Conformidades, ver sección Visualizar cantidad de No Conformidades. - Si selecciona la opción modificar cantidad de No Conformidades, ver sección modificar cantidad de No Conformidades.	
Sección "Visualizar cantidad de No Conformidades"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra la cantidad de No Conformidades según su clasificación e importancia por iteración.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Sección "Modificar cantidad de No conformidades"	
Acción del Actor	**Respuesta del Sistema**

1. El actor modifica cualquiera de los datos que compone el reporte estadísticos de las No Conformidades: - Clasificación de la No conformidad - Iteración - Cantidad - Proyecto Y presiona modificar.	2. El sistema muestra una ventana para modificar la cantidad de no conformidades de los siguientes datos: - Cantidad de No Conformidades por clasificación. - Cantidad de No Conformidades por importancia.
3. El actor modifica algunos de los siguientes datos: - Cantidad de No Conformidades por clasificación. - Cantidad de No Conformidades por importancia.	4. El sistema valida los datos y modifica los datos, finalizando así el caso de uso.
Prototipo de Interfaz	
Pos condiciones:	Queda insertado, modificado o eliminado la cantidad de No Conformidades por su clasificación e importancia.

Conclusiones del Capítulo 2

En este capítulo se dio una breve descripción acerca de los roles que desarrollan dichos procesos y de otras personas que puedan estar involucradas. Se definió además la información manejada y la propuesta del sistema. Se identificaron los requisitos funcionales y no funcionales, los casos de uso con sus descripciones, los diagramas de actividades y el modelo de objeto.

Capítulo 3. Análisis y Diseño del Sistema.

Introducción.

En este capítulo se dará una breve descripción del análisis de los casos de uso del sistema para diseñar las clases que se implementarán. Se representan además los diagramas de colaboración del diseño, el diagrama de las clases diseñadas con sus relaciones, los principios utilizados para el diseño de dichas clases, los diagramas de clases persistentes, el diagrama entidad relación y la descripción de las tablas de la base de datos.

3.1 Modelo de Análisis.

El Modelo de análisis es una jerarquía de paquetes del análisis que contienen clases del análisis y las realizaciones de los casos de uso, además ofrece una especificación más precisa de los requisitos, sin dejar de mencionar que el modelo de análisis se describe utilizando el lenguaje de los desarrolladores, puede por tanto ser utilizado para razonar sobre los funcionamientos internos del sistema. El modelo de análisis brinda la facilidad de estructurar los requisitos de un modo que facilite su comprensión, su preparación, su modificación y en general su mantenimiento; además se puede considerar como la primera aproximación al modelo del diseño.

3.1.1 Diagramas de Clases del Análisis.

Una clase de análisis representa una abstracción de los subsistemas del diseño del sistema. Las clases del análisis siempre encajan en uno de los tres estereotipos básicos: de interfaz, de control o entidad. Cada estereotipo implica una semántica específica, lo cual constituye un método potente y consistente de identificar y describir las clases del análisis.

Ver información acerca de los diagramas de clases del análisis en el Anexo 1.

3.2 Modelo del Diseño.

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Una clase de diseño es una abstracción o construcción similar de una clase en la implementación del sistema, debido a que:

• El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación. Esto significa que las operaciones, parámetros, atributos y tipos son especificados utilizando sintaxis del lenguaje de programación elegido.

• Las relaciones de aquellas clases del diseño implicadas con otras clases, a menudo tienen un significado directo cuando la clase es implementada. Por ejemplo, la generalización o algún estereotipo de generalización tienen una semántica que se corresponde con el significado de generalización (o herencia) en el lenguaje de programación. Estas son, las asociaciones y agregaciones que a menudo se corresponden con variables (atributos) de clases en la implementación para proporcionar referencias entre objetos.

• Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases (esto es, en el código). Si los métodos se especifican en el diseño, se suelen especificar en lenguaje natural, o en seudo código, y por eso pueden ser reutilizados como comentarios en las implementaciones del método. Esto es una de las principales abstracciones entre el diseño y la implementación y es raramente necesario por lo que se recomienda que el mismo desarrollador diseñe e implemente una clase.

UML posee una extensión para el modelado de aplicaciones Web, dicha extensión es usada para el diseño de las clases. Los estereotipos que usa esta extensión son:

<<Server page>>Representa la página Web que tiene códigos que se ejecutan en el servidor. Estos códigos interactúan con recursos en el servidor. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.

Esta clase sólo puede tener relaciones con objetos en el servidor, una relación 1:1 con un fichero en el servidor. En las aplicaciones en PHP se corresponde con un fichero **.**PHP.

<<Client Page>> Una instancia de Página Cliente es una página Web, con formato HTML, mezcla de datos, presentación y lógica. Cada página cliente solo puede ser construida por una página servidor.

<<Form>> Grupo de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (Text Field, Text Area, Button, Label, Radio Button, Radio Group, Select, Check Box y HiddenFields).

3.2.1 Diagramas de Colaboración del Diseño.

Ver descripción acerca de los diagramas de colaboración en el Anexo 2.

3.2.2 Diagramas de clase del Diseño.

Ver información acerca de los diagramas de clase del diseño en el Anexo 3.

3.3 Arquitectura Web de tres Niveles.

La arquitectura de las aplicaciones Web suelen presentar un esquema de tres niveles. El primer nivel consiste en la capa de presentación que incluye no solo el navegador, sino también el servidor Web que es el responsable de dar a los datos un formato adecuado. El segundo nivel está referido habitualmente algún tipo de programa o script. Finalmente, el tercer nivel proporciona al segundo los datos necesarios para su ejecución.

Una aplicación Web típica recogerá datos del usuario (primer nivel), los enviará al servidor, que ejecutará un programa (segundo y tercer nivel) y cuyo resultado será formateado y presentado al usuario en el navegador (primer nivel otra vez).

3.3.1 Patrón arquitectónico. Modelo Vista Controlador del framework Symfony.

Un patrón arquitectónico caracteriza los componentes de acuerdo a la responsabilidad, relación y la forma en la que colaboran para darle solución a un problema dentro del desarrollo de software. La arquitectura del Modelo Vista Controlador (MVC) está determinada por tres niveles:

- Modelo: Información con la que trabaja la aplicación (lógica de negocio), haciendo que la vista y las acciones sean independientes.

- Vista: Realiza la transformación del modelo de una página web de forma tal que le permite al usuario interactuar con ella (presentación).

- Controlador: Es el encargado de procesar la interacción del usuario, aislando al modelo de la vista.

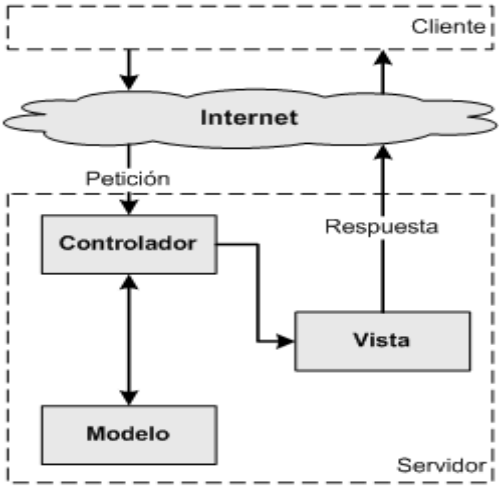


Figura 10: Patrón MVC.

3.3.1 Descripción del MVC del framework Symfony

Controlador

La capa del controlador en Symfony se encuentra dividido en dos partes: el controlador frontal y las acciones. La única entrada a la aplicación es a través del controlador frontal, es decir este carga la configuración y determina la acción a ejecutarse. La lógica de las páginas ocurre mediante las acciones, realizando la verificación de la integridad de las peticiones y preparan los datos requeridos por la capa de presentación.

Vista

La vista en Symfony está compuesta por las plantillas, presentación de los datos de las acciones que se están ejecutando, y el layout, el cual contiene el código HTML común a todas las páginas.

Modelo

La capa de tipo Object/Relational Mapping (ORM) es el componente que se encarga de gestionar el modelo en Symfony mediante el proyecto Doctrine. El acceso y la modificación de los datos almacenados en la base de datos de los proyectos desarrollados bajo el framework Symfony se realiza

mediante objetos, lo cual permite un alto nivel de abstracción y una fácil portabilidad. Para la abstracción de las base de datos Symfony se utiliza el lenguaje PHP Data Object (PDO).

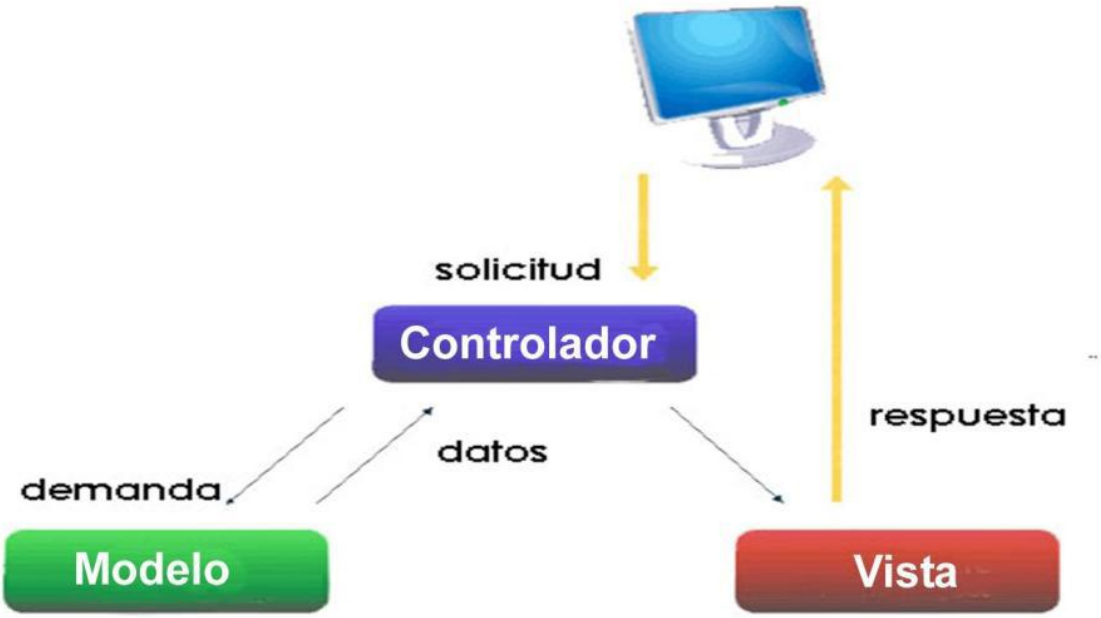


Figura 11: Patrón MVC de Symfony.

3.4 Diseño de la Base de Datos

Las bases de datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. Es por ello que su diseño es muy importante en su desarrollo. Al diseñar una base de datos, se refleja la estructura del problema en el mundo real, evita el almacenamiento de información redundante y permite además representar todos los datos esperados, incluso con el paso del tiempo. Para lograr diseñar la base de datos se partió de la definición de las clases persistentes, lo cual podemos definir como la capacidad de un objeto de mantener su valor en el espacio y tiempo. Después del refinamiento y clasificación de estas clases y sus atributos se realizó la construcción del diagrama de clases persistentes.

3.4.1 Modelo lógico de datos

El modelo lógico de datos representa la información que maneja el sistema, siendo una fuente de información para el modelo físico. A continuación se muestra el Modelo lógico de datos diseñado para la base de datos de la aplicación de este trabajo de diploma, el cual no es más que el diagrama de

clases persistentes. El diagrama de clases persistentes es un diagrama de clases en el que, como bien indica su nombre, solo aparecen las clases persistentes, de las que hay que expandir detalles estructurales. Estas clases persistentes tuvieron como origen las clases clasificadas como entidad, ya que estas modelan la información y el comportamiento asociado al fenómeno en cuestión.

Ver información acerca del diagrama de clases persistentes en el Anexo 4.

3.5 Diseño de la Base de Datos. Modelo Entidad Relación.

Ver la información más detallada en el Anexo 5.

3.5.1 Descripción de las Tablas.

Se representa la descripción de todas las tablas de la Base de Datos en el Anexo 6.

3.6 Principios del Diseño.

Para garantizar un vínculo entre el usuario y el sistema llevado a cabo se ha hecho un esfuerzo por garantizar la mejor claridad posible en la interfaz procurando que sea intuitiva y que garantice una funcionalidad óptima del sistema y una comodidad en el trabajo de los usuarios finales.

Se trabajó con formularios Web, para lograr una mayor visualización de la información, se utilizó un mismo color en todas las páginas y un mismo tipo y tamaño de letra con textos claros, el color predominante es el azul. El envío de la información debe ser lo más rápida posible por lo que no se utilizaron muchas imágenes ni funciones que atenten contra esto y que posibiliten además una navegación rápida y eficiente. Cada página está representada por un título acorde a su funcionalidad. Los botones principales sirven de referencia a la función de los mismos.

