Universidad de las Ciencias Informáticas
Facultad 7

Título: Plataforma para la Gestión de la Calidad en el Centro de Informática Médica (CESIM)

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

Autores: Lisdany González Pardo Moralez
Yordan Cruz Betancourt

Tutores: Ing. Arian Seguí García
Ing. Yanitza Ramírez Estambor

Ciudad de La Habana, Junio 2010

"Año 52 de la Revolución"

Declaración de autoría

Declaramos ser los únicos autores del trabajo de diploma titulado: ***Plataforma para la Gestión de la Calidad en el Centro para la Informática Médica (CESIM)*** y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 28 días del mes de Junio del año 2010.

_____ _____

Lisdany González Pardo Morales. Yordan Cruz Betancourt.

_____ _____

Ing. Arian Seguí García. Ing. Yanitza Ramírez Estambor.

TUTOR: Ing. Arian Seguí García. Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el curso 2008-2009. Instructor recién graduado. Ha impartido la asignatura de Preparación para la Prueba de Nivel de Programación (PNP). Pertenece actualmente al Grupo de Calidad de la Facultad 7, desempeñando las funciones de Jefe de pruebas.

Correo electrónico: aseguí@uci.cu

TUTOR: Ing. Yanitza Ramírez Stambor, graduada de Ingeniería en Ciencias Informáticas en la UCI (2006-2007), Profesor Instructor. Ha impartido las asignaturas Matemática 1, Matemática 2, Teleinformática 1 y Teleinformática 2. Pertenece a la Subdirección de Calidad del Centro de Informática Médica (CESIM).

Correo electrónico: yramirez@uci.cu

Resumen

La Universidad de Ciencias Informáticas ha puesto en práctica varias estrategias para mejorar y agilizar la calidad de los productos, es por ello que se utilizan diversas herramientas como son DotProject, Subversion, RedMine y otras desarrolladas en el propio centro que facilitan la gestión de la información en cada unos de los proyectos. El Grupo de Calidad de la Facultad 7 perteneciente al CESIM ha organizado su estrategia de trabajo con el fin de controlar que los productos de la facultad cumplan con los requisitos de calidad exigidos.

El objetivo del presente trabajo de diploma es obtener un sistema informático que integre todos los procesos que se realizan en el Grupo de Calidad, lo que trae consigo que los productos cuenten con la calidad requerida.

El sistema ofrece significativas ventajas, pues con la integración de los diferentes módulos el trabajo se torna mucho más rápido y eficiente. Además se recoge gran cantidad de información redundante que hace difícil su utilización.

Con el uso de la aplicación, se facilitará el trabajo para el proceso de desarrollo de software en los proyectos productivos. La aplicación evita que exista información duplicada, por lo que garantiza la entrega en tiempo de la documentación.

Palabras claves: Integración, Gestión, Procesos de Calidad.

Índice

INTRODUCCIÓN

La tecnología informática hace posible encontrar nuevas vías para el desarrollo científico-técnico de la sociedad. Al mismo tiempo, la informática como ciencia, se encarga de apoyar el desarrollo de las tecnologías y por consiguiente se han logrado grandes cambios a nivel mundial en todas las esferas. A raíz de todos los cambios tecnológicos que se vienen produciendo, el hombre juega un papel muy importante, pues constituye el protagonista de los logros que se han alcanzado.

La actual industria del software se encuentra en un proceso de crecimiento, es por ello que se ha logrado alcanzar mayor producción en el desarrollo de aplicaciones. Los sistemas informáticos tienen múltiples usos y funcionalidades, todo esto provoca un aumento en el mercado internacional y la competencia en el mismo, por lo que se hace una necesidad de primer orden obtener el producto con la mejor calidad posible.

La calidad en el proceso de desarrollo de software ha tomado mayor importancia dadas las tendencias competitivas actuales y las exigencias de los usuarios. Se considera la calidad del software como el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. (1) De la misma manera, existen modelos de mejora de calidad como herramientas y marcos de referencia para garantizar la calidad máxima en el proceso de desarrollo de un software determinado. Entre los que se encuentra el proceso de mejora de software, (2) el cual, dentro de las iniciativas contemporáneas, es el más ampliamente utilizado.

Cuba no se encuentra ajena al avance de las tecnologías, teniendo como meta alcanzar un mayor crecimiento en la producción de software y a su vez que cuenten con la calidad requerida por el cliente y las reglas establecidas, tarea en la que se encuentran enfrascadas todas las instituciones vinculadas al desarrollo de software del país, entre ellas la Empresa Nacional de Software DESOFT, con sedes en todas las provincias del país, algunos Polos Científicos como son el de Villa Clara, Holguín y La Habana. Así surge la Universidad de las Ciencias Informáticas (UCI), creada en el marco de la Batalla de Ideas con el objetivo de impulsar el desarrollo de software en el país, se ha convertido en un centro por excelencia de referencia en la industria de software cubano.

La UCI, única en su tipo, combina la docencia con la producción y tiene como principal objetivo formar especialistas de alto nivel profesional, además de crear productos, servicios informáticos y soluciones

tecnológicas integrales, tanto para la informatización nacional como para la exportación. A poco tiempo de su creación, la UCI ya ha obtenido buenos resultados en el desarrollo de aplicaciones informáticas, por lo que constituye un reto, ganar en calidad y organización, en el proceso de producción. La universidad cuenta con varios proyectos productivos, los cuales han sido agrupados de acuerdo a las temáticas afines en diferentes centros de desarrollo.

La Facultad 7, cuenta con el Centro Especializado en Soluciones para la Informática Médica (CESIM), en el cual se desarrollan aplicaciones destinadas al sector de la salud; este se encuentra estructurado en distintos departamentos productivos los cuales son: Sistemas de Apoyo a la Salud (SAS), Atención Primaria de la Salud (APS), Software Médico Imageneológico (SWMI), Gestión Hospitalaria (GEHOS), además, cuenta con un Grupo de Calidad quien es el encargado de garantizar los procesos de aseguramiento, planificación y control de la calidad de los procesos y productos que se llevan a cabo en el mismo. Este cuenta con un nivel de organización interno por el cual se rigen para lograr mejores resultados en sus tareas y que estas cuenten con la calidad requerida.

Aun así, en el Grupo de Calidad del CESIM persisten algunos problemas para lograr la gestión de toda la información que se genera, ya que no existe un mecanismo que centralice todos los procesos que se realizan en el mismo. Existen aplicaciones que le han dado solución a la problemática de algunos de los procesos, pero estos no se encuentran integrados, actualmente están totalmente aisladas estas aplicaciones, creando problemas a la hora de unir toda la información que se genera en cada una de ellas, además, el resto de los procesos se realizan manualmente en documentos, por separado. Todo esto provoca que el trabajo sea engorroso, exista información duplicada, lo que prolonga el tiempo de entrega, y la calidad de la información que se genera tras los procesos fundamentales que se ejecutan tales como la planificación, las pruebas y las revisiones, entre otros, se ve afectada.

Teniendo en cuenta la situación planteada, el problema radica en la integración de los procesos que se desarrollan en el Grupo de Calidad del CESIM. El objeto de estudio consiste en el proceso de gestión de la calidad. Para esto se especifica como campo de acción el proceso de gestión de la calidad de software en el CESIM.

Para solucionar el problema planteado se define como objetivo general: desarrollar una plataforma que permita la integración de las herramientas existentes y los procesos que se realizan en el Grupo de Calidad del CESIM para garantizar la gestión de calidad de los productos del centro.

Para llevar a cabo el objetivo planteado se proponen las siguientes tareas de la investigación:

- Realizar un análisis de las herramientas que existen en el Grupo de Calidad para que puedan ser integradas a un mismo sistema.
- Definir los procesos del Grupo de Calidad a incluir en la plataforma.
- Analizar el flujo de trabajo de las herramientas de gestión administrativa más populares.
- Definir la arquitectura de la información que debe tener la plataforma.
- Realizar el diseño de una plataforma que centralice todos los procesos a desarrollar en el Grupo de Calidad.
- Diseñar la base de datos.
- Implementar la plataforma.

La investigación está estructurada en cuatro capítulos:

- Capítulo I: Fundamentación teórica, se describe la fundamentación teórica que sustenta el desarrollo de la investigación para el posterior desarrollo del sistema propuesto. Se plantean los principales conceptos y términos abordados en la investigación, se analiza el estado del arte del tema tratado, a nivel internacional, nacional y en la Universidad de las Ciencias Informáticas, de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad o en las que se apoya para darle solución al problema planteado.
- Capítulo II: Características del sistema, se especifican las características que debe tener el sistema, se hace una valoración del flujo actual de los procesos, realizándose un análisis crítico y valorativo de cómo se ejecutan estos, la información que se maneja en ellos y el objeto de automatización; aspectos que están relacionados con el objeto de estudio.
- Capítulo III: Análisis y diseño**,** se hace referencia al Modelo de Análisis y al Modelo de Diseño. Estos modelos contienen los diagramas de clases de diseño, los diagramas de interacción y la descripción de las clases de diseño.
- Capítulo IV: Implementación, se realiza la implementación, que contiene el diagrama de componentes para cada uno de los casos de uso, así como su representación gráfica. Se refleja una serie de explicaciones de la estructura interna de la aplicación y también códigos de los métodos más relevantes.

FUNDAMENTACIÓN TEÓRICA

En este capítulo se tratan temas relacionados con la gestión de procesos que lleva a cabo el Grupo de Calidad del CESIM para así contribuir a una mejor organización y calidad de los servicios que brinda. Se realiza un estudio de las herramientas que se utilizan para la gestión de los procesos fundamentales tales como la planificación, las revisiones, auditorías y el seguimiento de errores detectados en un software; también se tratan las razones para las cuales se hace necesaria la búsqueda de un mecanismo que integre todos lo procesos y no se vean por separado, además de justificar las metodologías y tecnologías a usar para la elaboración de la misma.