Modificar  *Eliminar* 

3.7 Tratamiento de Errores.

Se contará con un sistema de tratamiento de errores para disminuir la posibilidad de cometerlos. Para esto se detallará con la validación de la información introducida en el sistema por la validación de los formularios utilizando funciones Java Scripts. Mediante la interfaz Web se evitará que el usuario asuma un papel activo en la introducción de la información, para esto se contará con cuadros de opción, menú

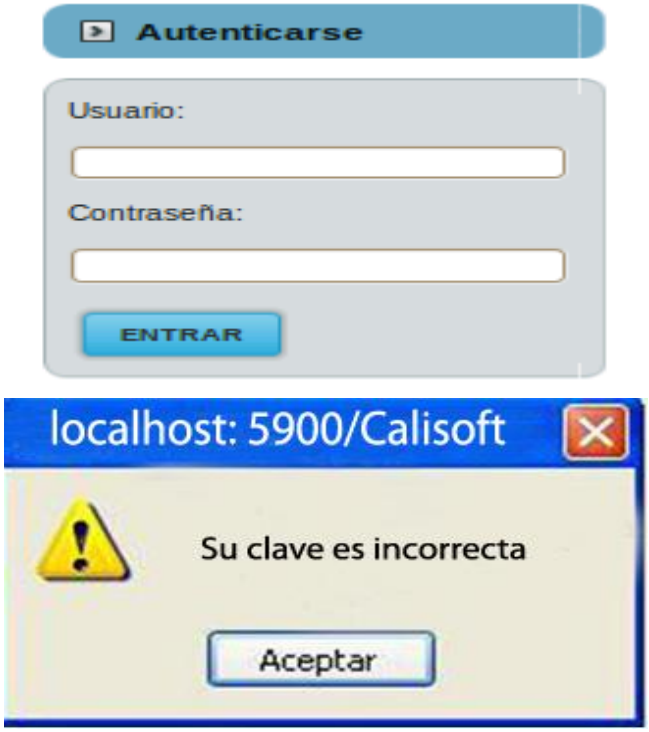
de selección lo cual facilitará la entrada de datos. La información que pretenda ser adicionada por el usuario se validará mediante funciones que avalen que sea válida y que el cuadro de texto no se encuentre vacío si es obligatorio llenarlo. Si hay un error en la información le saldrá al usuario un mensaje en pantalla indicándole el error, al oprimir Aceptar el mensaje desaparece y el usuario podrá seguir introduciendo los datos en el formulario.

También se validarán las opciones correspondientes a la extracción o modificación de datos del servidor de base datos. Si se desea eliminar algún elemento de la base de datos se preguntará al usuario si está seguro de realizar dicha acción, al igual que cuando desee modificar alguna información, antes de actualizarla se le preguntará si desea realizarla o no. Así se logra que se ejecuten las operaciones que se desean y que se rectifique al cometer un error. A continuación se muestran algunos de estos mensajes:

3.8 Seguridad.

La seguridad en el sitio está implementada a través del servidor de base de datos PostgreSQL y el uso de variables de sesión para restringir el acceso de los usuarios a determinadas páginas. Así como también un Sub_ Módulo de autenticación que verifica que el usuario existe en la aplicación y que los datos que introduce sean correctos, en caso de no ser así no tendrá acceso a la aplicación. Para

garantizar la seguridad de la información se crearon varios niveles de seguridad, definidos como tipos de usuario, que pueden ser: especialista de calidad, jefe del laboratorio de prueba y el administrador. Este último es el encargado del buen funcionamiento del sistema por lo que tendrá derecho al control total del mismo. Los demás usuarios no tendrán acceso a la información restringida para ellos, para esto se trabaja con variables de sesión de forma tal que siempre se sabe qué usuario intenta visitar dichas páginas y estas se muestran sólo para aquellos que pueden tener acceso a ellas. A continuación se muestran algunos de estos mensajes:



Conclusiones del Capítulo 3.

En este capítulo se dio una breve explicación acerca del análisis de los casos de uso del sistema para diseñar las clases que se implementarán, se representaron los diagramas de colaboración del diseño, los diagramas de las clases diseñadas con sus relaciones, los principios utilizados para el diseño de dichas clases, los diagramas de clases persistentes, el diagrama entidad relación y la descripción de las tablas de la base de datos. También se explicó la definición del diseño que se aplicó, la forma de tratar los errores, se definió la seguridad del sistema y la concepción de la ayuda.

Capítulo 4. Implementación y prueba.

Introducción.

En este capítulo se reflejan los diferentes diagramas correspondientes a este flujo de trabajo, donde se implementará la propuesta realizada en el análisis y diseño. Se representa el diagrama de despliegue, los de componentes y el modelo de prueba correspondiente a la aplicación desarrollada.

4.1 Implementación.

En la implementación se comienza con el resultado del análisis y diseño, y se implementa el sistema en términos de componentes, es decir: ficheros de código fuente, scripts, ficheros de códigos binarios y ejecutables. Dentro de los principales objetivos se encuentran: distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue e implementar las clases y subsistemas encontrados en el diseño.

4.1.1 Diagrama de Despliegue.

El modelo de despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Se puede observar lo siguiente sobre el modelo de despliegue:

- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar.

- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como HTTP, USB, TCP/IP, etc.

- Puede describir diferentes configuraciones de red, incluidas las configuraciones para pruebas y para simulación.

- La funcionalidad (los procesos) de un nodo se define por los componentes que se distribuyen sobre ese nodo.

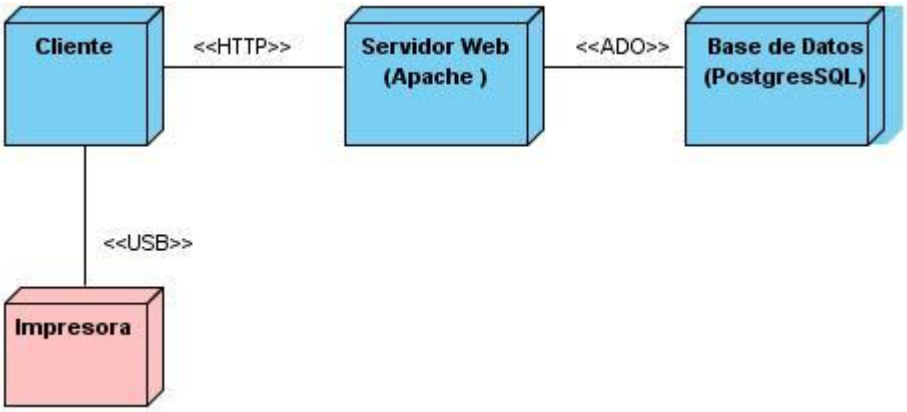


Figura 12: Diagrama de Despliegue.

4.1.2 Diagramas de Componentes.

Ver información acerca de los diagramas de componentes en el Anexo 7.

4.2 Modelo de Prueba.

En el modelo de prueba se describirán los niveles de prueba a utilizar ya mencionados al principio del desarrollo de la investigación. Dentro de estas pruebas se van a utilizar las pruebas modulares y de integración las cuales se llevan a cabo mediante la técnica de prueba de Caja Negra, así como las pruebas del sistema. Además se puede observar que existen dentro del diseño de pruebas varios tipos de prueba y dentro de ellas se van a utilizar las pruebas de funcionalidad, la cual se le realiza a la prueba de integración, y se tienen además las pruebas de carga y stress, para verificar en qué estado se encuentra el sistema.

4.2.1 Estrategias de pruebas.

La estrategia de prueba que se va a implementar a esta aplicación es la que se muestra a continuación:

1. **Pruebas de Integración**. A partir del esquema del diseño, los módulos probados se vuelven a probar combinados para probar sus interfaces.

2. **Prueba del Sistema**. El software ensamblado totalmente con cualquier componente hardware que requiere de prueba para comprobar que se cumplen los requisitos funcionales.

4.2.2 Herramientas para las pruebas.

El propósito de un caso de prueba es especificar una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, los resultados esperados y las condiciones bajo las que ha de probarse. Los casos de prueba son un producto de desarrollo de software que ayudan a validar y verificar las expectativas de los stakeholders.

Los requerimientos son la fuente principal para obtener los casos de pruebas pero no son el único medio. Y muchas veces son insuficientes para proporcionar una base completa que permita desarrollar las pruebas. Por lo que es necesario considerar otros elementos como riesgos, restricciones, tecnologías, cambios y fallos. Normalmente, un caso de prueba se deriva de un caso de uso en el modelo de casos de uso o de una realización de un caso de uso en el modelo del diseño. Con estos casos de prueba se validan los requerimientos funcionales del sistema. Los siguientes son casos de pruebas comunes:

- Un caso de prueba especifica cómo probar un caso de uso o un escenario específico de un caso de uso. Un caso de prueba de este tipo incluye la verificación del resultado de la interacción entre los actores y el sistema, donde se satisfacen las precondiciones y pos condiciones especificadas por el caso de uso y que se sigue la secuencia de acciones especificadas por el caso de uso.
- Un caso de prueba basado en un caso de uso especifica típicamente una prueba del sistema como “caja negra”, es decir, una prueba del comportamiento observable externamente del sistema.
- Un caso de prueba que especifica cómo probar una realización de caso de uso-diseño o un escenario específico de la realización, puede incluir la verificación de la interacción entre los componentes que implementan dicho caso de uso. Los casos de pruebas basados en una realización de un caso de uso típicamente especifican una prueba del sistema como “caja blanca”, es decir, una prueba de la interacción interna entre los componentes del sistema.

Se debería desarrollar un caso de prueba para cada escenario del caso de uso. Los escenarios de un caso de uso se identifican describiendo los distintos caminos del flujo básico y flujo alternativo del caso de uso.

Además, especificar como otra herramienta JMeter, la cual es una herramienta de software libre que ofrece la posibilidad de realizar pruebas de carga para llevar acabo simulaciones sobre cualquier recurso de software.

Inicialmente diseñada para pruebas de stress en aplicaciones web, hoy en día su arquitectura ha evolucionado no sólo para llevar a cabo 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, o de cualquier otro cliente, siendo capaz de manipular resultados en determinada requisición y reutilizarlos para ser empleados en una nueva secuencia. (Pello)

Se representa la descripción de las secciones por escenarios en el Anexo 8.

4.2.3 Resultado de las Pruebas.

En las pruebas funcionales realizadas a la herramienta se obtuvieron 160 No Conformidades en las tres iteraciones, de ellas 145 fueron significativas, 11 no significativas y 4 de recomendaciones. Las No Conformidades encontradas más comunes fueron de validación, opciones que no funcionan y funcionalidad. Los detalles del resultado se muestran en la siguiente tabla.

No Conformidades	1ra Iteración	2da A Iteración	3ra Iteración
No Significativas	6	4	1
Recomendaciones	3	1	0
Formato	3	1	0
Error técnico	1	0	0
Validaciones	22	15	5
Opciones que no funcionan	13	7	2
Errores de interfaz	7	2	2
Funcionalidades	16	11	9
Excepciones	7	3	1
Ortografía	8	3	1
Redacción	1	0	0
Errores de idioma	2	1	0
Otras documentaciones	1	0	0

Otras implementaciones	1	0	0
Total Significativas	82	43	20

En la prueba de carga y stress realizada a la herramienta se obtuvieron los siguientes resultados, para 2640 peticiones que se le hicieron al servidor, los tiempos mínimos fueron de 0 milisegundos y los tiempos máximos fueron de 11671 milisegundos. El 90 % de las páginas respondió en 7422 milisegundos. El rendimiento de las páginas que se probaron se encuentra en 13.5 segundos.
Ver el resultado de la prueba de carga y stress en el Anexo 9.

Conclusiones del Capítulo 4.

En este capítulo se realizó toda la implementación del sistema y se mostraron los diagramas de despliegue y componentes, con los casos de pruebas elaborados se comprobó la integración de los módulos del sistema.

El uso del patrón MVC del framework Symfony, al igual que la aplicación de patrones de diseño, ha hecho más fácil el diseño de las clases permitiendo que se obtengan clases mejor diseñadas, modulares, flexibles y reutilizables. A partir del diseño de clases persistentes y la estructuración del diagrama de clases persistentes se obtuvo el modelo de datos, lo que facilitó el diseño de las tablas de la base de datos. La realización del diagrama de despliegue propició una visión de cómo está distribuido el sistema físicamente.

Conclusiones

- La elaboración de la herramienta informática, implementada a partir de la necesidad de automatizar los procesos que se llevan a cabo en el Laboratorio Central de Calidad de Software de la UCI, permitirá elevar la eficiencia en el proceso de revisión de los productos, brindando la posibilidad de dar respuesta a las solicitudes de los clientes en el menor tiempo posible además se obtendrá como resultado la centralización y el control estadístico de todas las No Conformidades encontradas de todas las pruebas realizadas a un software en la Universidad de las Ciencias Informáticas.
- Con la definición del proceso anteriormente expuesto se garantizará una excelente gestión de las No Conformidades de un laboratorio de pruebas de software.

Recomendaciones

- Aplicar esta herramienta en otros laboratorios de pruebas de software con el objetivo de generalizar la propuesta para elevar la eficiencia de estos procesos y evaluar las mejoras posibles para una nueva versión.

- Diseñar a partir del sistema desarrollado y utilizando los datos que se almacenen durante un tiempo, un sistema automatizado que permita la generación automática de algunos datos y ayude a mantener el control estadístico en cantidades exactas de estos datos.