1.1. Calidad de software

La calidad del software es una preocupación a la que se dedican muchos esfuerzos. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir un software con la mejor calidad posible que cumpla, y si puede supere, las expectativas de los usuarios.

Se considera calidad de software la concordancia entre los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente. (3)

1.2. Procesos de calidad de software

En el proceso de gestión de calidad de software se definen los principios de la gestión de calidad presentados en la norma ISO 9004:2000, los cuales se enfocan a una estructura que permite que las organizaciones mejoren su rendimiento. Estos principios se originan en las mejores prácticas y la experiencia de numerosas compañías e instituciones internacionales. (4)

La ***gestión de la calidad de software*** es un conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades. El propósito de la gestión de

la calidad del software es entender las expectativas del cliente en términos de calidad, y poner en práctica un plan proactivo para satisfacer esas expectativas.

Desde el punto de vista de la calidad, la gestión de la calidad del software (CS) está formada por 4 partes, las cuales son: Planificación de la CS, Control de la CS, Aseguramiento de la CS y Mejora de la CS.

La ***planificación de la calidad del software*** es la parte de la gestión de la calidad encargada de realizar el proceso administrativo de desarrollar y mantener una relación entre los objetivos y recursos de la organización y las oportunidades cambiantes del mercado. En la planificación se debe determinar: el rol de la planificación, requerimientos de la CS, preparación de un plan de CS, implementación de un plan de CS y preparar un manual de calidad.

El ***control de la calidad del software*** son las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en 2 objetivos fundamentales: mantener bajo control un proceso y eliminar las causas de los defectos en las diferentes fases del ciclo de vida. Está formado por actividades que permiten evaluar la calidad de los productos de software desarrollados. El aspecto a considerar en el control de la CS es la “prueba del software”.

El ***aseguramiento de calidad del software*** es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza que el software necesita dados requisitos de calidad. El aseguramiento de la calidad del software engloba: un enfoque de gestión de calidad, métodos y herramientas de ingeniería del software, revisiones técnicas formales aplicables en el proceso de software, una estrategia de prueba multiescala, el control de la documentación del software y de los cambios realizados, procedimientos para ajustarse a los estándares de desarrollo del software y mecanismos de medición y de generación de informes.

La ***mejora de la calidad del software*** es la parte de la gestión de la calidad que contribuye, por medio de las mediciones, a los análisis de los datos y auditorías, a efectuar mejoras en la calidad del software.

Una ***auditoría de calidad*** tiene como objetivo mostrar la situación real para aportar confianza y destacar las áreas que pueden afectar adversamente esa confianza. Otro objetivo consiste en

suministrar una evaluación objetiva de los productos y procesos para corroborar la conformidad con los estándares, las guías, las especificaciones y los procedimientos. (5)

1.3. Herramientas para los diferentes procesos

Dentro del aseguramiento de la calidad se encuentran ***las pruebas***; las mismas juegan un papel importante pues son procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallas de implementación, calidad o usabilidad de un programa. Además de verificar productos complejos de forma efectiva, esto requiere de un proceso de investigación más que seguir un procedimiento al pie de la letra.

Existen algunas herramientas para la realización de pruebas al software entre las que se destacan:

JUnit: Framework de pruebas unitarias que permite crear, ejecutar y hacer reportes de estado de conjuntos de pruebas automatizadas. Es una herramienta multiplataforma y aporta un diseño simple. El sistema operativo que soporta es Windows Vista Ultimate. (6)

JMeter: Es un proyecto de Apache Jakarta que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios. Es un completísimo banco de pruebas para servidores y aplicaciones web. Las pruebas son configuradas jerárquicamente. Dicha herramienta se destaca por su versatilidad, estabilidad y por ser de uso gratuito. Además de ser usado como:

- Una herramienta de pruebas unitarias para conexiones de bases de datos con JDBC, FTP, LDAP, servicios web, JMS, HTTP y conexiones TCP genéricas.
- Un monitor.
- Una herramienta de "generación de carga". Aunque no es una descripción completa de la herramienta. JMeter soporta aserciones para asegurarse que los datos recibidos son correctos, por cookies de hilos, configuración de variables y una variedad de reportes.

Herramienta para viabilizar el proceso de pruebas de caja blanca: dicha herramienta aplica la Técnica del Camino Básico hasta la generación de caminos independientes. Es capaz además de comprobar el cumplimiento del código fuente, de los estándares de codificación establecidos y determinar los parámetros de calidad. Específicamente los resultados que obtiene son: la complejidad

ciclomática de cada función, la matriz del grafo de flujo y un conjunto de caminos independientes. Se pueden definir y comprobar estándares de codificación que abarca la mayoría de los estilos posibles en la escritura del código. Además, se obtienen y tabulan algunos parámetros de calidad como las diferentes variantes de la definición de líneas de código. (7)

1.3.1. Herramientas para el seguimiento de errores

Bugzilla: Con seguridad es el más conocido de todos los sistemas. Creado por Mozilla, permite tener un seguimiento de errores y cambios, comunicación entre los miembros del proyecto, enviar y revisar parches y manejar la garantía de calidad. Además, entre otras características, permite llevar un control de tiempo, distintas formas de autenticación, gráficos multilenguaje. Requiere Perl[1], MySQL o PostgreSQL, servidor web (Apache recomendado), módulos Perl específicos y Mail Transfer Agent[2]. (8)

Otra herramienta realizada para el seguimiento de errores es creada por el Grupo de Calidad de la Facultad 7 esta se denomina:

Herramienta para la revisión y seguimiento de errores de la documentación generada: dicha herramienta fue creada con el fin de obtener un sistema informático para la automatización del seguimiento de errores en el proceso de pruebas a la documentación, que favorezca el proceso de control y aseguramiento de la calidad de los productos de software desarrollados para así facilitar un marco de trabajo sencillo. (9)

1.3.2. Herramientas para la planificación

La planificación constituye un proceso de toma de decisiones para alcanzar un futuro deseado, teniendo en cuenta la situación actual y los factores internos y externos que pueden influir en el logro de los objetivos. (10)

Del proceso de planificación se pueden mencionar varias herramientas, entre dichas herramientas se encuentran:

[1] Lenguaje de programación web (Practical Extraction and Report Language).

[2] Agente de Transferencia de Correo

Gantt PV: es un programa gratuito, de apariencia sencilla y sin grandes complicaciones, para planificación de proyectos, descomposición, representación y seguimiento de tareas sobre diagramas de Gantt. Dispone de una serie de iconos desde los que puedes crear tareas nuevas, asignar recursos y hacer una previsión de fechas para la ejecución de cada tarea. Para poder utilizarlo se necesita sistema operativo: Mac OS X. (11)

DotProject: Algo más veterana esta solución en entorno web, ofrece un marco completo para la planificación, gestión y seguimiento de múltiples proyectos para clientes diferentes. Permite gestionar las tareas y fases de un proyecto, ofrece para ello una gran cantidad de módulos que permiten realizar la correcta planificación y gestión de las múltiples actividades que surgen.

RedMine: Es una herramienta de gestión de proyectos software con interfaz web. Una vez instalada, el administrador da de alta a los proyectos a través de la interfaz web; puede además, dar de alta a los desarrolladores y jefes de proyecto. Es utilizada para la administración de proyectos, desarrollada bajo licencia GNU–GPL (Licencia Publica General). Algunas de sus funcionalidades básicas son: calendario, diagrama de Gantt, gestión de tareas, equipo de desarrollo y creación de subproyectos.

1.3.3. Herramientas para revisiones

Otro de los procesos con los que se cuenta son ***las revisiones***, cuyo objetivo es evaluar un producto intermedio para ver que se ajusta a las especificaciones, para comprobar que el desarrollo se está haciendo de acuerdo con los planes y que los cambios en el producto se realizan adecuadamente. Las revisiones son técnicas estáticas que se aplican en varios momentos del desarrollo del software y sirven para detectar defectos que pueden así ser eliminados. En las revisiones se pretende:

- Señalar la necesidad de mejorar en el producto.
- Confirmar las partes de un producto que no es necesario mejorar.
- Conseguir un trabajo técnico de una calidad más uniforme

MGRSoft: Es un modelo que consta de un sistema de procesos que contempla procedimientos, roles y métricas y se basa en la utilización de la experiencia acumulada por la empresa en materia de revisiones que es almacenada en una base de conocimientos que posee toda la información referente al proceso de revisiones. Además, se incluye un conjunto de herramientas automatizadas integradas en un paquete que permite gestionar dicho conocimiento en los dos procesos definidos dentro del

modelo. La base de conocimientos contiene la información referente a las revisiones que permite contar con la experiencia acumulada en este proceso para que pueda ser utilizada en el desarrollo y revisión de nuevos proyectos. (12)

1.3.4. Herramientas para auditorías

Dentro del control de la calidad se encuentran ***las auditorías de software,*** las cuales son un término general que se refiere a la investigación y al proceso de entrevistas que determina cómo se adquiere, distribuye y usa el software en la organización. La auditoría de calidad es una herramienta de gestión empleada para verificar y evaluar las actividades relacionadas con la calidad en el seno de una organización; además constituye una herramienta gerencial utilizada para evaluar, confirmar o verificar las actividades relacionadas con la calidad de un producto o servicio. Una auditoría de calidad, si es utilizada de una manera adecuada, es un proceso positivo y constructivo. (13)

1.3.5. Herramientas para la gestión de la calidad

Las tecnologías permiten actualmente la producción de productos de una forma tan rápida, que es incompatible cualquier descuido que permita la aparición de defectos. Por eso fue necesario implementar sistemas de gestión de la calidad para garantizar la calidad de los productos. Inicialmente, la implementación de sistemas de calidad produce una formalización muy grande de los procesos de producción y garantiza la organización y control efectivo de los productos. Es por ello que algunas empresas se han dado la tarea de desarrollar productos donde integran los procesos fundamentales de la gestión de la calidad para así contribuir a mejorar y agilizar el trabajo. Siendo así, se cita como ejemplo una plataforma desarrollada para la gestión de proyectos; *Knowledge Management Key* (KMKey) es un software libre de gestión del conocimiento vía web que agrupa tres productos bajo una misma plataforma:

- KMKey Project: para la planificación, gestión y control de sus proyectos.
- KMKey Quality: integra el Sistema de Gestión de la Calidad de la organización.
- KMKey HelpDesk: donde apoyar el servicio de atención al cliente y mantenimiento.