Referencias Bibliográficas

- **Roche, Raul. 2008 - 2009.** Revista Bioanalisis de Grupo Bio SRL. [Citado el: 15 de Diciembre de 2010.]. Disponible en: http://www.revistabioanalisis.com/ejemplares/ejemplar/notas/index.php?id_edicion=25&id_nota=313. *[**Omega 4**]*
- **Grimán, Anna C., Pérez, María y Mendoza, Luis E.** Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales. Guiado por Roger Pressman. [Citado el: 14 de Abril de 2010.] Disponible en: http://www.lisi.usb.ve/publicaciones/03%20evaluacion/evaluacion_15.pdf. *[**Tipos de Pruebas**]*
- **Pressman, Roger S., y otros.1990.** Generar un Plan de Pruebas. [Citado el: 14 de Abril de 10.] Disponible en: http ://www.mitecnologico.com/Main/GenerarUnPlanDePruebas. *[**Tipos de Pruebas**]*
- **Profesor: López., Juan Antonio.** Pruebas de Software. [En línea] [Citado el: 14 de Abril de 2010.] . Disponible en: http://www.google.com.cu/search?hl=es&client=firefox-a&hs=Dhm&rls=org.mozilla%3Aen-GB%3Aofficial&q=Tipos+pruebas+%2B+Pressman&meta=lr%3Dlang_es&aq=f&aqi=&aql=&oq=&gs_rfai=, http://biblioteca.reduc.edu.cu/biblioteca.virtual/cgi/CD-ROM/otros/U. [***Niveles de Pruebas***].
- **Mares Amézquita, Saúl. 2007.** Gestión de No Conformidades. [En línea] 2007. [Citado el: 19 de Diciembre de 2010.] Disponible en: http://www.mex.ops-oms.org/contenido/tuberculosis/cdtaller/presentaciones/M%C3%B3dulo%208%20Gesti%C3%B3n%20de%20No%20Conformidades.pdf; http://iesprimerodemayo.wordpress.com/2007/10/29/%C2%BFque-es-una-no-conformidad. *[**Concepto de No Conformidades**]*
- **Social, Sede.** AENOR (Asociación Española de Normalización y Certificación). [Citado el: 3 de Marzo de 2010.] Disponible en: http://www.bicgalicia.es/documentos/noticias/doc01_010704.pdf. [***Significado de: NORMA UNE EN ISO 9001:2000]***
- **Fernández Pereda, Héctor.** Norma ISO 9001:2000 Sistema de Gestión de la Calidad y requisitos. [Citado el: 3 de Marzo de 2010.] Disponible en: http://www.buscarportal.com/articulos/iso_9001_2000_gestion_calidad.html. .*[**Significado de: NORMA UNE EN ISO 9001:2000]*.

- **2008.** Norma ISO 9001:2000 AENOR (Asociación Española de Normalización y Certificación). [En línea] Noviembre de 2008. [Citado el: 3 de Marzo de 2010.] Dsiponible en: http://www.webproyecto.com/asercal/normas.html/; http://www.webproyecto.com/asercal/isoenisouneeniso.html.*[**Significado de: NORMA UNE EN ISO 9001:2000**]*.
- **Grupo Soluciones, GSINNOVA. 2007.** RUP, CMM, CMMI. [Citado el: 16 de Abril de 2010.] Disponible en: http://www.rational.com.ar/herramientas/rup.html. *[**Concepto de RUP**].*
- **Zapata, Luis Giraldo y Yuliana. 2005.** Herramientas de Desarrollo de Ingenieria de Software para Linux .[Citado el: 7 de Marzo de 2010.] Disponible en: http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas%20de%20ISW.pdf,_id=17. *[**Qué es UML**].*
- **Hernández Orallo, Enrique.** Lenguaje Unificado de Modelado (UML). [Citado el: 17 de diciembre de 2010.] Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF. *[**Qué es UML**].*
- **2007.** Sitios de Descarga de Software. 7. [Citado el: 20 de Febrero de 2010.] Disponible en: Página de productos:http://www.visual-paradigm.com, 5 de Marzo de 2007, http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_1471_p/.*[**Qué es Visual Paradigm**].*
- **Álvarez, Miquel Angel. 2002.** Desarrolloweb.com. [Citado el: 20 de Febrero de 2010.] Disponible en: http://www.desarrolloweb.com/articulos/332.php. [***Concepto de***
- **García, Joaquín. 1998 – 2004 (última modificación en el 2004.).** WebEstilo: Conceptos Básicos. [Citado el: 14 de Abril de 2010.] Disponible en: http://www.webestilo.com/php/php00.phtml.. ***[Conceptos Básicos de PHP]**.*
- **McCool, Rob. 1995.** Master Magazine: Tu portal de Tecnología actualizado a diario. *Su desarrollo comenzó con una tentativa de mejorar el servidor existente en el NCSA. La 1ra Versión.* [Citado el: 14 de Abril de 2010.] Disponible en: http://www.mastermagazine.info/termino/6051.php. *[**Apache Servidor Web (Código Abierto)**]*
- **Grupo de Trabajo, SuperUser. 2002. (última modificación: 2005 (2010).).** TiendaLinux.com. 8.1, TiendaLinux.com, 26 de 12 de. 26 -12- 2002. (última modificación: 2005 (2010).). [Citado el:] Disponible en: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.*[**Concepto de PostgreSQL**].*
- **Pérez Valdés, Damián. 2007.** Maestros de la Web. Publicado el 31 de Julio, 2007. Publicado el 31 de Julio, de 2007. Disponible en:

- http://www.maestrosdelweb.com/editorial/los-frameworks-de-php-agilizan-tu-trabajo/.. [***Qué es un Framework***]
- **Giraldo, Luis y Zapata, Yuliana. 2005.** HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX. 24 de Septiembre de 2005. Disponible en: http://www.librosweb.es/symfony_1_2. [***Qué es un Framework***]
- **Zaninotto Potencier, François, y otros.** SYMFONY 1.2 la quía definitiva. .] [Citado el: 2 de marzo de 2010.] Disponible en: http://www.librosweb.es/symfony_1_2.*[**Qué es SYMFONY**].*
- **Pello, Javier.** SoftQaNetwork. [Citado el: 10 de Mayo de 2010.] Disponible en: http://www.softqanetwork.com/2005/10/jmeter/.*[**Qué es JMeter**]*

Bibliografía

- **Rojas Aros, Rodrigo y Calidad, Tec. Médico Especialista en Aseguramiento de. 2005.** La Gestión de Calidad y el Mejoramiento del Diagnóstico de Microorganismos Fúngicos. Universidad Chile. Facultad de Química y Farmacia. [En línea] 15 de Enero de 2005. [Citado el: 15 de Febrero de 2010.] Disponible en: http://www.cienciaytrabajo.cl/pdfs/15/pagina%2001.pdf.
- **Expósito, Antonia, y otros.** ACREDITACIÓN Y CERTIFICACIÓN EN LABORATORIOS DE REPRODUCCIÓN ASISTIDA. *Unidad de Reproducción Humana. Departamento de Ginecología y Obstetricia. Hospital de Cruces. Plaza de Cruces S/N, Baracaldo (Vizcaya).* [En línea] [Citado el: 17 de 02 de 2010.] Disponible en:http://www.asebir.com/congreso/granada/acreditacion.pdf.
- **Pzo Zulueta, Delmys, y otros. 2009.** Revista Española de Innovación, Calidad e Ingeniería de Software (REICIS). *Procedimientos parapruebas de intrusión en aplicaciones Web. Universidad de las Ciencias Informáticas (UCI).* [En línea] 31 de Julio de 2009. [Citado el: 16 de 02 de 2010.] Disponible en:
- http://www.ati.es/IMG/pdf/PozoVol5Num2.pdf.
- **2007.** Boletín Trimestral del Instituto Tecnológico de Informática, dedicado a las Tecnologías de la Información y las Comunicaciones. Actualidad_TIC. *Revista del Instituto Tecnológico de Informática. Java Libre y Abierto.* [En línea] Mayo de 2007. [Citado el: 13 de Febrero de 2010.] Disponible en: http://www.iti.es/media/about/docs/tic/12/n12.pdf.
- **2009.** Clasificaciones de las No Conformidades en un proceso de pruebas de liberación (UCIENCIA).doc. [En línea] 2009. [Citado el: 14 de 02 de 2010.]
- **Laborde Reybaud, Alejandro, y otros.** IV SIMPOSIO INTERNACIONAL DE SISTEMAS DE INFORMACIÓN E INGENIERÍA DE SOFTWARE EN LA SOCIEDAD DEL CONOCIMIENTO. SISOFT2006 VOLUMEN 1. *INGENIERÍA DEL SOFTWARE. Excepciones en las Políticas en un sistema de Gestión de Procesos de Negocios. Estado del -Arte.* [En línea] [Citado el: 19 de Febrero de 2010.] Disponible en:
- http://novella.mhhe.com/sites/dl/free/8448118952/540197/ActasVol1SISOFT2006.pdf.**Góngora Rodríguez, Asnier E. y Nieves Borrero, Martha. 2007.** *Tesis: Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad de Software: Módulo Gestión de las No Conformidades. Ciudad de la Habana Facultad 6.* 2007. [Citado el: 18 de 02 de 2010.] Disponible en: http://biblioteca.reduc.edu.cu/biblioteca.virtual/cgi/CD-

ROM/otros/UCIENCIA%202007%20(E)/ponencias/tcis/PDF/Proceso%20de%20pruebas%20de%20caja%20negra-3730465792/PROCES~1.pdf.

- **Morales Sánchez, Verónica y Hernández Mendo, Antoni. 2004.** La calidad y su gestión. [En línea] Septiembre de 2004. [Citado el: 25 de Diciembre de 2009.] Disponible en: http://www.efdeportes.com/efd76/calidad.htm., http://techerald.com/post.view?jmeter-una-herramienta-para-pruebas-de-carga-aplicaciones-web-051220083713.html., http://www.lisi.usb.ve/publicaciones/03%20evaluacion/evaluacion_15.pdf.
- **Roche, Raul. 2008 - 2009.** Revista Bioanalisis de Grupo Bio SRL. [Citado el: 15 de Diciembre de 2010.]. Disponible en: http://www.revistabioanalisis.com/ejemplares/ejemplar/notas/index.php?id_edicion=25&id_nota=313. *[**Omega 4**]*
- **Grimán, Anna C., Pérez, María y Mendoza, Luis E.** Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales. Guiado por Roger Pressman. [Citado el: 14 de Abril de 2010.] . Disponible en: http://www.lisi.usb.ve/publicaciones/03%20evaluacion/evaluacion_15.pdf. *[**Tipos de Pruebas**]*
- **Pressman, Roger S., y otros. 1990.** Generar un Plan de Pruebas. [Citado el: 14 de Abril de 10.] Disponible en: http ://www.mitecnologico.com/Main/GenerarUnPlanDePruebas. *[**Tipos de Pruebas**]*
- **Profesor: López., Juan Antonio.** Pruebas de Software. [En línea] [Citado el: 14 de Abril de 2010.] . Disponible en: http://www.google.com.cu/search?hl=es&client=firefox-a&hs=Dhm&rls=org.mozilla%3Aen-GB%3Aofficial&q=Tipos+pruebas+%2B+Pressman&meta=lr%3Dlang_es&aq=f&aqi=&aql=&oq=&gs_rfai=, http://biblioteca.reduc.edu.cu/biblioteca.virtual/cgi/CD-ROM/otros/U. [***Niveles de Pruebas***].
- **Mares Amézquita, Saúl. 2007.** Gestión de No Conformidades. [En línea] 2007. [Citado el: 19 de Diciembre de 2010.] Disponible en:http://www.mex.ops-oms.org/contenido/tuberculosis/cdtaller/presentaciones/M%C3%B3dulo%208%20Gesti%C3%B3n%20de%20No%20Conformidades.pdf; http://iesprimerodemayo.wordpress.com/2007/10/29/%C2%BFque-es-una-no-conformidad. *[**Concepto de No Conformidades**]*
- **Social, Sede.** AENOR (Asociación Española de Normalización y Certificación). [Citado el: 3 de Marzo de 2010.] Disponible en:

http://www.bicgalicia.es/documentos/noticias/doc01_010704.pdf. [***Significado de: NORMA UNE EN ISO 9001:2000]***

- **Fernández Pereda, Héctor.** Norma ISO 9001:2000 Sistema de Gestión de la Calidad y requisitos. [Citado el: 3 de Marzo de 2010.] Disponible en: http://www.buscarportal.com/articulos/iso_9001_2000_gestion_calidad.html. .*[**Significado de: NORMA UNE EN ISO 9001:2000]**.*
- **2008.** Norma ISO 9001:2000 AENOR (Asociación Española de Normalización y Certificación). [En línea] Noviembre de 2008. [Citado el: 3 de Marzo de 2010.] Dsiponible en: http://www.webproyecto.com/asercal/normas.html/; http://www.webproyecto.com/asercal/isoenisouneeniso.html.. .*[**Significado de: NORMA UNE EN ISO 9001:2000]**.*
- **Grupo Soluciones, GSINNOVA. 2007.** RUP, CMM, CMMI. [Citado el: 16 de Abril de 2010.] Disponible en: http://www.rational.com.ar/herramientas/rup.html. *[**Concepto de RUP**].*
- **Zapata, Luis Giraldo y Yuliana. 2005.** Herramientas de Desarrollo de Ingenieria de Software para Linux . [Citado el: 7 de Marzo de 2010.] Disponible en: http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas%20de%20ISW.pdf,_id=17. *[**Qué es UML**].*
- **Hernández Orallo, Enrique.** Lenguaje Unificado de Modelado (UML). [Citado el: 17 de diciembre de 2010.] Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF. *[**Qué es UML**].*
- **2007.** Sitios de Descarga de Software. 7. [Citado el: 20 de Febrero de 2010.] Disponible en: Página de productos:http://www.visual-paradigm.com, 5 de Marzo de 2007, http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_1471_p/..*[**Qué es Visual Paradigm**].*
- **Álvarez, Miquel Angel. 2002.** Desarrolloweb.com..[Citado el: 20 de Febrero de 2010.] Disponible en: http://www.desarrolloweb.com/articulos/332.php. [***Concepto de***
- **García, Joaquín. 1998 – 2004 (última modificación en el 2004.).** WebEstilo: Conceptos Básicos. [Citado el: 14 de Abril de 2010.] Disponible en: http://www.webestilo.com/php/php00.phtml.. ***[Conceptos Básicos de PHP]***.
- **McCool, Rob. 1995.** Master Magazine: Tu portal de Tecnología actualizado a diario. *Su desarrollo comenzó con una tentativa de mejorar el servidor existente en el NCSA. La 1ra Versión.* [Citado el: 14 de Abril de 2010.] Disponible en: http://www.mastermagazine.info/termino/6051.php. *[**Apache Servidor Web (Código Abierto)**]*

- **Grupo de Trabajo, SuperUser. 2002. (última modificación: 2005 (2010).).** TiendaLinux.com. 8.1, TiendaLinux.com, 26 de 12 de. 26 -12- 2002. (última modificación: 2005 (2010).). [Citado el:] Disponible en: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.. *[**Concepto de PostgreSQL**].*
- **Pérez Valdés, Damián. 2007.** Maestros de la Web. Publicado el 31 de Julio, 2007. Publicado el 31 de Julio, de 2007. Disponible en:
- http://www.maestrosdelweb.com/editorial/los-frameworks-de-php-agilizan-tu-trabajo/..[***Qué es un Framework***]
- **Giraldo, Luis y Zapata, Yuliana. 2005..** HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX. 24 de Septiembre de 2005. Disponible en: http://www.librosweb.es/symfony_1_2. [***Qué es un Framework***]
- **Zaninotto Potencier, François, y otros.** SYMFONY 1.2 la quía definitiva. .] [Citado el: 2 de marzo de 2010.] Disponible en: http://www.librosweb.es/symfony_1_2..*[**Qué es SYMFONY**].*
- **Pello, Javier.** SoftQaNetwork. [Citado el: 10 de Mayo de 2010.] Disponible en: http://www.softqanetwork.com/2005/10/jmeter/..*[**Qué es JMeter**]*

Anexo 1: Diagramas de Clases del Análisis.