KMKey Project es un software de gestión de proyectos donde cualquier empresa puede disponer de toda la información necesaria para desarrollar su negocio, desde la oferta hasta la entrega del proyecto. KMKey Project es un software especialmente indicado para llevar el control de proyectos de

cualquier tipo: desarrollo de proyectos de ingeniería, gestión de despachos de arquitectura, planificación, seguimiento y control de obras, proyectos en tecnologías de la información, gestión de consultorías, ingeniería medioambiental, son algunas de las funcionalidades que actualmente son trabajadas con KMKey Project.

KMKey Quality es un software de gestión de calidad ideal para la implantación y mantenimiento de un sistema de gestión de calidad de cualquier tipo: ISO 9001, ISO 14001, OHSAS 18001, o de una combinación de los mismos, facilitando su integración. Mediante ***KMKey Quality*** se podrá gestionar y mantener la documentación del sistema, los registros, y los flujos de información propios que se generan en acciones como la gestión de No Conformidades, Acciones Correctivas/Preventivas, Reclamaciones, Auditorías, Indicadores, Evaluaciones.

KMKey Help Desk es un software de gestión de incidencias indicado para servicios de mantenimiento, ayuda al usuario y resolución de problemas en cualquier sector. Permite definir flujos de trabajo para abordar problemáticas derivadas de anomalías en servicios y maquinaria.

Las herramientas existentes para los diferentes procesos que se realizan en la gestión de calidad le dan solución satisfactoriamente a cada uno de ellos, pero están desarrollados en diferentes tecnologías por lo que los artefactos que se producen, por lo general, no son compatibles entre sí y a la hora de agrupar estos, existen algunos problemas como lo son: documentación redundante, abundante información innecesaria junto con la que realmente sí es importante, los documentos generados no poseen un formato estándar, la información correspondiente a un proceso está aislada de otros que necesiten de la misma y se necesita, para poder trabajar en todos ellos, tener en ejecución todas estas aplicaciones.

Debido a los problemas planteados se hace necesario crear un mecanismo en el cual se integren todos los procesos que se realizan para agilizar el trabajo que se lleva a cabo y este cuente con la calidad requerida.

1.4. Necesidad de la realización de una herramienta

Con el objetivo de agilizar y organizar el trabajo que se desarrolla en los proyectos, la Facultad 7 específicamente el grupo de Calidad del (CESIM) se trazó una estrategia para garantizar la calidad de los procesos que se realizan en dicho grupo; muchos de estos procesos se realizan de forma manual y

otros se ejecutan por medio de las herramientas antes mencionadas pero como se ha dicho en varias ocasiones estas trabajan aisladas unas de las otras creando varios problemas a la hora de unir todo la información generada. Es por ello que se hace inminente la creación de un mecanismo que basado en los problemas existentes permita visualizar de una forma sencilla y rápida al usuario la información necesaria en dependencia del rol que desarrolle, para así ganar en eficiencia y calidad.

1.5. Tecnologías y herramientas

1.5.1. Lenguaje de programación: Python

Es un lenguaje de programación de propósito general, cuya expansión y popularidad es relativamente reciente. Se trata de Python, una apuesta por la simplicidad, versatilidad y rapidez de desarrollo; Python es un lenguaje de *scripting* independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. En realidad, sí se realiza una compilación, pero esta se realiza de manera transparente para el programador. En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.

La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C. La cantidad de plataformas en las que se puede desarrollar, como Unix, Windows, OS/2, Mac y otros. Además, Python es gratuito, incluso para propósitos empresariales.

Tiene como propósito general crear todo tipo de programas. No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas. Es un lenguaje multiplataforma donde existen versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él. En esencia, significa que el programa, que se realice en Gnu/Linux, funciona sin ningún tipo de limitación en un entorno Windows o Mac Os. Lo anterior brinda una gran facilidad a la hora de portar código a diferentes sistemas

operativos. Si todo esto no es suficiente, Python, es compatible con la licencia GPL (Licencia Pública General), con lo cual, los programas que se creen, no estarán limitados por ningún tipo de licencia.

Ventajas

- Python es un lenguaje muy "expresivo", los programas son muy compactos, un programa en Python suele ser bastante más corto que su equivalente en lenguajes como C; es considerado por muchos, un lenguaje de programación de muy alto nivel.
- Python es muy legible, la sintaxis es muy elegante y permite la escritura de programas cuya lectura resulta fácil, en comparación con otros lenguajes.
- Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objeto.
- Python es un muy buen lenguaje para empezar a programar.
- Una ventaja fundamental de Python es la gratuidad de sus intérpretes.

1.5.2. Entorno de desarrollo

Wing IDE: Es un entorno integrado de desarrollo especialmente diseñado para el lenguaje de programación Python, disponible en Linux y en Windows. Permite el desarrollo rápido de aplicaciones de plataforma cruzada para escritorio, web y empresariales. Wing IDE se enfoca en incrementar la productividad y la calidad del código, especialmente en proyectos complejos con requerimientos cambiantes. Entre las principales características del Wing IDE se encuentran:

- Autocompletado de código.
- Asistente de codificación.
- Identificación automática.
- Búsquedas en múltiples archivos.
- Búsquedas en todo el disco.
- Búsquedas con expresiones regulares.
- Soporte de CVS, Subversión.
- Sintaxis coloreada.
- Evaluación de archivos o selecciones de archivo.
- Gratuitos para desarrollos Open Source.

1.5.3. Sistema Gestor de Base de Datos

PostgreSQL es un sistema administrador de bases de datos relacionales con licencia de software libre. Es un servidor de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD40.

Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*). (14)

Principales características:

- Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente Multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Gracias a su licencia BSD, se permite la utilización del código para ser comercializado. Uno de los casos ejemplo es la de Enterprise DB (PostgreSQL Plus), la cual incluye varios agregados y una interfaz de desarrollo basada en Java. Entre otras empresas que utilizan PostgreSQL para comercializar se encuentra CyberTech (Alemania), con su producto CyberCluster.

1.5.4. Metodología de desarrollo de software

RUP: El Proceso Unificado Racional (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El ciclo de vida RUP es una implementación del desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

Las fases son: Inicio, Elaboración, Construcción y Transición. Y nueve flujos de trabajo, seis de ingeniería (Modelado del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue) y tres de apoyo (Gestión de la configuración, Gestión de proyecto y Ambiente).

RUP pretende implementar las mejores prácticas actuales en ingeniería de software:

- Desarrollo iterativo del software.
- Administración de requerimientos.
- Uso de arquitecturas basadas en componentes.
- Modelación visual del software.
- Verificación de la calidad del software.
- Control de cambios.

1.5.5. Framework: Django

Es un framework de desarrollo web que ahorra tiempo y permite crear y mantener aplicaciones web de alta calidad con un mínimo esfuerzo. Django es un marco de trabajo de alto nivel, para el desarrollo de aplicaciones web en el lenguaje de programación Python. Entre las principales características de Django se encuentran:

- Mapeo Objeto-Relacional.
- Interfaces de administración automáticas listas para ambientes de producción.
- Diseño de URLs elegante.
- Sistema de plantillas.
- Sistema de *cache*.
- Soporte de internacionalización.

La arquitectura Modelo Vista Controlador surgió como patrón arquitectónico para el desarrollo de interfaces gráficas de usuario en entornos Smalltalk.

Su concepto se basa en la necesidad de reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, mediante la división de la aplicación en tres partes fundamentales:

- El modelo, que contiene la lógica de negocio de la aplicación.

- La vista, que muestra al usuario la información que este necesita.
- El controlador, que recibe e interpreta la interacción del usuario, actuando sobre modelo y vista de manera adecuada para provocar cambios de estado en la representación interna de los datos, así como en su visualización.

Estas tres piezas juntas, la lógica de acceso a la base de datos, la lógica de negocios, y la lógica de presentación comprenden un concepto, que a veces es llamado, el patrón de arquitectura de software Modelo-Vista-Controlador (MVC). En este patrón, el modelo hace referencia al acceso a la capa de datos, la vista se refiere a la parte del sistema que selecciona qué mostrar y cómo mostrarlo, y el controlador implica la parte del sistema que decide qué vista usar, dependiendo de la entrada del usuario, accediendo al modelo si es necesario.

Django sigue el patrón MVC tan al pie de la letra que puede ser llamado un framework MVC. Someramente, el Modelo, la Vista y el Controlador se separan en Django de la siguiente manera:

- ***Modelo***, la porción de acceso a la base de datos, es manejada por la capa de la base de datos de Django.
- ***Vista***, la porción que selecciona qué datos mostrar y cómo mostrarlos, es manejada por la vista y las plantillas.
- ***Controlador***, la porción que delega a la vista dependiendo de la entrada del usuario, es manejada por el framework.

Debido a que el ***Controlador*** es manejado por el mismo framework y la parte más emocionante se produce en los modelos, las plantillas y las vistas, Django es conocido como un Framework ***MTV***. En el patrón de diseño MTV:

- ***M*** significa (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- ***T*** significa (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.

- ***V*** significa (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre los modelos y las plantillas. (15)

1.6. Tecnologías a usar

El lenguaje que se va a utilizar es Python ya que sin duda su código es el más sencillo de todos. Los programas en Python suelen ser entre 2 y 4 veces más cortos que sus equivalentes en Java, no sólo por la simplicidad de su sintaxis, sino también por lo avanzado de ciertos tipos (listas, tuplas, diccionarios) que implementa y es un lenguaje multiplataforma.

En esencia, significa que el programa, que se realice en Gnu/Linux, funciona sin ningún tipo de limitación en un entorno Windows o Mac Os; si todo esto no es suficiente. Como sistema gestor de base de datos se utilizará PostgreSQL por su rapidez cuando se trabaja con una base de datos pequeña.

El entorno de desarrollo Wing IDE***,*** diseñado para Python permite un desarrollo rápido de aplicaciones de plataforma cruzada para escritorio, web, en este caso para una plataforma web.

RUP es la metodología estándar más utilizada para el diseño implementación y documentación del sistema y se basa a su vez en UML. Es altamente configurable, ya que permite construir solamente los artefactos que se necesiten para el desarrollo de un producto, iterativo e incremental, centrado en la arquitectura y dirigido por casos de usos, ellas son sus principales características.

Como resultado del análisis realizado durante el presente capítulo se concluye que: las herramientas estudiadas no responden a las necesidades del Grupo de Calidad de la Facultad 7, debido a esto se necesita implementar un sistema informático para mejorar la calidad y rapidez del trabajo desarrollado. Se seleccionó dentro de un grupo de tecnologías, herramientas y metodologías las adecuadas para realizar el software que se requiere en la universidad: Python como lenguaje de programación, Wing IDE como entorno de desarrollo, Framework Django, RUP como metodología de desarrollo y se determinó como arquitectura Modelo Vista Controlador.

CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se tratan los aspectos fundamentales relacionados con el objeto de estudio. Se explican los procesos del negocio sobre los que se basa la organización para su funcionamiento, se describe el flujo actual de los procesos y se realiza un análisis crítico de cómo estos se realizan actualmente -incluyendo toda la información que se maneja, o sea documentos específicos que se procesan-.

Además se presenta el objeto de automatización. Se aborda el tema relacionado con la modelación del negocio, siendo este el primer flujo de trabajo durante el proceso de desarrollo de un software, el cual tiene entre sus objetivos comprender los procesos de negocio de la organización, identificar las mejoras potenciales y señalar puntualmente los requerimientos del sistema que va a soportar la organización. Se muestra la especificación de los requisitos de software (los requisitos funcionales y no funcionales), además de la descripción de los actores y casos de uso del sistema.

2.1. Análisis crítico del proceso actual de la situación problémica

El Grupo de Calidad del CESIM, desempeña diversas tareas relacionadas con la revisión y entrega de productos de alta calidad, siendo el tiempo un factor fundamental e imprescindible para la entrega del software y que a su vez satisfaga los requerimientos planteados por el cliente.

Actualmente los artefactos que se generan, son desarrollados en su mayoría por los distintos roles que conforman el Grupo de Calidad; este grupo ha desarrollado algunas herramientas que automatizan algunos de estos procesos obteniendo los artefactos en menor tiempo y con mejor calidad, además se han realizado algunos estudios sobre algunas herramientas de nivel internacional que pudiesen ser utilizadas y aún se hace uso del paquete de Office. Muchas de ellas no son compatibles entre sí, y al no estar integradas entre ellas gestionan por separado la información que utilizan y generan como

resultado final. A la hora de reunir todos los artefactos generados, existen algunos problemas, como son:

- Información duplicada.
- Abundante información innecesaria junto con la que realmente es importante.
- Los documentos generados no poseen un formato estándar.
- La información correspondiente a un proceso está aislada de otros que necesiten de la misma.
- Se necesita, para poder trabajar en todos los procesos, tener en ejecución varias aplicaciones.

Todo esto provoca que los probadores no se sientan satisfechos a la hora de realizar el trabajo encomendado pues este se demora más y con el uso de la herramienta que se propone en este trabajo se garantiza la integración de los procesos y la solución real a estos problemas que se vienen afrontando, así como el seguimiento y control de todas las actividades realizadas.

2.2. Objeto de automatización

El mecanismo propuesto para la integración de todos los procesos de gestión de la calidad ha sido concebido con el fin de mejorar los procesos que se realizan en el Grupo de Calidad de la Facultad 7. Dicho mecanismo se encargará de automatizar los procesos antes mencionados y facilitará la comunicación y organización entre los integrantes del Grupo de Calidad, pues todos trabajarán sobre la aplicación para así ganar en calidad y eficiencia.

La aplicación posibilitará que todo el personal tenga acceso a ella y en dependencia del rol que le haya sido asignado podrá interactuar con ella. Al mismo tiempo, los datos estarán disponibles para los Jefes de Prueba, el Asesor de Calidad y el Administrador de Calidad, lo que posibilitará tener un control de las actividades que se han estado realizando.

De una u otra forma todos se verán beneficiados con el uso de esta herramienta porque agiliza y optimiza el trabajo, pues con acceder a la aplicación ya estarán disponibles, para cada usuario, sus tareas, donde tendrá la posibilidad una vez terminada, de subir la misma.

2.3. Descripción de términos comunes

Se denomina ***cronograma*** a un esquema básico donde se distribuye y organiza un conjunto de actividades diseñadas para un tiempo determinado.

Se denomina ***revisión*** al proceso que se utiliza para purificar las actividades que suceden en torno a un problema determinado.

Se denomina ***prueba*** a una serie de pasos que se realizan para determinar la existencia de errores en el software que se está probando.

Se denomina ***reporte*** a los informes generados en la aplicación de las tareas realizadas, así como del control de la asistencia.

2.4. Modelo de dominio

El modelo de dominio muestra las clases conceptuales significativas en el dominio del problema, captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará la herramienta. Es considerado un subconjunto del llamado modelo de objetos del negocio. (16)

Durante el desarrollo de la herramienta no se pudo contactar procesos bien definidos en el entorno del negocio. Se hizo difícil determinar los elementos más importantes del sistema y sus interconexiones, así como el establecimiento de las reglas de funcionamiento. Sin embargo, se pueden identificar personas, eventos, transacciones y objetos involucrados en ese entorno que no está bien delimitado, por lo que se hizo necesario un modelado del domino perteneciente a la solución.

Esto ayuda a los usuarios; a utilizar un vocabulario común para poder entender el contexto en el cual se ubica el sistema. Y para capturar correctamente los requisitos y poder construir un sistema correcto se necesita tener un firme conocimiento del funcionamiento del objeto de estudio. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema. Modelo de Dominio se encuentra en el Anexo 1.

2.5. Especificación de los requisitos de software

El término requerimiento no se utiliza de forma consistente en la industria del software. En algunos casos, un requerimiento se visualiza como una declaración abstracta de alto nivel de un servicio que debe proveer el sistema o como una restricción de este. Por otro lado, es una definición matemática detallada y formal de una función del sistema.

Se plantea una descripción detallada de lo que el sistema va a desarrollar. Se incluyen una serie de casos de uso los cuales son los encargados de describir las interacciones que tendrán los usuarios con el software. Dichos casos de uso también son llamados requisitos funcionales del sistema. Además de los casos de uso, se encuentran los requerimientos no funcionales que imponen restricciones en el diseño o la implementación. (17)

2.5.1. Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, especifican acciones que debe ser capaz de realizar. Ellos deben de ser comprensibles por los clientes, usuarios y desarrolladores, deben tener una sola interpretación y estar definidos en forma medible y verificable. (18)

Es por ello que una vez conocidos los conceptos que rodean al objeto de estudio, se analiza ¿Qué debe hacer el sistema para que se cumplan los objetivos planteados al inicio del trabajo de diploma?, enumerándose a través de requerimientos funcionales las instrucciones que este debe ser capaz de efectuar. Dentro de ellos se incluyen las acciones que podrán ser ejecutadas por el usuario, las acciones ocultas que debe realizar el sistema, y las condiciones extremas a determinar.

1. ***Autenticar usuario.***
2. ***Administrar proyectos.***
   - 2.1. Asignar proyecto.
   - 2.2. Modificar proyecto.
   - 2.3. Eliminar proyecto.
3. ***Administrar roles.***
   - 3.1. Adicionar rol.
   - 3.2. Modificar rol.
   - 3.3. Eliminar rol.
   - 3.4. Modificar permisos del rol.
4. ***Administrar usuarios.***
   - 4.1. Adicionar usuario.
   - 4.2. Modificar usuario.
   - 4.3. Eliminar usuario.
5. ***Administrar tareas.***
   - 5.1. Asignar tareas.
   - 5.2. Modificar tareas.
   - 5.3. Eliminar tareas.
6. ***Listar tarea.***
   - 6.1. Modificar tarea.
   - 6.2. Eliminar tarea.
7. ***Generar reportes de tareas.***
   - 7.1. Buscar tareas.
8. ***Gestionar Fase***
   - 8.1. Adicionar Fase.
   - 8.2. Modificar Fase.
   - 8.3. Eliminar Fase.

2.5.2. Requerimientos no funcionales

Los requerimientos no funcionales poseen características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en el tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares. Con el propósito de responder las necesidades del Grupo de Calidad de la Facultad 7, se definió un conjunto de propiedades o cualidades que debe cumplir la plataforma, las cuales se describen a continuación:

Interfaz externa:

- Herramienta sencilla.
  - ✓ Amigable al usuario y fácil de usar.
  - ✓ Funcionalidades explícitas.

Usabilidad:

- Sistema orientado a personas con conocimientos de computación.
- La interfaz es de un flujo sencillo.
- Será útil para manejar la información de los procesos que se desarrollen.

Soporte:

- Consta con la documentación necesaria para el aprendizaje sobre el uso de la aplicación.