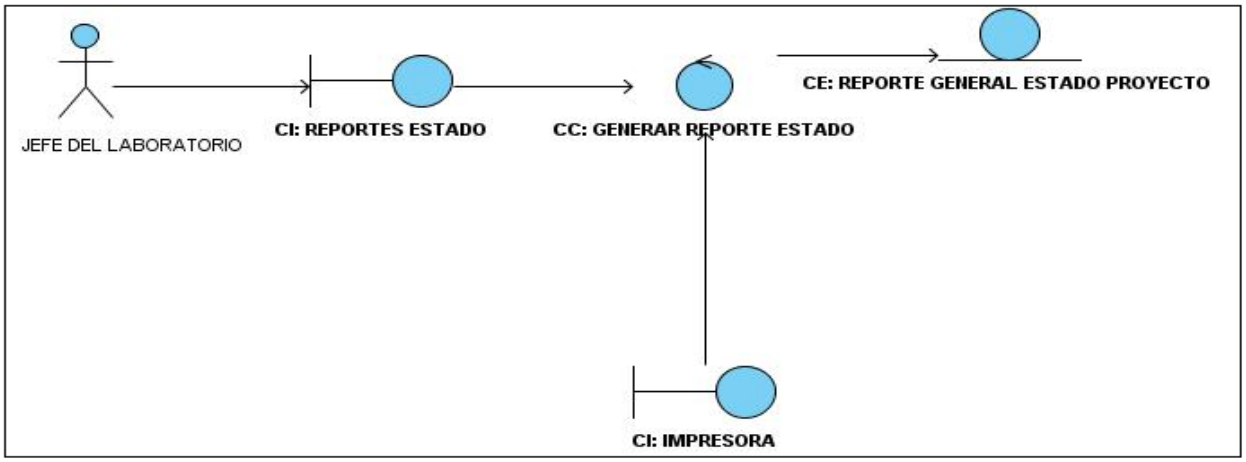


Figura 13: Diagrama de Clase del Análisis: Generar Reporte del Estado del Proyecto.

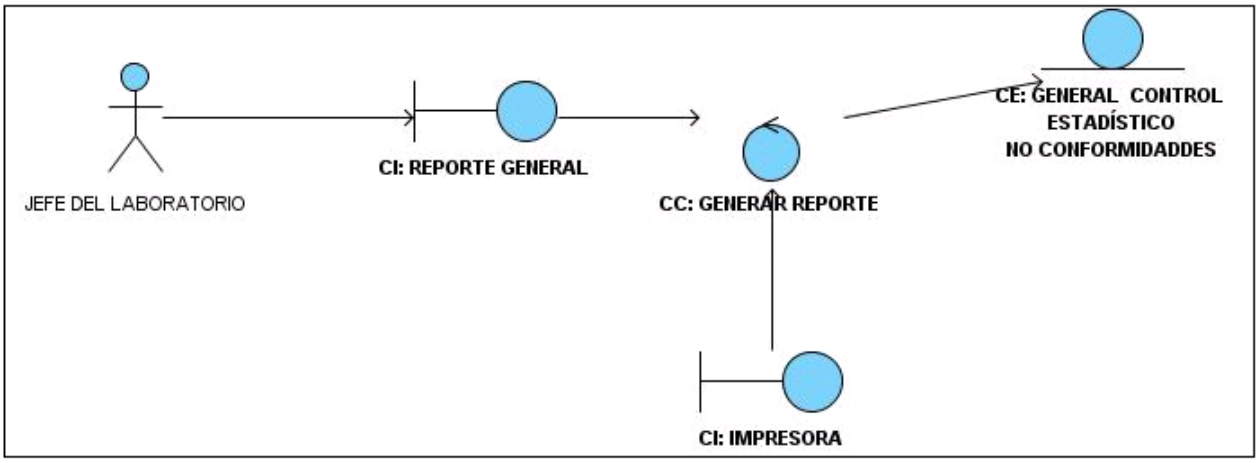


Figura 14: Diagrama de Clase del Análisis: Generar Reporte Cuantitativo de las No Conformidades.

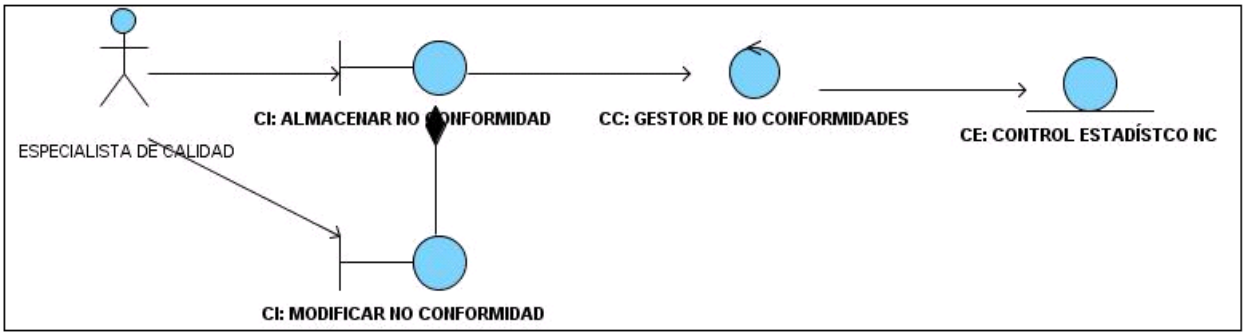


Figura 15: Diagrama de Clases de Análisis: Gestionar Control Estadístico de las No Conformidades.

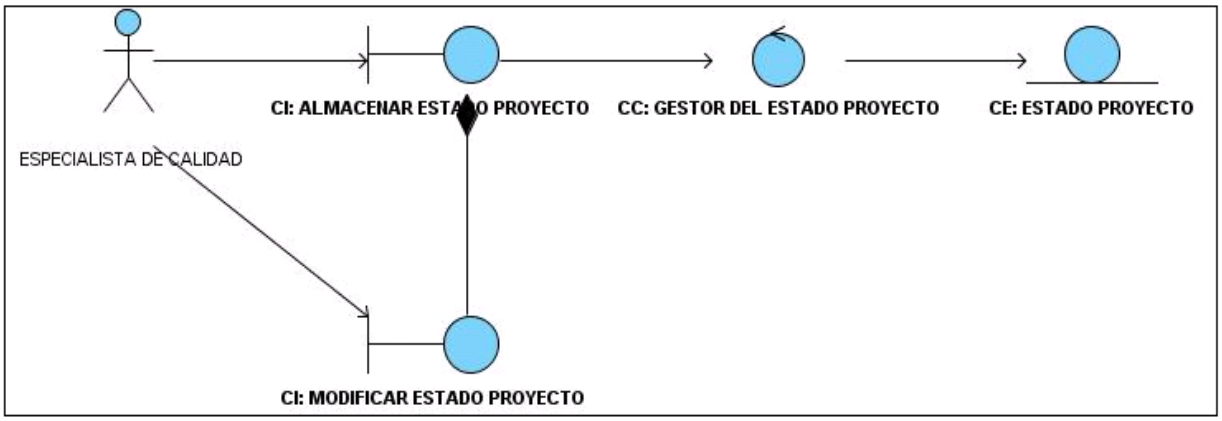


Figura 16: Diagrama de Clases de Análisis: Gestionar Estado del Proyecto.

Anexo 2: Diagramas de Colaboración del Diseño.

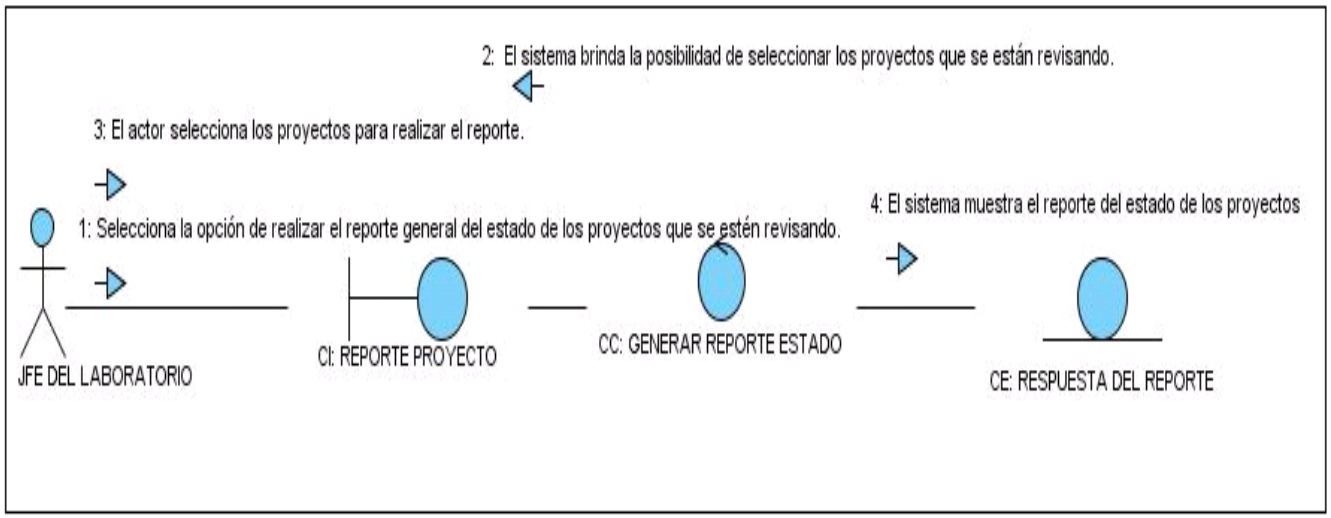


Figura 17: Diagrama de Colaboración del Diseño: Generar Reporte del Estado del Proyecto.

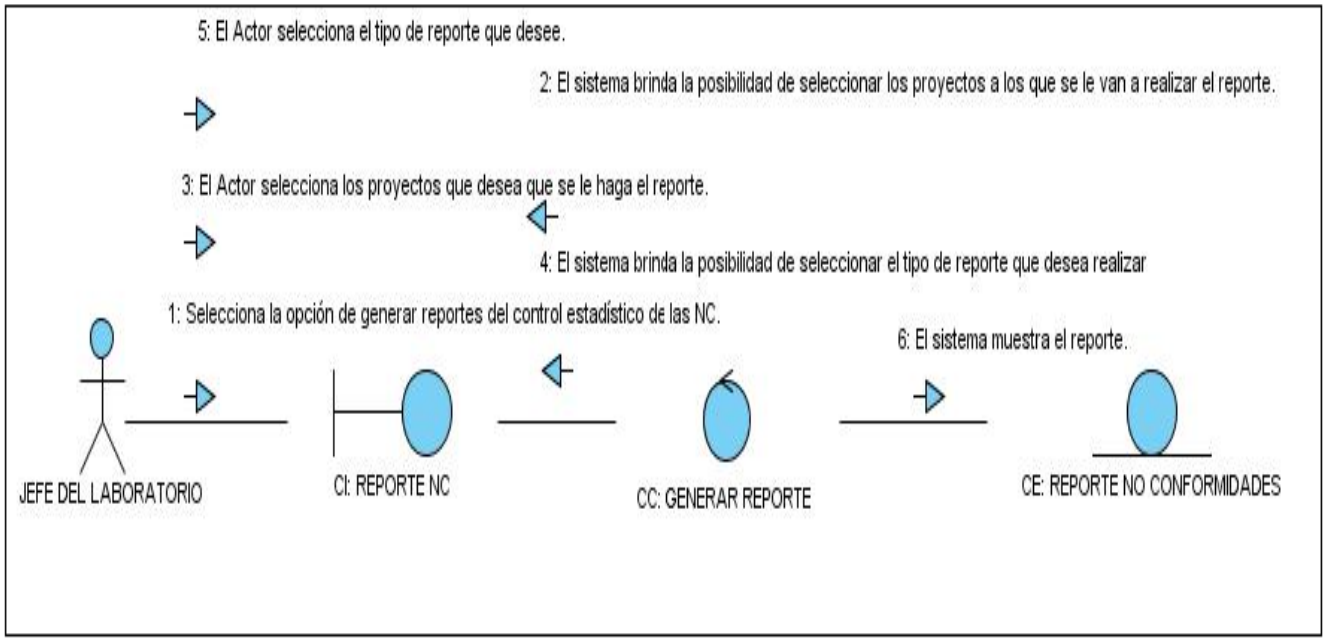


Figura 18: Diagrama de Colaboración del Diseño: Generar Reporte Cuantitativo de las No Conformidades.