Portabilidad:

- Es compatible con varios navegadores de internet como Mozilla Firefox, Google Chrome y Opera***.***

Software (Servidor):

- Sistema operativo Linux o Windows.
- Servidor de aplicaciones web Apache.
- Servidor de Base de Datos PostgreSQL.

Seguridad:

- La herramienta protege la información que se maneja otorgando los permisos necesarios.
- En caso de algún error la información debe ser guardada de forma que las pérdidas sean mínimas.

Hardware:

- Rendimiento mínimo del hardware.
  - Procesador 600 MHz o superior.
  - 128 MB de memoria RAM.

2.6. Modelo del sistema

A continuación se realizará el modelado del sistema a través de un diagrama de casos de uso donde se representan gráficamente los procesos que se llevarán a cabo y su interacción con los actores.

2.6.1. Definición de los actores del sistema

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a 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 no son parte de él, pueden intercambiar información con él, ser un recipiente pasivo de información y representar el rol que juega una o varias personas, un equipo o un sistema automatizado.

Actores	***Descripción***
Administrador de Calidad	Representa al equipo de desarrollo designado para dar respuestas y soluciones a las acciones realizadas por el equipo de pruebas. Es el responsable, por parte de los desarrolladores, de llevar el control del equipo que interviene en la revisión correspondiente a su producto de software. Controla las no conformidades generadas en el proceso.
Jefe de Pruebas	Controla las pruebas realizadas y los errores que fueron encontrados.
Usuario	Realiza las acciones comunes definidas para todos los que interactúan con la aplicación.
Profesor	Se encarga de orientar y dirigir las tareas orientadas. Además de realizar las acciones comunes definidas para el Jefe de Pruebas y Jefe de Revisiones.

Tabla 2.1. Descripción de los actores del sistema.

2.6.2. Diagrama de casos de uso del sistema

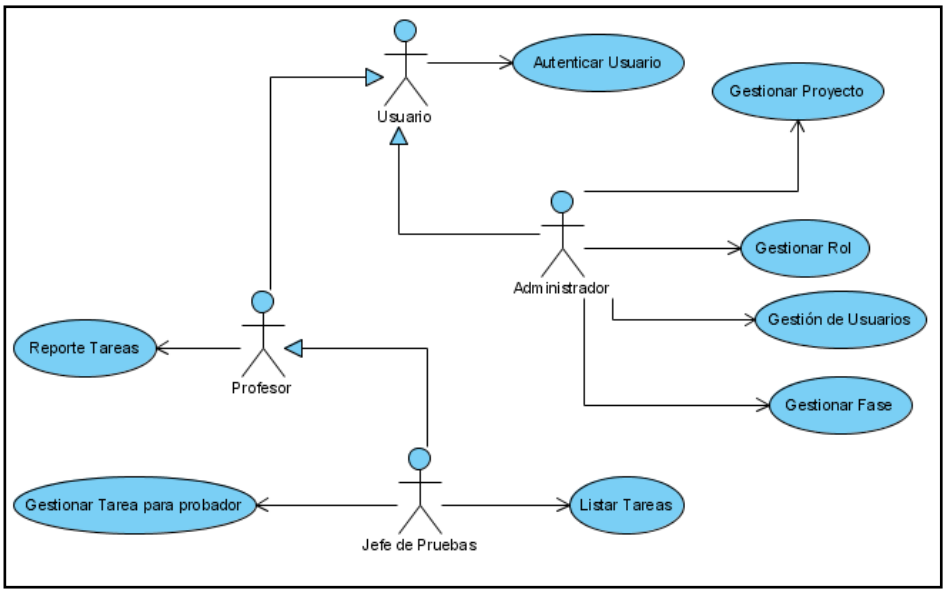

Figura 2. Diagrama de casos de uso del sistema.

Las descripciones de los casos de uso están disponibles en el Anexo 2.

En este capítulo se realizó un análisis crítico de los procesos actuales. Este análisis permitió precisar el estado actual de los procesos que se desarrollan en el Grupo de Calidad de la Facultad 7, para así asegurar la calidad de los mismos y apoyar la integración de las diferentes herramientas para lograr mayor calidad y eficiencia en el trabajo desarrollado. De la información obtenida, se definieron las necesidades de funcionamiento de la aplicación a implementar.

ANÁLISIS Y DISEÑO

En el presente capítulo se abordarán los aspectos relacionados con este flujo de trabajo; incluyendo, los diagramas de clases tanto del análisis como del diseño, diagramas de interacción y la descripción de las clases de diseño.

Esta disciplina define la arquitectura del sistema y tiene como objetivo trasladar requisitos en especificaciones de implementación; al decir análisis se refiere a obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado, el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos.

3.1. Flujos de trabajo Análisis y Diseño

El objetivo principal de esta disciplina es transformar los requerimientos a una especificación que describa cómo implementar el sistema. El análisis fundamentalmente consiste en obtener una visión que se preocupa de ver qué hace el software a desarrollar, por tal motivo este se interesa en los requerimientos funcionales. Por otro lado, el diseño es un refinamiento que toma en cuenta los requerimientos no funcionales, por lo cual se centra en cómo el sistema cumple sus objetivos.

Los objetivos específicos del análisis y diseño son:

1. Transformar los requerimientos al diseño del futuro sistema.
2. Desarrollar una arquitectura para el sistema.
3. Adaptar el diseño para que sea consistente con el entorno de implementación.

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los casos de uso con las interacciones de las clases de análisis. Durante la fase de elaboración se va

refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes.

3.2. Modelo de análisis

El modelo de análisis es usado para representar la estructura global del sistema. Este no es un diagrama final que describe todos los posibles conceptos y sus relaciones, es un primer intento por definir los conceptos claves que describen el sistema. Este artefacto es opcional, pero también tiene a su vez la propiedad de ser temporal.

Para representar los diagramas del Modelo de Análisis se pueden emplear diferentes diagramas de UML tales como:

- Diagramas de Clase.
- Diagramas de Secuencia.
- Diagramas de Colaboración.

Durante el análisis, se analizan los requisitos que fueron descritos en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar todo el sistema, incluyendo su arquitectura.

3.2.1. Diagrama de clases

Los diagramas de clases del análisis representan la relación entre las clases, las cuales se centran en los requisitos funcionales, tienen atributos y entre ellas se establecen relaciones de asociación, agregación/composición, generalización/especialización y tipos asociativos. RUP propone clasificar a las clases en: interfaz, controladora y entidad.

- Las clases Interfaz modelan la interacción entre el sistema y sus actores.
- Las clases Controladoras coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.
- Las clases Entidad modelan información que posee larga vida y que es a menudo persistente.

Se presentarán los diagramas de clases del análisis de los casos de uso identificados como principales: Gestionar roles, Gestionar tareas y Gestionar proyectos, el resto de los diagramas se encuentran en el Anexo 3.

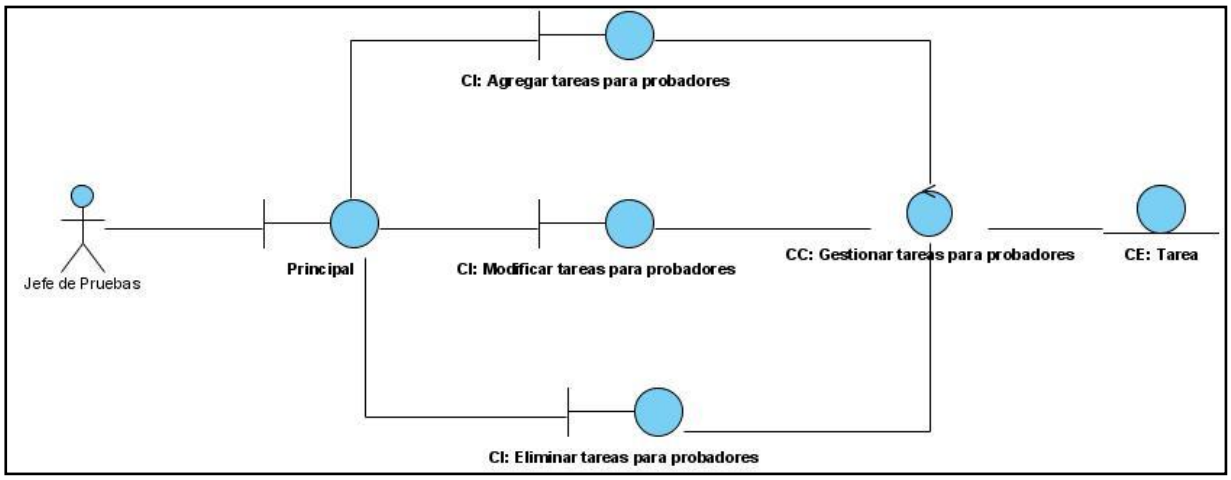

Figura 4. Diagrama de clases del análisis CU Gestionar tareas para probadores.

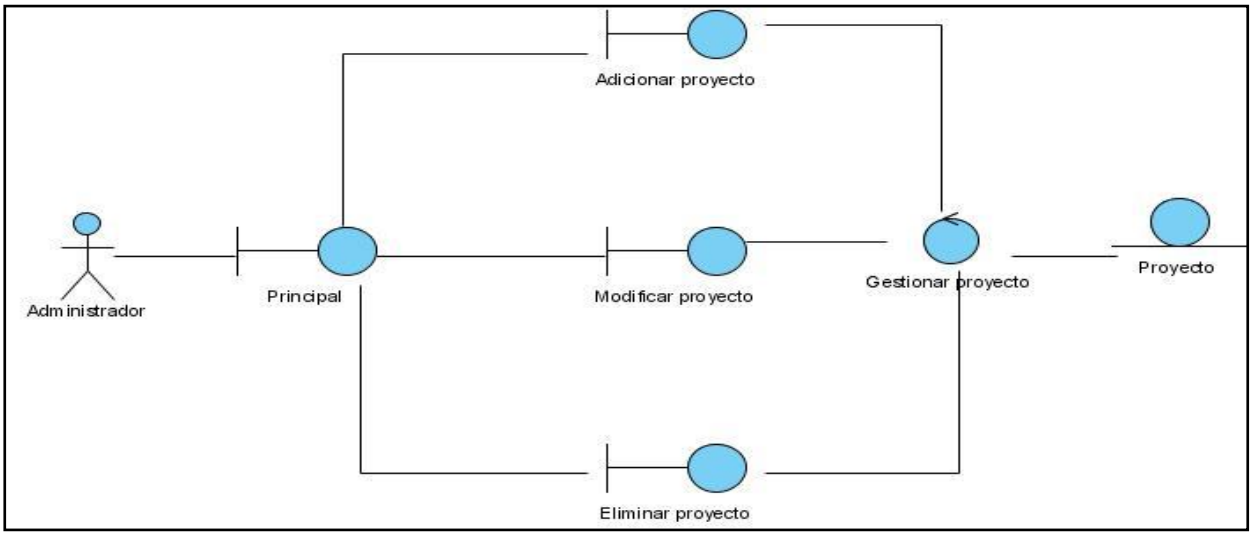

Figura 5. Diagrama de clases del análisis CU Gestionar proyectos.

3.3. Modelo de diseño

El diseño dice cómo lograr los requerimientos relevados en el análisis, usando la metodología definida anteriormente, se puede describir, cómo el sistema llevará a cabo cada una de las funciones definidas,

este es uno de las etapas más largas en el desarrollo de un sistema. La especificación del diseño aborda diferentes aspectos del modelo de diseño y se completa a medida que el diseñador refina su propia representación del software.

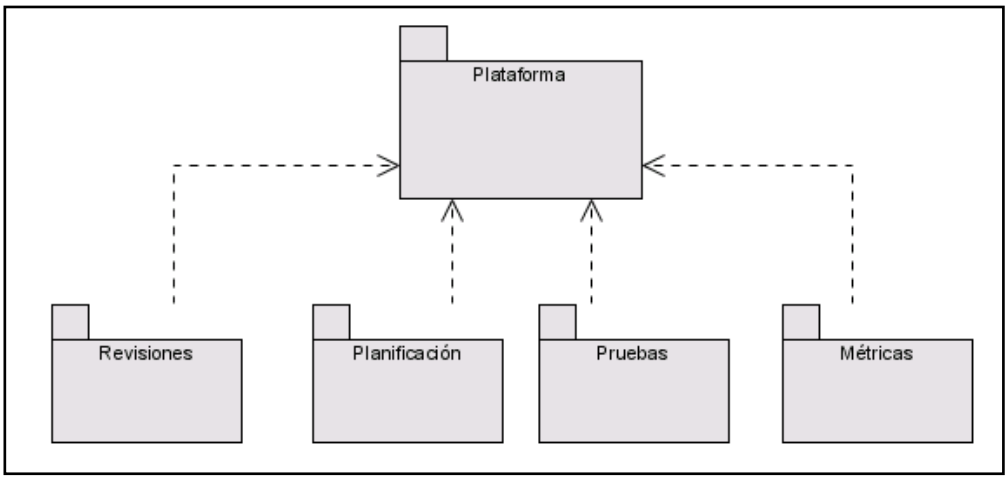

Figura 6. Modelo de diseño.

3.3.1. Definición de los elementos de diseño

El Modelo Vista Controlador es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Sin lugar a dudas, ha tenido un gran auge en el desarrollo de aplicaciones web.

Modelo: Es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación.

Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

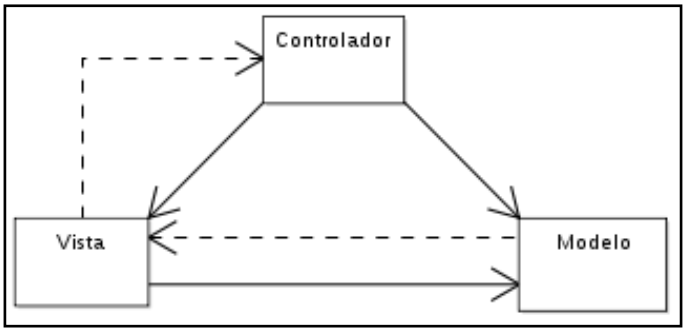

Figura 7. Modelo Vista Controlador.

3.3.2. Fundamentación del uso de patrones

Para llevar a cabo un buen diseño se han definido una serie de patrones. Los patrones de diseño de software constituyen un conjunto de principios generales y expresiones que ayudan a desarrollar software. Los patrones GRASP[3] describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Dentro de este grupo se identifican 5 patrones fundamentales experto, creador, alta cohesión, bajo acoplamiento y el controlador. En los diagramas de clases elaborados se aplican dichos patrones, se utilizan a fin de distribuir responsabilidades en las mismas, y establecer sus relaciones, tratando de que no estén muy sobrecargadas de funcionalidades ni exista mucha dependencia entre ellas.

El patrón ***Experto*** es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Su problema radica en el principio general para asignar responsabilidades a los objetos, teniendo como solución asignar una responsabilidad al experto en información.

Algunos beneficios que proporciona son:

- El encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas.
- Se distribuye el comportamiento entre las clases que contienen la información requerida.
- Son más fáciles de entender y mantener.

[3] Patrones Generales de Software para Asignación de Responsabilidades (General Responsability Assignment Software Patterns)

El patrón ***Creador*** ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.

El patrón ***Controlador*** es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Este patrón sugiere que la lógica de negocio debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.

El patrón de ***Alta cohesión*** dice que la información que almacena una clase debe ser coherente y está, en la mayor medida de lo posible, relacionada con la clase.

El patrón de ***Bajo acoplamiento*** es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en las demás, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

3.3.3. Diagrama de clases

En el diagrama de clases de diseño se muestran los atributos y métodos de cada clase y se representa de una forma sencilla la colaboración y las responsabilidades de las distintas clases que forman el sistema. El diagrama de clases de diseño representa la parte estática del sistema. Contiene las clases del diseño y sus relaciones.

A continuación presentarán los diagramas de clases del diseño de los casos de uso: Gestionar tareas para un probador y Gestionar usuarios, el resto de los diagramas se encuentran en el Anexo 6.

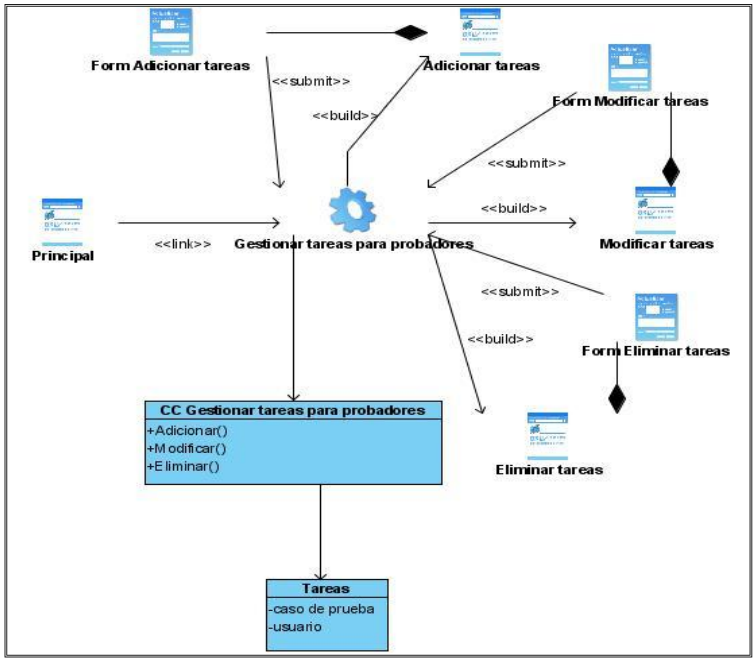

Figura 9. Diagrama de clases del diseño CU Gestionar tareas.

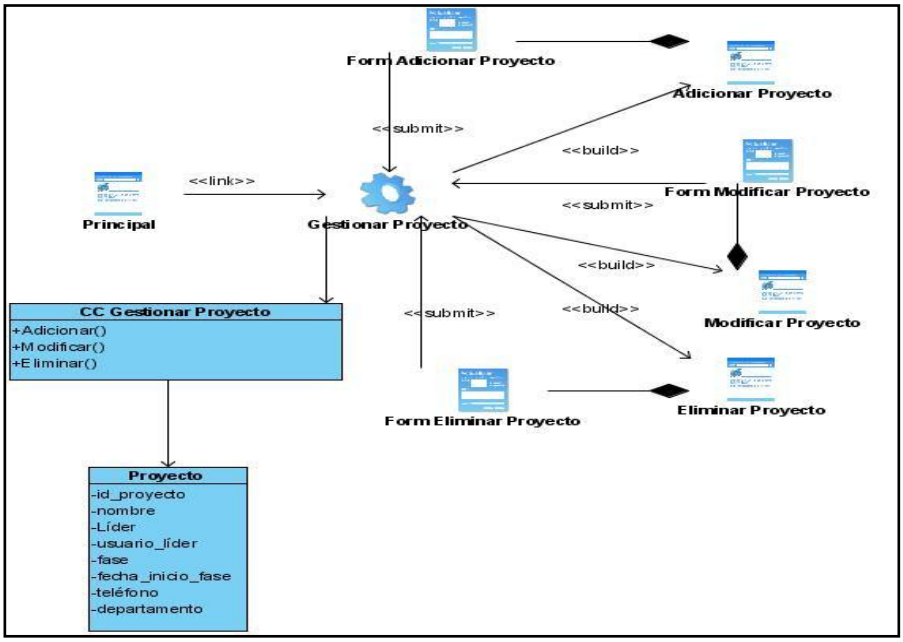

Figura 10. Diagrama de clases del diseño CU Gestionar proyectos.