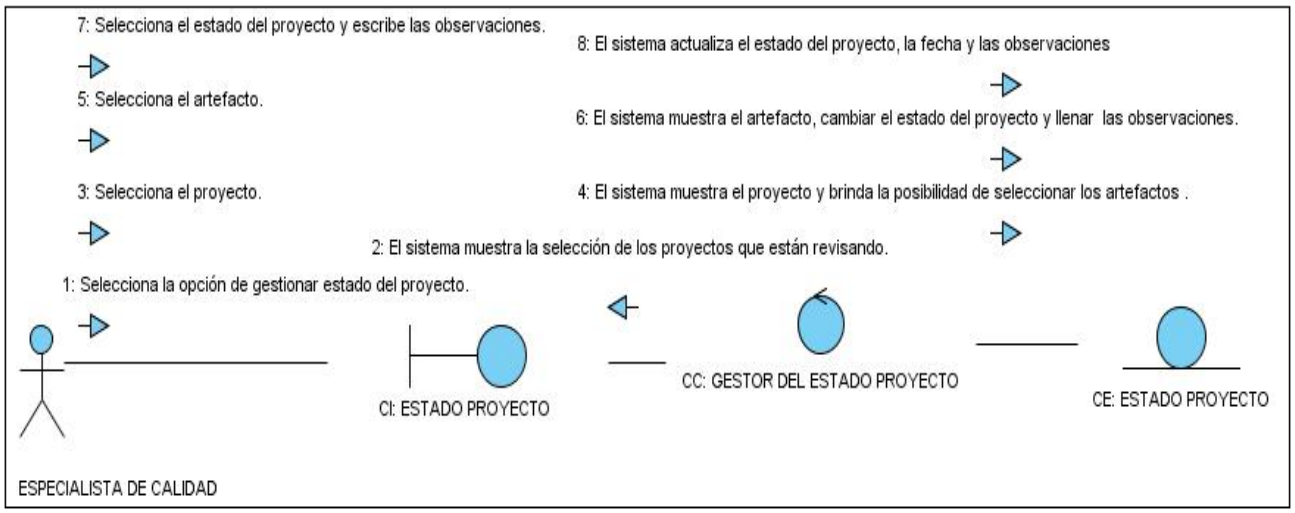


Figura 19: Diagrama de Colaboración del Diseño: Gestionar Estado del Proyecto.

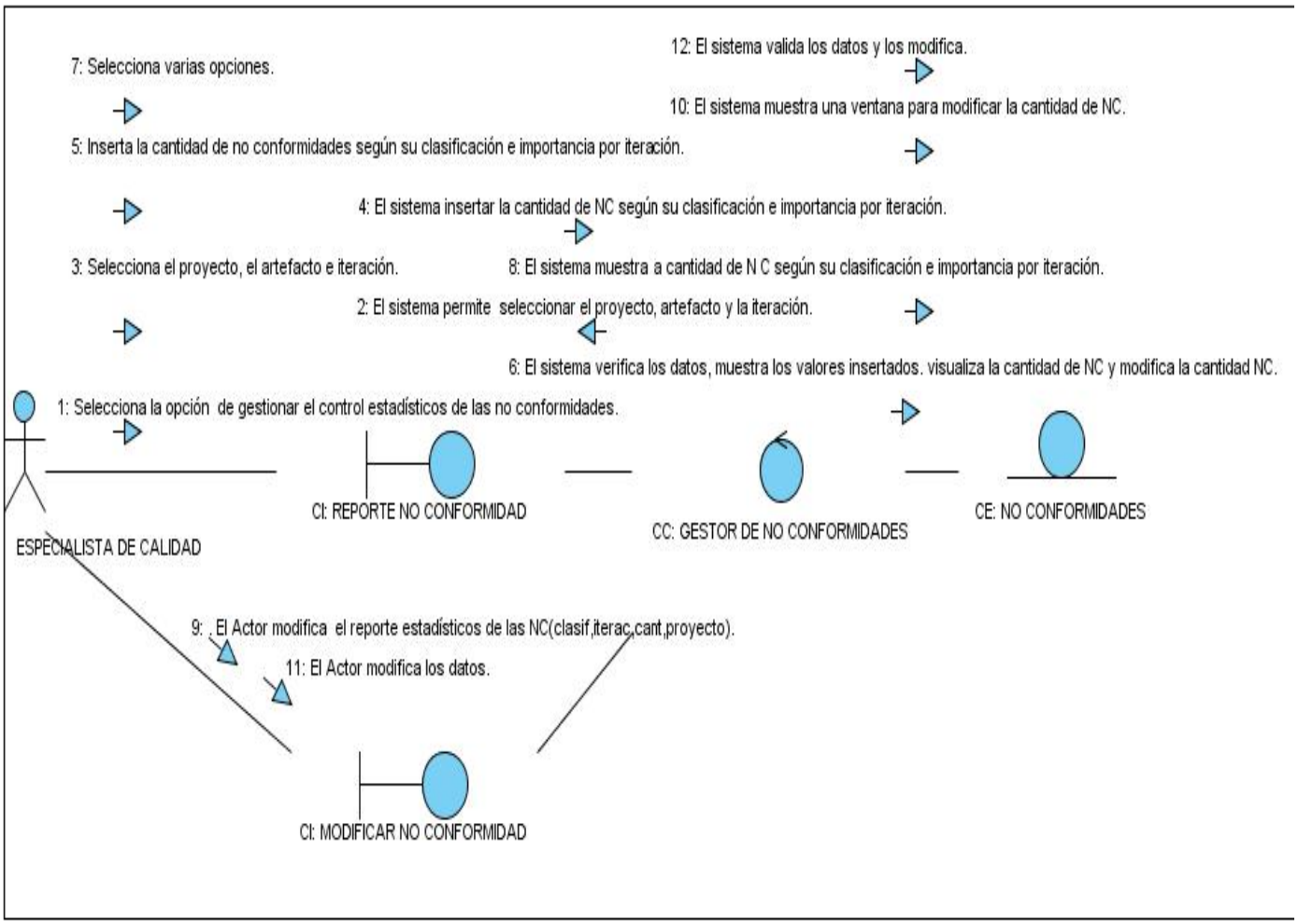


Figura 20: Diagrama de Colaboración del Diseño: Gestionar el Control Estadístico de las No Conformidades.

Anexo 3: Diagramas de Clases del Diseño.

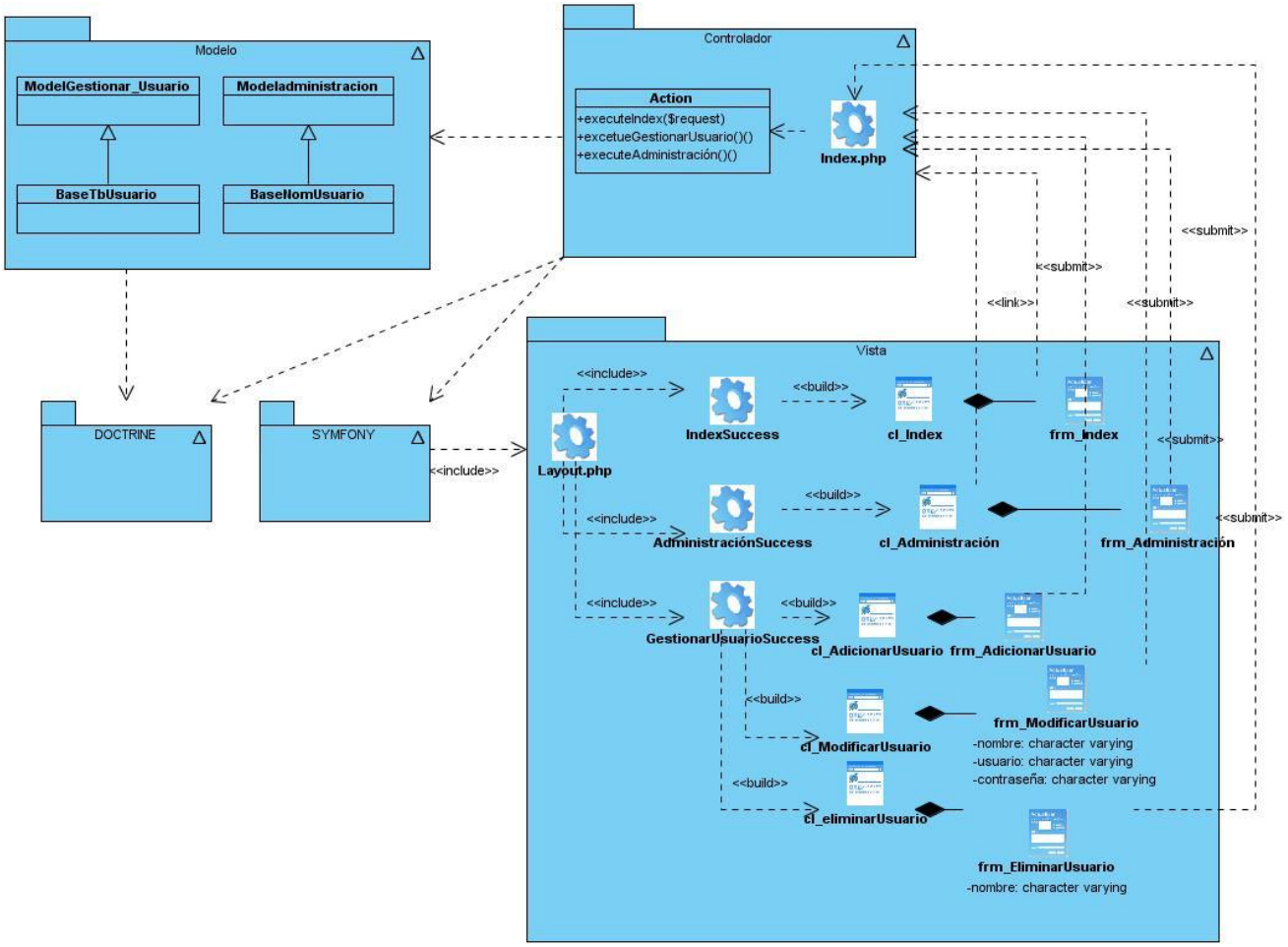


Figura 21: Diagrama de Clases del Diseño: Gestionar Usuario.

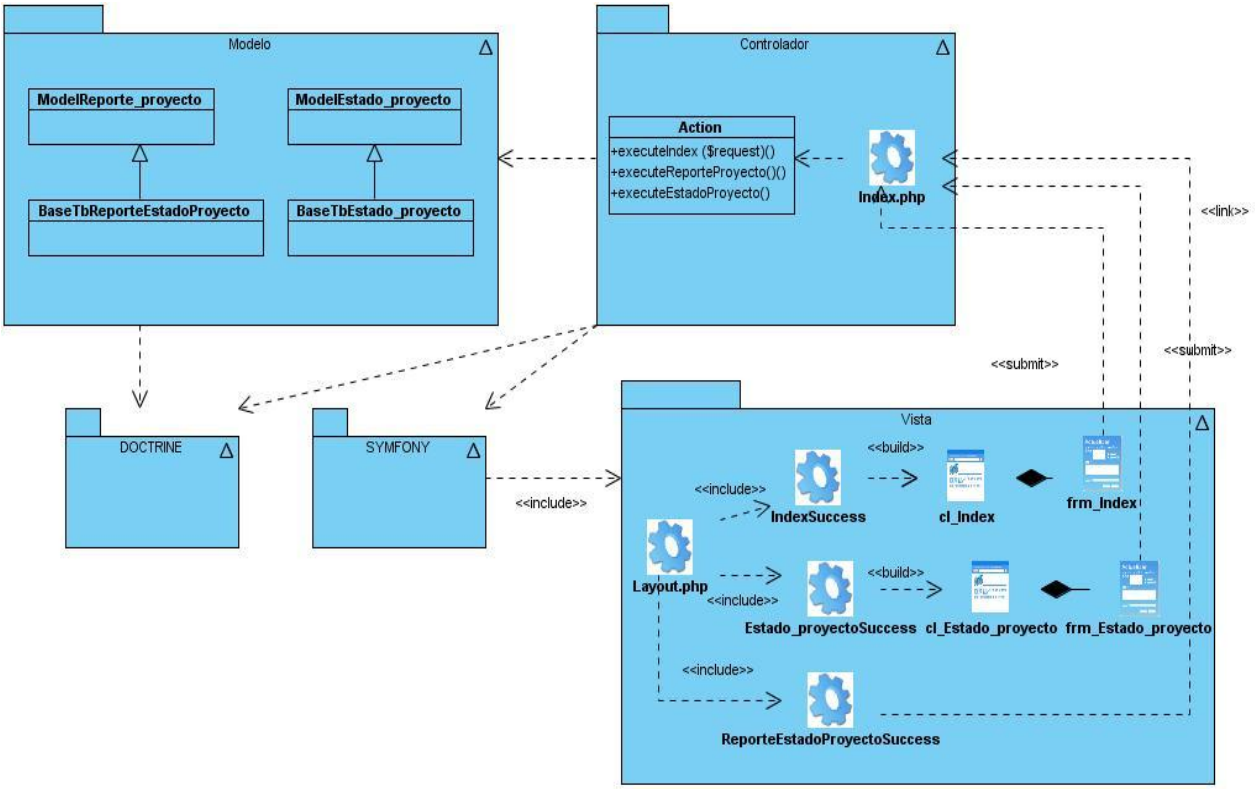


Figura 22: Diagrama de Clases del Diseño: Generar Reporte del Estado del Proyecto.

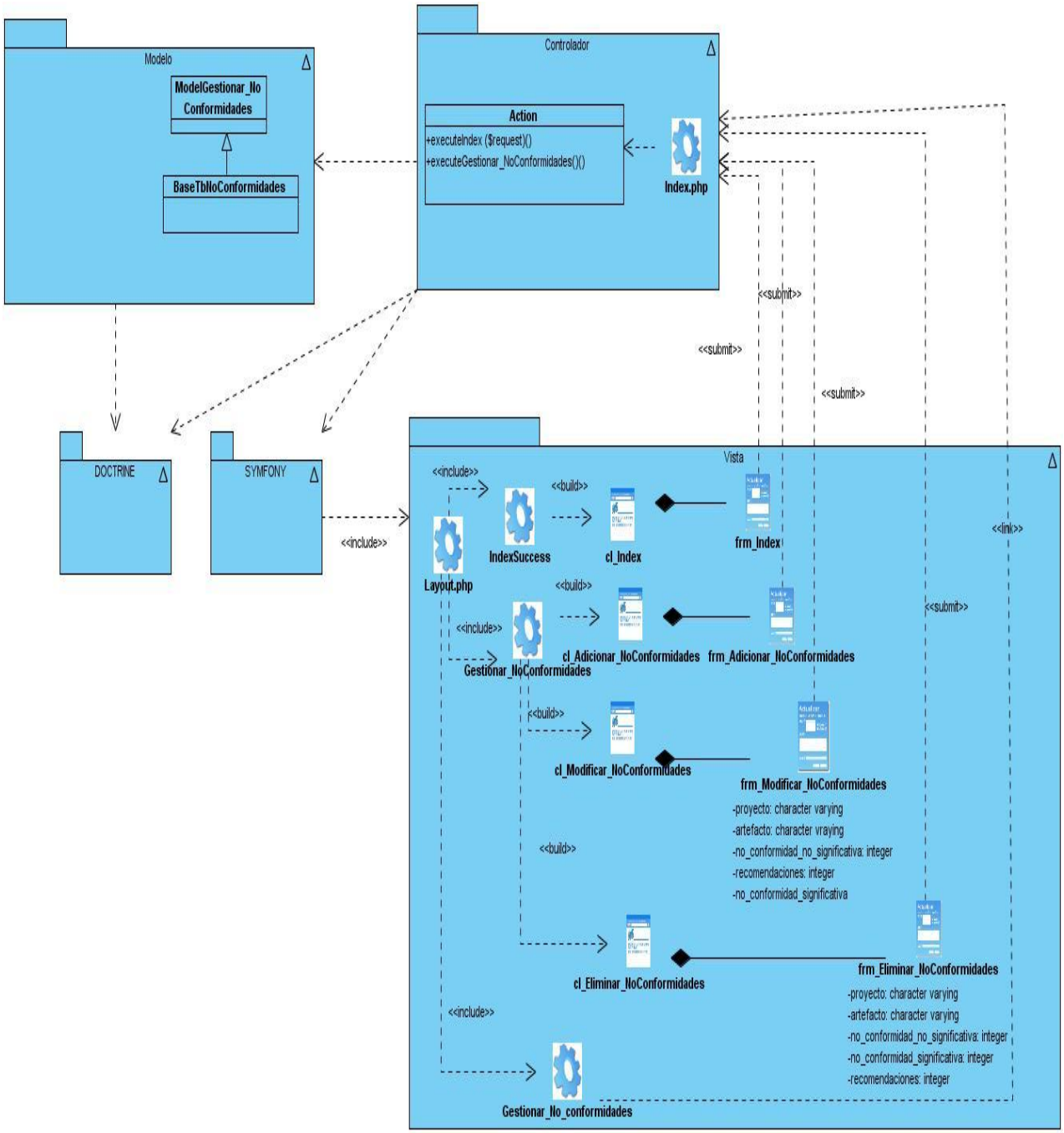


Figura 23: Diagrama de Clases del Diseño: Gestionar las Estadísticas de las No Conformidades.

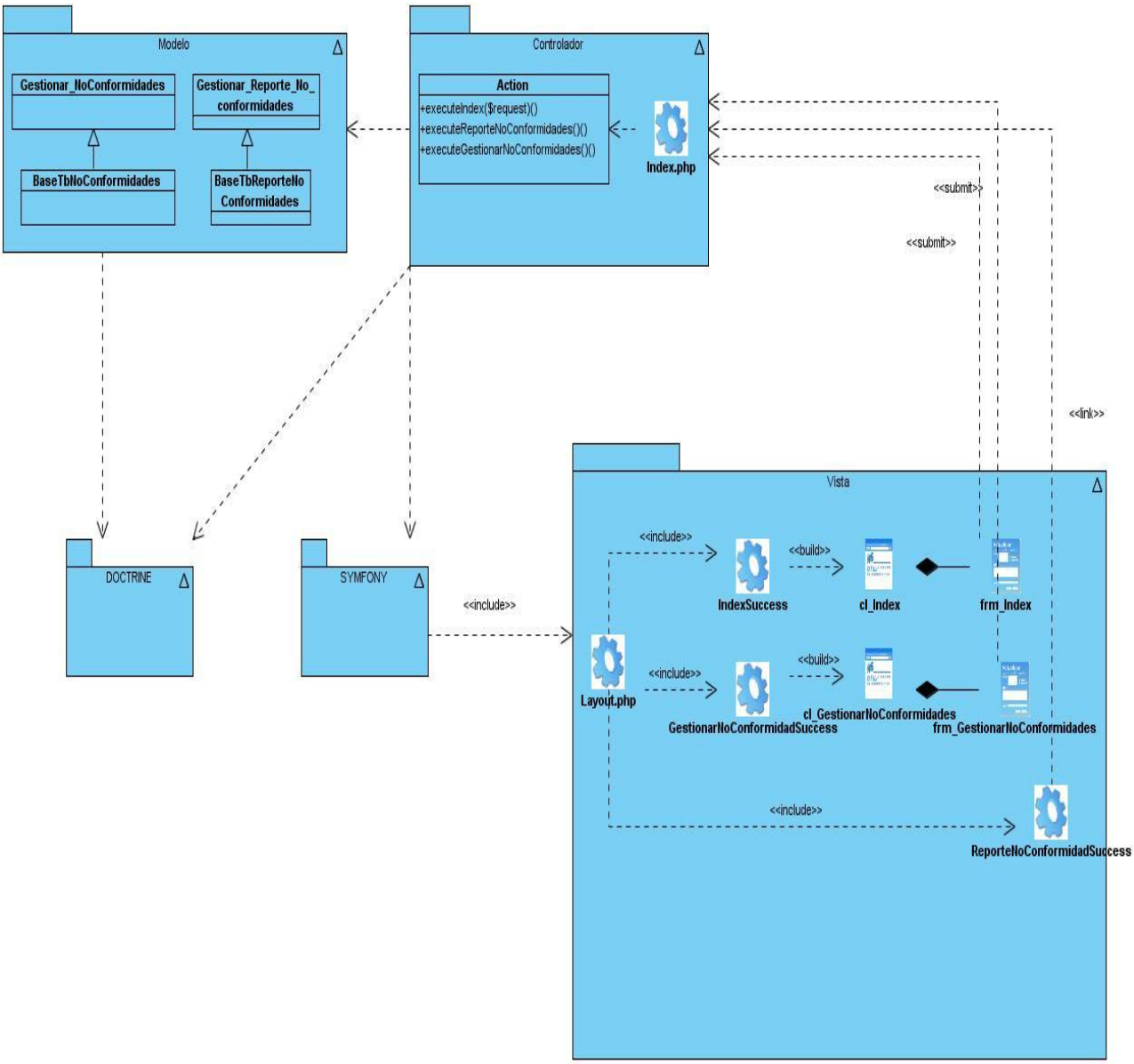


Figura 24: Diagrama de Clases del Diseño: Generar Reporte Estadístico de las No Conformidades.

Anexo 4: Diagrama de Clases persistentes.

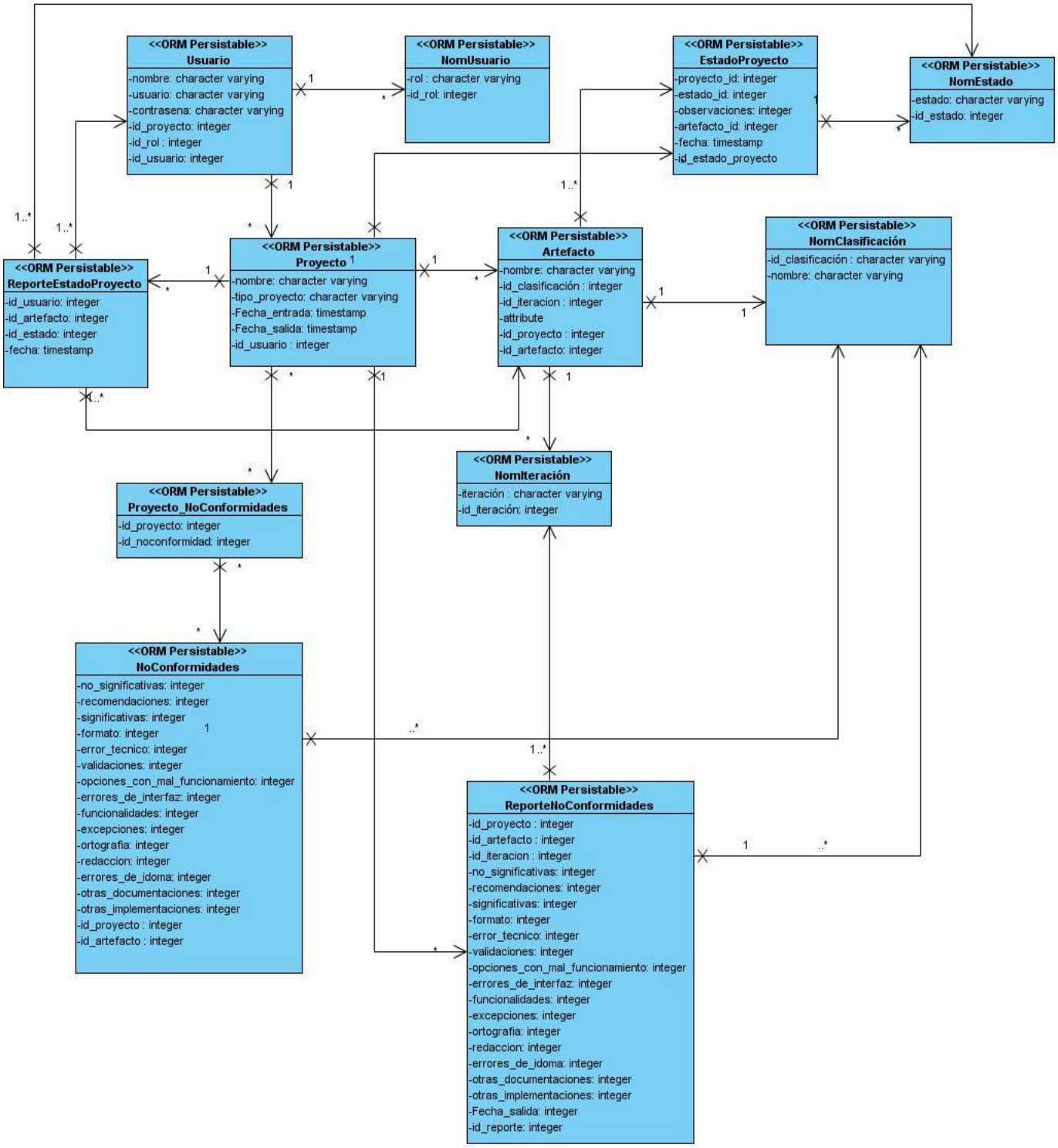


Figura 25: Diagrama de Clases Persistentes.

Anexo 5: Modelo Entidad Relación.

Figura 26: Modelo Entidad Relación.

Anexo 6: Descripción de las Tablas de la Base de Datos.

Tabla 9: Descripción de la tabla de la Base de Datos "Usuario".

Nombre: Usuario.		
Descripción: Se almacena los datos de los usuarios del sistema.		
Atributo	Tipo	Descripción
Id_usuario	INTIGER	Es la llave primaria y almacena el nombre de usuario relacionado con los especialistas de calidad y el jefe del laboratorio de pruebas.
Contraseña	VARCHAR	Almacena la contraseña que tiene el usuario.
Nombre	VARCHAR	Este campo almacena los nombres de los usuarios que se registren.

Tabla 10: Descripción de la tabla de la Base de Datos "Especialista de Calidad".

Nombre: Especialista de Calidad.		
Descripción: Se almacenan los datos del especialista.		
Atributo	Tipo	Descripción
Id_usuario	INTIGER	Es la llave foránea y almacena el identificador de cada usuario registrado.
Nombre	VARCHAR	Este campo almacena los nombres de los usuarios que se registren por especialistas de calidad.

Tabla 11: Descripción de la tabla de la Base de Datos "Rol".

Nombre: Rol		
Descripción: Se almacenan los roles que van a obtener los usuarios registrados.		
Atributo	Tipo	Descripción
Id_rol	INTIGER	Es la llave primaria y almacena el identificador del rol que van a ocupar los usuarios que se registren, ya sea el jefe del laboratorio ó los especialistas de calidad.
Nombre	VARCHAR	Este campo almacena los nombres de los roles que se registren por usuario.

Tabla 12: Descripción de la tabla de la Base de Datos "Rol_Especialista".

Nombre: Rol _Especialista.		
Descripción: Se almacenan los especialistas según el rol.		
Atributo	Tipo	Descripción
Id_usuario	INTIGER	Es la llave foránea y almacena el identificador de por cada usuario registrado ya sea el jefe del laboratorio ó los especialistas de calidad.
Id_rol	INTIGER	Este campo almacena el identificador de cada rol según los usuarios registrados.

Tabla 13: Descripción de la tabla de la Base de Datos "Proyecto".