En el capítulo se detallaron las fases de Análisis y Diseño de la aplicación, donde se definen las clases y modelos que las conforman. Se especificaron los principios y patrones de diseño del sistema y se realizó una descripción de las clases del diseño de la aplicación. La arquitectura definida está orientada al logro de una base sólida para la construcción del sistema. Fueron obtenidos los artefactos correspondientes a los flujos de trabajo desarrollados acorde a la metodología utilizada. Además se detallan los tipos de datos o componentes que las conforman.

IMPLEMENTACIÓN

En el presente capítulo se realiza la implementación del sistema tal y como lo define RUP para esta disciplina dentro de la fase de Construcción. Además se muestra el diseño de la base de datos, el diagrama de despliegue, el diagrama de componentes y se define cómo estará organizado el sistema en términos de códigos y subsistemas.

4.1. Modelo de datos

El modelo de datos es un lenguaje orientado a describir las bases de datos. Un modelo de datos permite describir la estructura de datos de la base y su relación, las restricciones de integridad y las operaciones de manipulación de datos. Para ello se realiza el diagrama de clases persistentes y el modelo entidad relación.

4.1.1. Diagrama de clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Lo contrario son las clases temporales que son manejadas y almacenadas por el sistema en tiempo de ejecución, por lo que dejan de existir cuando termina el programa. El diagrama de clases persistentes se encuentra en el Anexo 4.

4.1.2. Diagrama Entidad-Relación

El Diagrama Entidad-Relación lo conforman las tablas que contiene la base de datos del sistema con sus atributos y relaciones entre ellas. En el Anexo 5 y 7 se encuentra el modelo de datos y las descripciones de cada una de las tablas que conforman la base de datos.

4.2. Implementación.

Uno de los flujos de trabajo que define RUP como metodología de desarrollo, es el de implementación. En este se comienza con el resultado del diseño y se implementa el sistema en término de componentes, es decir, ficheros de código fuente, *scripts*, ficheros de código binario, ejecutables y similares. Su objetivo principal es desarrollar la arquitectura del sistema como un todo. Los propósitos de la implementación son:

- Definir la organización del código.
- Planificar las integraciones del sistema necesarias en cada iteración.
- Implementar las clases y subsistemas encontrados durante el Diseño.

4.2.1. Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware y muestra cómo los elementos y artefactos de software se trazan.

Se utiliza en el diseño y en la implementación (como el diagrama de componentes). Es más limitado que el diagrama de componentes; en el sentido que representa la estructura del sistema sólo en tiempo de ejecución, pero no en tiempo de desarrollo o compilación, por lo que resulta más amplio pues puede contener más clases de elementos.

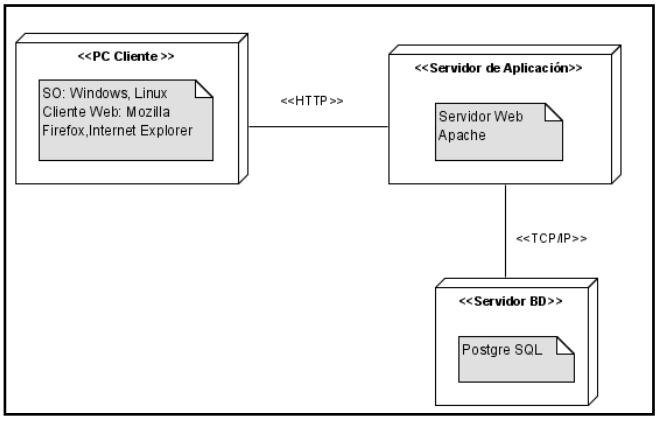

Figura 13. Diagrama de despliegue.

4.2.2. Diagrama de componente

El diagrama de componentes ilustra los componentes del software que serán usados para construir el sistema. Estos pueden ser construidos para el modelo de clase y escritos para satisfacer los requisitos del nuevo sistema, o puede ser dada para otros proyectos o vendedores de tercera persona. El diagrama de componente muestra la relación entre los componentes del software, sus dependencias, comunicaciones, localización y otras condiciones. Los diagramas de componentes son usados para estructurar los componentes en los sistemas del software. Ellos examinan y controlan las dependencias entre componentes o interfaces de los componentes. Un componente representa una parte modular, desplegable y reutilizable de un sistema. (19)

Para llevar a cabo la implementación se trabajó con el patrón de arquitectura Modelo-Vista-Controlador.

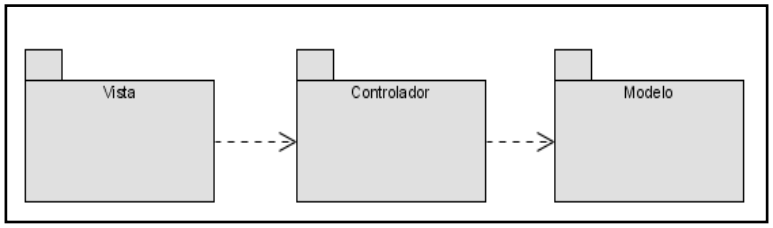

Figura 14. Diagrama de paquetes.

Cada uno de los paquetes anteriores encapsula uno o más componentes que se interrelacionan.

Detalles de la Vista

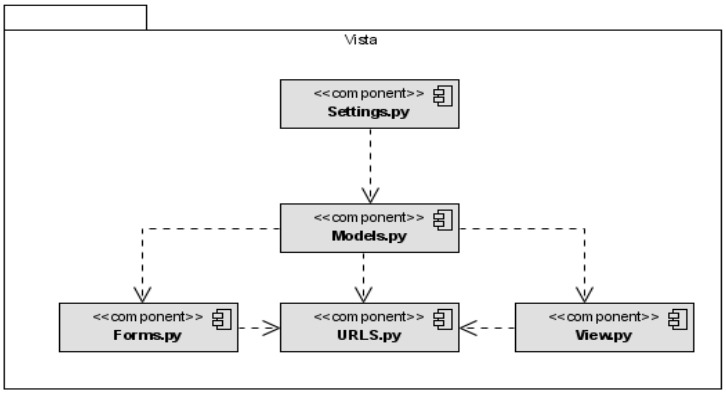

Figura 15. Detalles de la Vista.

Detalles del Modelo

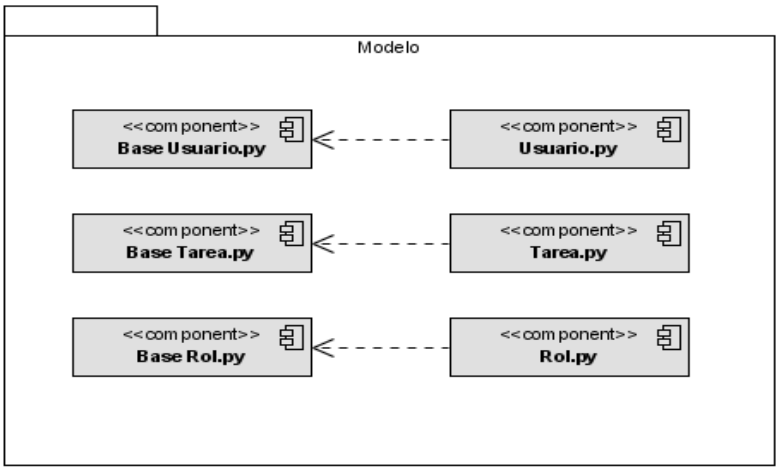

Figura 16. Detalles del Modelo.

Detalles del Controlador

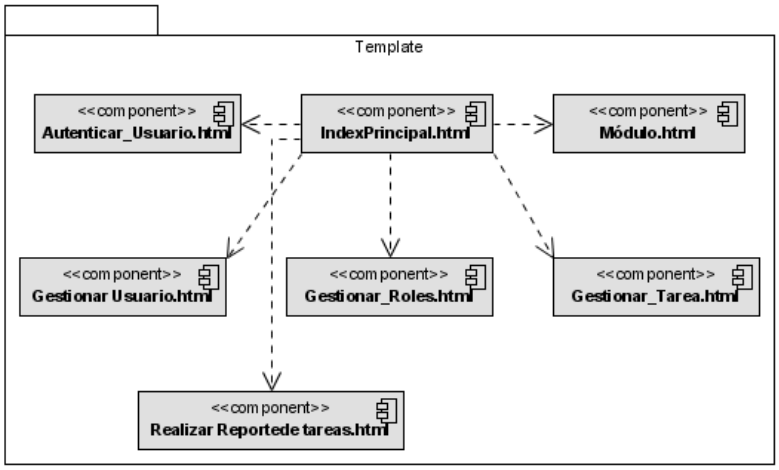

Figura 17. Detalles del Controlador.

4.2.3. Seguridad

Debido a que el sistema implementado está basado en la web, la seguridad juega un papel fundamental para el correcto funcionamiento. La seguridad es un elemento esencial para lograr la integridad, confidencialidad y autenticidad de la información almacenada y manejada por el sistema. Para ello el sistema implementa mecanismos que permitan asegurarla.

La gestión de permisos de acceso sobre los recursos del sistema se realiza de forma dinámica; mediante las variables de sesión se controla en todo momento la información vital sobre el usuario que se encuentra haciendo uso de las funcionalidades del sistema (las variables de sesión, a diferencia de otras tecnologías, se manejan del lado del servidor, evitando la modificación con fines dañinos al sistema). Para la interacción con el sistema se han definido una serie de roles que se enumeran según el orden jerárquico de la siguiente forma: Administrador, Jefe de Pruebas, Jefe de Revisiones, Probador y Revisor. Los permisos para cada grupo de usuarios se obtienen al realizarse una autenticación exitosa.