Nombre: Proyecto.		
Descripción: Se almacenan todos los proyectos que se vayan a revisar.		
Atributo	Tipo	Descripción

Id_proyecto	INTIGER	Es la llave primaria y almacena el identificador de cada proyecto registrado.
Id_usuario	INTIGER	Es la llave foránea y almacena el identificador de cada usuario registrado.
Id_estado	INTIGER	Es la llave foránea y almacena el número por cada estado registrado.
Nombre	VARCHAR	Este campo almacena los nombres que se registren por proyecto.
Estado	VARCHAR	Este campo almacena los estados en el que se van a encontrar los proyectos registrados.
Observaciones	VARCHAR	Este campo almacena las observaciones realizadas, por cada proyecto.
Fecha	DATETIME	Este campo almacena la fecha en que se realizó la revisión, por proyecto.

Tabla 14: Descripción de la tabla de la Base de Datos "Reporte".

Nombre: Reporte.		
Descripción: Se almacenan los reportes realizados.		
Atributo	Tipo	Descripción
Id_reporte	INTIGER	Es la llave primaria y almacena el identificador de un reporte realizado por proyecto, después de ser revisado por el especialista de calidad.
Id_proyecto	INTIGER	Es la llave foránea y almacena el identificador de cada proyecto al cual se le va a realizar el reporte.
Fecha	DATETIME	Este campo almacena la fecha en que se realizó el reporte, por proyecto.

Cant_No_Conformidad	INTIGER	Este campo almacena la cantidad de No Conformidades encontradas por proyecto, para luego registrarlas en el reporte.
Iteración	INTIGER	Este campo almacena la cantidad de iteraciones que fueron llevadas a cabo para la revisión del proyecto.

Tabla 15: Descripción de la tabla de la Base de Datos "Estado".

Nombre: Estado.		
Descripción: Se almacena el estado en el cual se va a encontrar el proyecto.		
Atributo	Tipo	Descripción
Id_estado	INTIGER	Es la llave primaria y almacena el identificador del estado.
nombre	VARCHAR	Este campo almacena el nombre que lleva el estado.

Tabla 16: Descripción de la tabla de la Base de Datos "No Conformidad".

Nombre: No_Conformidad.		
Descripción: Se almacenan las No Conformidades encontradas por proyecto.		
Atributo	Tipo	Descripción
Id_No_Conformidad	INTIGER	Es la llave primaria y almacena el identificador de la No Conformidad encontrada según su clasificación.
Id_iIteración	INTIGER	Es la llave foránea y almacena el identificador de las iteraciones llevadas a cabo.
Id_clasificación_No_Conformidad	INTIGER	Es la llave foránea y almacena el identificador de la No Conformidad según su clasificación.

Id_artefacto	INTIGER	Es la llave foránea y almacena el identificador del tipo de artefacto que se registró.
Id_proyecto	INTIGER	Es la llave foránea y almacena el identificador del proyecto donde se encontró la No Conformidad.
Cantidad	VARCHAR	Este campo almacena el nombre que lleva el estado.
Clasificación	VARCHAR	Este campo almacena el tipo por el cual se van a clasificar las No Conformidades.
Tipo_Artefacto	VARCHAR	Este campo almacena el tipo de artefacto que se va a utilizar.
Iteraciones	*VARCHAR*	Este campo almacena las iteraciones que se van a llevar a cabo para la revisión del proyecto.

Tabla 17: Descripción de la tabla de la Base de Datos "Clasificación_No_Conformidad".

Nombre: Clasificación_No_Conformidad.		
Descripción: Se almacenan las No Conformidades según su clasificación.		
Atributo	Tipo	Descripción
Id_clasificaión_No_conformidad	INTIGER	Es la llave primaria y almacena el identificador del estado.
nombre	VARCHAR	Este campo almacena el nombre que lleva el estado.

Tabla 18: Descripción de la tabla de la Base de Datos "Tipo_Artefacto".

Nombre: Tipo_artefacto.
Descripción: Se almacenan los tipos de artefactos según el artefacto registrado.

Atributo	Tipo	Descripción
Id_Tipo_artefacto	INTIGER	Es la llave primaria y almacena el identificador del tipo de artefacto.
nombre	VARCHAR	Este campo almacena el nombre que lleva el tipo de artefacto.

Tabla 19: Descripción de la tabla de la Base de Datos "Iteración".

Nombre: Iteración.		
Descripción: Se almacenan las iteraciones del proyecto en revisión.		
Atributo	Tipo	Descripción
Id_iteración	INTIGER	Es la llave primaria y almacena el identificador de la iteración.
nombre	VARCHAR	Este campo almacena el nombre que lleva la iteración.

Anexo 7: Diagrama de Componentes

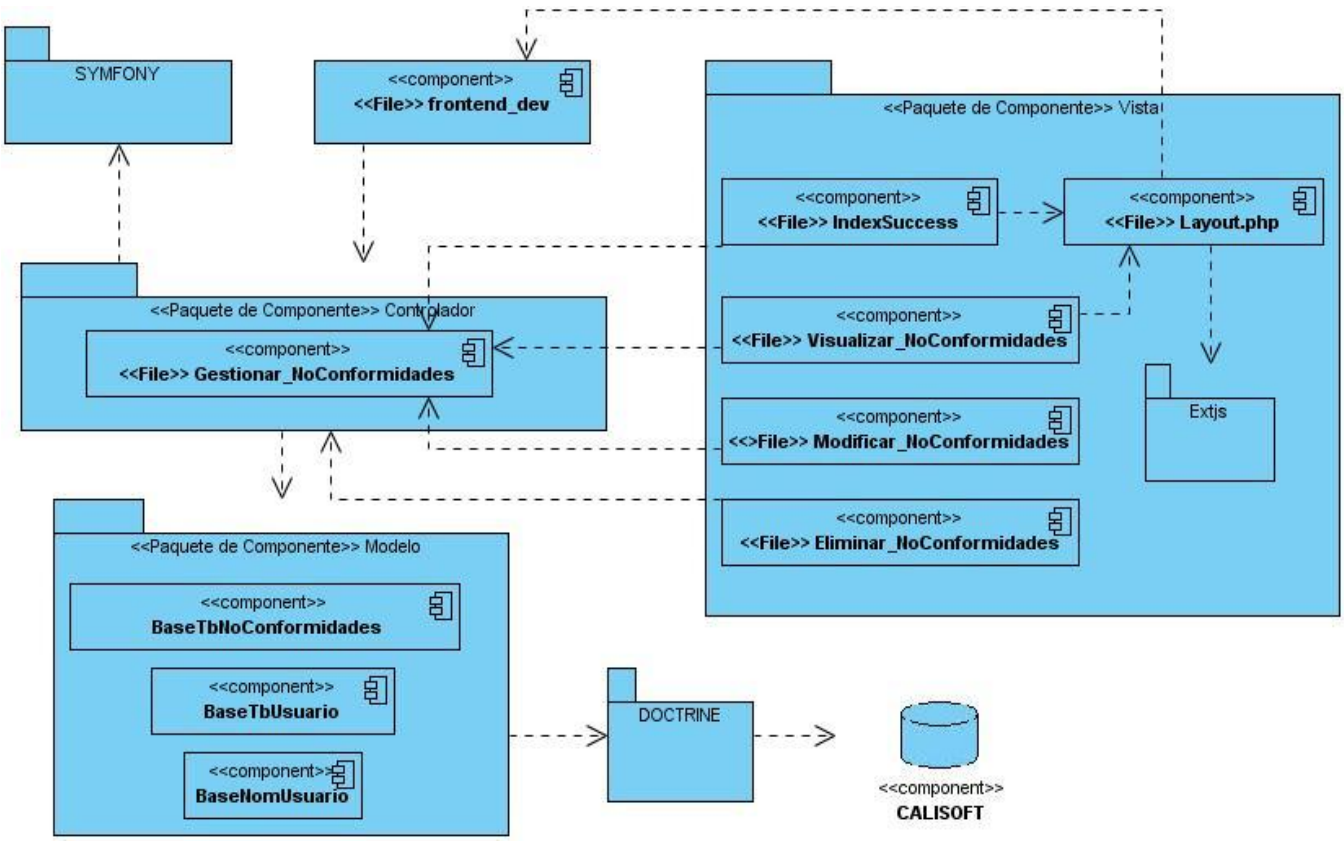


Figura 27: Diagrama de Componentes: Gestionar el Control Estadístico de las No Conformidades.

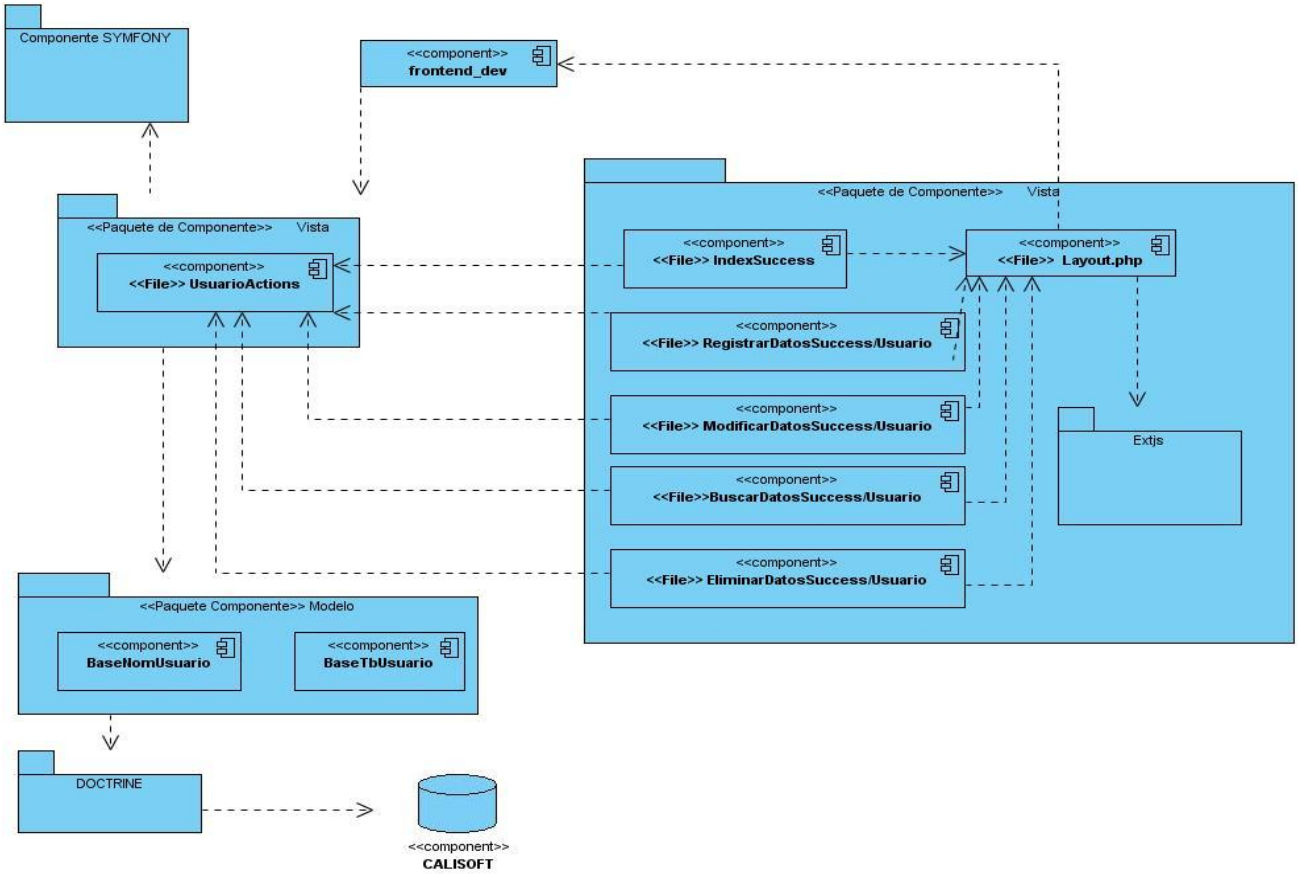


Figura 28: Diagrama de Componentes: Gestionar Usuario.

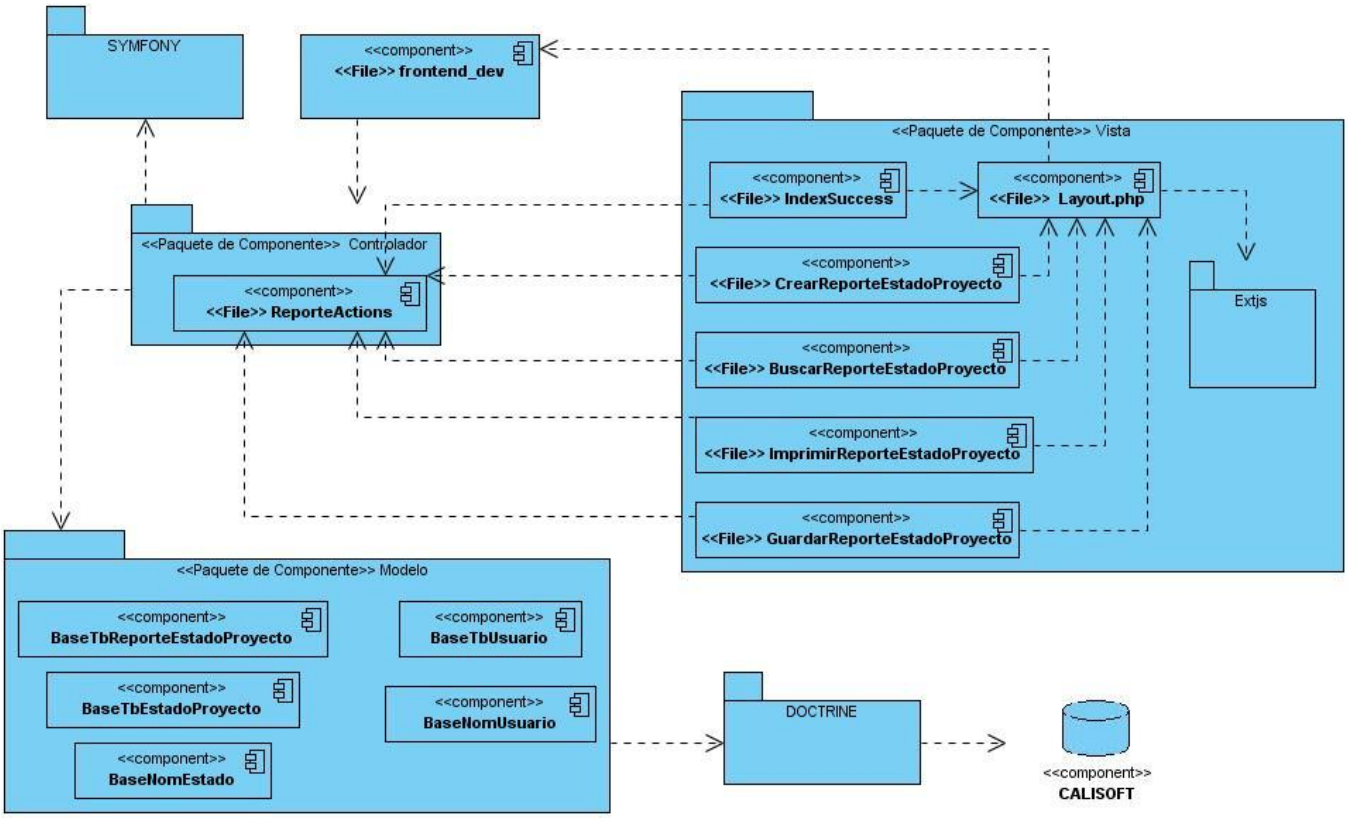


Figura 29: Diagrama de Componentes: Generar Reporte del Estado de los Proyectos.

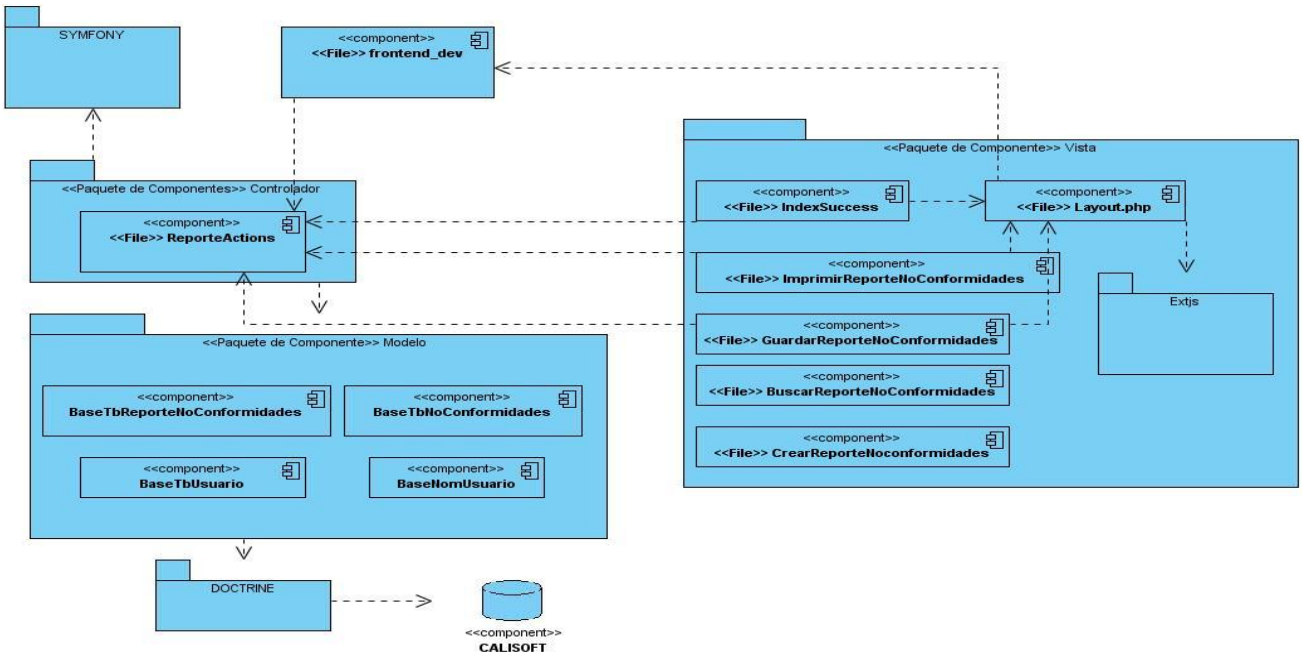


Figura 30: Diagrama de Componentes: Generar Reporte de las No Conformidades.

Anexo 8: Descripción de las secciones por escenarios.

Tabla 20: Secciones a probar en el Caso de Uso Gestionar clasificación de las No Conformidades.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Adicionar las clasificaciones de las No Conformidades.	EC 1.1: Agregar clasificación de las No Conformidades.	Se agregan en cantidades exactas las No Conformidades encontradas según sus clasificaciones.

	EC1.2: Existen campos incompletos	No se pueden agregar las No Conformidades encontradas según sus clasificaciones porque existen campos incompletos.
	EC 1.3: Existen campos incorrectos.	No se pueden agregar las No Conformidades encontradas según sus clasificaciones porque existen campos incorrectos.
SC 2: Modificar las NC.	EC 2.1: Modificar las No Conformidades.	Desea modificar las No Conformidades.
	EC 2.2: Existen campos incompletos.	No se pueden modificar puesto que existen campos incompletos.
	EC 2.3: Existen campos incorrectos.	No se pueden modificar puesto que existen campos incorrectos.
SC 3: Eliminar las NC.	EC 3.1: Eliminar No Conformidad.	Desea eliminar la No Conformidad.
	EC 3.2: Existen campos incompletos.	No se puede eliminar puesto que existen campos incompletos.
SC 4: Reportes acerca del Estado de los proyectos.	EC 4.1: Generar reporte exitosamente.	Se envía el reporte con el estado de los proyectos que son revisados.
SC5: Reportes estadísticos de las no Conformidades.	EC 5.1: Generar reporte exitosamente.	Se envía el reporte estadístico con las No Conformidades encontradas.

Tabla 21: Secciones a probar en el Caso de Uso Gestionar Proyecto.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Asignar proyectos.	EC1.1: Asignar un nuevo proyecto exitosamente.	Se le asignan los proyectos a los especialistas de calidad para realizarles las pruebas.
	EC1.2: Existen campos incompletos	No se pueden asignar los proyectos porque existen campos incompletos.
	EC 1.3: Existen campos incorrectos	No se pueden asignar los proyectos porque existen campos incorrectos.
SC 2: Modificar proyectos.	EC 2.1: Modificar proyecto.	Desea modificar algún proyecto.
	EC 2.2: Existen campos incompletos.	No se puede modificar el proyecto porque existen campos incompletos.
	EC 2.3: Existen campos incorrectos.	No se puede modificar el proyecto porque existen campos incorrectos.
SC 3: Eliminar proyectos.	EC 3.1: Eliminar proyecto.	Desea eliminar algún proyecto.
	EC 3.2: Existen campos incompletos.	No se puede eliminar el proyecto porque existen campos incompletos.
SC 4: Reportes sobre el Estado de los proyectos.	EC 4.1: Generar reporte exitosamente.	Se envía el reporte con el estado de los proyectos que son revisados.
SC5: Reportes estadísticos de las no Conformidades.	EC 5.1: Generar reporte exitosamente.	Se envía el reporte estadístico con las No Conformidades encontradas.

Tabla 22: Secciones a probar en el Caso de Uso Gestionar Artefacto.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Asignar artefacto.	EC 1.1: Asignar artefacto exitosamente.	Se le asigna un tipo de artefacto al proyecto que está siendo revisado.
	EC1.2: Existen campos incompletos	No se le puede asignar un tipo de artefacto al proyecto que está siendo revisado porque existen campos incompletos.
	EC 1.3: Existen campos incorrectos	No se le puede asigna un tipo de artefacto al proyecto que está siendo revisado porque existen campos incorrectos.
SC 2: Modificar artefacto.	EC 2.1: Modificar artefacto.	Desea modificar el artefacto.
	EC 2.2: Existen campos incompletos.	No se puede modificar el artefacto porque existen campos incompletos.
	EC 2.3: Existen campos incorrectos.	No se puede modificar el artefacto porque existen campos incorrectos.
SC 3: Eliminar artefacto.	EC 3.1: Eliminar artefacto.	Desea eliminar el artefacto.
	EC 3.2: Existen campos incompletos.	No se puede eliminar el artefacto porque existen campos incompletos.

SC 4: Reportes sobre el Estado de los proyectos.	EC 4.1: Generar reporte exitosamente.	Se envía el reporte con el estado de los proyectos que son revisados.
SC5: Reportes estadísticos de las no Conformidades.	EC 5.1: Generar reporte exitosamente.	Se envía el reporte estadístico con las No Conformidades encontradas.

Anexo 9: Resultados de la Prueba de Carga y Stress.

Tabla 23: Resultados de la Prueba de Carga y Stress.

Informe Agregado

Nombre: Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo Navegar... Log/Display Only: Escribir en Log Sólo Errores Successes

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento
/sf/sf_web_debug/images/warning.png	40	31	0	219	0	297	0,00%	3,7/sec
/sf/sf_web_debug/images/error.png	40	23	0	16	0	313	0,00%	3,7/sec
/images/Page-BgTexture.jpg	40	14	0	31	0	203	0,00%	3,8/sec
/images/nav.png	40	14	0	31	0	188	0,00%	3,8/sec
/sf/sf_web_debug/images/view.png	40	11	0	16	0	235	0,00%	3,8/sec
/images/Page-BgGradient.jpg	40	20	15	31	0	297	0,00%	3,8/sec
/images/rssIcon.png	40	16	0	16	0	296	0,00%	3,8/sec
/sf/sf_web_debug/images/info.png	40	112	0	47	0	3922	0,00%	3,8/sec
/sf/sf_web_debug/images/config.png	40	59	0	32	0	1828	0,00%	4,0/sec
/sf/sf_web_debug/images/time.png	40	6	0	16	0	63	0,00%	4,0/sec
/images/Footer.png	40	44	15	46	0	718	0,00%	4,0/sec
/images/item-left.png	40	24	0	31	0	328	0,00%	4,0/sec
/images/BlockHeaderIcon.png	40	7	0	16	0	31	0,00%	4,0/sec
/images/Button.png	40	14	15	31	0	203	0,00%	4,0/sec
/sf/sf_web_debug/images/memory.png	40	5	0	16	0	31	0,00%	4,0/sec
/images/BlockContent-s.png	40	13	0	16	0	219	0,00%	4,0/sec
/calisoft_dev.php/administracion/login	40	4541	4843	6969	282	7859	0,00%	2,5/sec
/calisoft_dev.php/Gestionar_artefactos	80	5528	5594	7078	1297	7656	0,00%	1,9/sec
/calisoft_dev.php/Gestionar_artefactos/new	40	5099	5343	6171	3093	7047	0,00%	1,4/sec
/calisoft_dev.php/Gestionar_artefactos/create	40	5249	5547	6547	3141	7016	0,00%	1,3/sec
/calisoft_dev.php/Gestionar_proyectos	80	6317	6328	7844	3656	8359	0,00%	1,5/sec
/calisoft_dev.php/Gestionar_proyectos/new	40	5701	5797	7188	3422	8203	0,00%	1,1/sec
/calisoft_dev.php/Gestionar_proyectos/create	40	6247	6375	7781	3750	8328	0,00%	1,1/sec
/calisoft_dev.php/Estado_proyecto	120	5933	6016	7125	3500	8374	0,00%	1,2/sec
/calisoft_dev.php/Estado_proyecto/edit/id/5	160	7461	7469	8922	4406	9734	0,00%	1,8/sec
/calisoft_dev.php/Estado_proyecto/update/id/5	80	9253	9390	10593	6546	11671	0,00%	1,1/sec
/calisoft_dev.php/Gestionar_NoConformidades	160	5521	5735	7250	781	7953	0,00%	1,8/sec
/calisoft_dev.php/Gestionar_NoConformidades/...	160	6348	6313	7937	3156	8641	0,00%	1,9/sec
/calisoft_dev.php/Gestionar_NoConformidades/...	80	6851	7094	8640	2515	9375	0,00%	1,1/sec

Glosario de Términos

CMMI: Capability Maturity Model Integration. Modelo para la mejora o evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. Fue desarrollado por el Instituto de Ingeniería del Software de la Universidad Carnegie Mellon.

• **CUN**: Casos de Uso del Negocio.

• **CUS**: Casos de Uso del Sistema.

• **Herramientas CASE**: Herramientas utilizadas para el desarrollo de proyectos de Ingeniería de Software.

• **IEEE**: Corresponde a las siglas de The Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos. Es una asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros de telecomunicaciones, ingenieros electrónicos, Ingenieros en informática.

• **ISO**: Organización Internacional para la Estandarización (Internacional Organization for Standarización). Es una organización que ha definido un conjunto de protocolos diferentes, llamados protocolos ISO/OSI. Esta organización de carácter voluntario fue fundada en 1946 y es responsable de la creación de estándares internacionales en muchas áreas, incluyendo la informática y las comunicaciones.

• **RF**: Requisitos Funcionales.

• **RNF**: Requisitos No Funcionales.