Los mensajes informativos que se muestran al usuario evitarán en todo momento mostrar información sensible del sistema.

En este capítulo se realizó la implementación del sistema propuesto, donde se generaron los modelos de datos y de implementación. Con la elaboración de estos artefactos, se obtiene una visión más detallada que permite facilitar y agilizar los procesos que se desarrollan en el Grupo de Calidad.

CONCLUSIONES

Al concluir la investigación se ha cumplido el objetivo y las tareas propuestas. Las herramientas existentes utilizadas para la gestión de procesos no responden a las necesidades que se presentan en el Grupo de Calidad de La Facultad 7. Estas no son compatibles entre sí y al no estar integradas gestionan por separado la información que utilizan. Es por ello que fue necesario el desarrollo de una herramienta para la gestión de los procesos que se desarrollan en el Grupo de Calidad.

Se realizó un análisis de las tendencias actuales de las herramientas y tecnologías, y se seleccionaron las más adecuadas para la implementación del sistema teniendo en cuenta las particularidades del mismo.

Se definió una arquitectura de diseño e implementación que dará solución a los principales problemas que existen en el Grupo de Calidad.

Como resultado final, se obtuvo una herramienta que posibilita resolver las dificultades presentes en el Grupo de Calidad y así agilizar la realización de las tareas.

RECOMENDACIONES

Para futuras investigaciones se recomienda:

- Implantar el sistema en el laboratorio de Calidad de la Facultad 7.
- Incorporar el módulo de Pruebas a la aplicación además de otros subsistemas de gestión administrativa según las necesidades, a fin de elevar las prestaciones del sistema propuesto.
- Actualizar y ofrecer soporte a la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. [En línea] Septiembre-Diciembre de 1995. [Citado el: 15 de noviembre de 2009.] http://www.bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm.

2. ***Raúl Martinez, Viviana Lacunza.*** Proceso de Mejora de Software. [En línea] FIUBA – Administración y Control de Proyectos II, abril de 2006. [Citado el: 20 de noviembre de 2010.] http://materias.fi.uba.ar/7546/material/Mejora%20del%20Proceso%20de%20Software.pdf.

3. Pressman, R.S. [En línea] 1982. [Citado el: 25 de noviembre de 2009.] http://www.monografias.com/trabajos16/calidad-sw-pymes/calidad-sw-pymes.shtml .

4. Lovelle, Juan Manuel Cueva. Universidad Nacional de la Pampa. [En línea] 21 de octubre de 1999. [Citado el: 28 de noviembre de 2010.] http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

5. Gestión de la Calidad del Software. [En línea] [Citado el: 30 de noviembre de 2010.] http://softqm.blogspot.com/2006/11/gestin-de-la-calidad-del-software.html.

6. Fernández, Sara Zapico. INFORME V. [En línea] 2 de febrero de 2009. [Citado el: 2 de diciembre de 2009.] http://www.jtech.ua.es/tutoriales/apuntes/sesion-junit-apuntes.htm.

6. Fuentes, Yurien Ricardo. Herramienta para viabilizar el proceso de pruebas de caja blanca. [En línea] [Citado el: 8 de diciembre de 2009.] Ciudad de la Habana.

7. Bugzilla. [En línea] [Citado el: 5 de diciembre de 2009.] http://www.bugzilla.org.

8. José Rolando Peréz, Lether Delgado. Herramienta para la revisión y seguimiento de errores de la documentación generada. [En línea] 8 de diciembre de 2009. [Citado el: 8 de diciembre de 2009.] Ciudad de la Habana.

9. Jiménez. Definición Planificación. [En línea] 1982. [Citado el: 12 de diciembre de 2009.] http://www.apuntesgestion.com/2008/08/13/definicion-planificaci%C3%B3n/.

10. GanttPV 1.0. *Softonic.* [En línea] 1997. [Citado el: 13 de diciembre de 2009.] http://ganttpv.softonic.com/mac.

11. Autores, Colectivo de. Una propuesta para la Gestión de Revisiones. [En línea] [Citado el: 15 de diciembre de 2009.]

12. María del Carmen Padrón, Yuneicy Pérez Peña. Propuesta de Programa de auditoría para evaluar la Gestión de la Calidad . [En línea] [Citado el: 17 de diciembre de 2009.] http://www.eumed.net/cursecon/ecolat/cu/2009/cppp.htm.

13. Acerca de PostgreSQL. [En línea] [Citado el: 28 de enero de 2010.] http://www.postgresql.cl/acerca_de.

14. Kaplan-Moss, Adrian Holovaty y Jacob. *Django Book.* 2008.

15. *Rational Enterprise Edition, Ayuda extendida.* 2003.

16. Especificaciones De Requerimientos. [En línea] [Citado el: 20 de febrero de 2010.] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

17. Ingienería de Requerimientos. [En línea] [Citado el: 23 de febrero de 2010.] http://www.mitecnologico.com/Main/ProcesosDeLaIngenieriaDeRequerimientos.

18. Especificaciones De Requerimientos. [En línea] [Citado el: 20 de febrero de 2010.] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

19. Diagrama de Componentes. [En línea] [Citado el: 25 de febrero de 2010.] http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml.

BIBLIOGRAFÍA

1. [En línea] Septiembre-Diciembre de 1995. [Citado el: 15 de noviembre de 2009.] http://www.bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm.

2. Raúl Martinez, Viviana Lacunza. Proceso de Mejora de Software. [En línea] FIUBA – Administración y Control de Proyectos II, abril de 2006. [Citado el: 20 de noviembre de 2010.] http://materias.fi.uba.ar/7546/material/Mejora%20del%20Proceso%20de%20Software.pdf.

3. Cuerda, Xavier García. MOSAIC. [En línea] 29 de noviembre de 2004. [Citado el: 22 de noviembre de 2009.] http://mosaic.uoc.edu/2004/11/29/introduccion-a-los-sistemas-de-gestion-de-contenidos-cms-de-codigo-abierto/.

4. Pressman, R.S. [En línea] 1982. [Citado el: 25 de noviembre de 2009.] http://www.monografias.com/trabajos16/calidad-sw-pymes/calidad-sw-pymes.shtml .

5. Lovelle, Juan Manuel Cueva. Universidad Nacional de la Pampa. [En línea] 21 de octubre de 1999. [Citado el: 28 de noviembre de 2010.] http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

6. Gestión de la Calidad del Software. [En línea] [Citado el: 30 de noviembre de 2010.] http://softqm.blogspot.com/2006/11/gestin-de-la-calidad-del-software.html.

7. Pruebas de software. [En línea] [Citado el: 1 de diciembre de 2009.] http://es.wikipedia.org/wiki/Pruebas_de_software.

8. Fernández, Sara Zapico. INFORME V. [En línea] 2 de febrero de 2009. [Citado el: 2 de diciembre de 2009.] http://www.jtech.ua.es/tutoriales/apuntes/sesion-junit-apuntes.htm.

9. Jakarta Cactus. [En línea] 2 de Febrero de 2009. [Citado el: 2 de diciembre de 2009.] http://petra.euitio.uniovi.es/~i1773418/HdD/trabajos/INFORME%20V.pdf.

10. JTest de Parasoft. [En línea] 2007. [Citado el: 4 de diciembre de 2009.] http://www.als-es.com/home.php?location=herramientas/entorno-desarrollo/jtest.

11. JMeter. [En línea] 2004. [Citado el: 5 de diciembre de 2009.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jmeter.

12. Mora, Roberto Canales. Herramientas Selenium . [En línea] [Citado el: 6 de diciembre de 2009.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=selenium_rc.

13. Bugzilla. [En línea] [Citado el: 5 de diciembre de 2009.] http://www.bugzilla.org.

14. Fuentes, Yurien Ricardo. Herramienta para viabilizar el proceso de pruebas de caja blanca. [En línea] [Citado el: 8 de diciembre de 2009.] Ciudad de la Habana..

15. José Rolando Peréz, Lether Delgado. Herramienta para la revisión y seguimiento de errores de la documentación generada. [En línea] 8 de diciembre de 2009. [Citado el: 8 de diciembre de 2009.] Ciudad de la Habana.

16. Jiménez. Definición Planificación. [En línea] 1982. [Citado el: 12 de diciembre de 2009.] http://www.apuntesgestion.com/2008/08/13/definicion-planificaci%C3%B3n/.

17. Grupo de gestión de la tecnología. [En línea] 1997. [Citado el: 12 de diciembre de 2009.]

18. GanttPV 1.0. *Softonic.* [En línea] 1997. [Citado el: 13 de diciembre de 2009.] http://ganttpv.softonic.com/mac.

19. Autores, Colectivo de. Una propuesta para la Gestión de Revisiones. [En línea] [Citado el: 15 de diciembre de 2009.]

20. María del Carmen Padrón, Yuneicy Pérez Peña. Propuesta de programa de auditoría para evaluar la Gestión de la Calidad . [En línea] [Citado el: 17 de diciembre de 2009.] http://www.eumed.net/cursecon/ecolat/cu/2009/cppp.htm.

21. Software para la gestión de proyectos en entorno web. [En línea] Earcon, S.L. - www.earcon.com . [Citado el: 10 de enero de 2010.] http://www.kmkey.com.

22. Portal de Python. [En línea] [Citado el: 12 de enero de 2010.] http://www.python.org.

23. Qué es Python? [En línea] [Citado el: 14 de enero de 2010.] http://www.desarrolloweb.com/articulos/1325.php.

24. El Lenguaje de Programación. [En línea] [Citado el: 15 de enero de 2010.] http://www.ecualug.org/files/Flisol%20-%20Python.pdf.

25. Entorno de desarrollo. [En línea] [Citado el: 17 de enero de 2010.] http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado.

26. ¿Qué es un Sistema Gestor de Bases de Datos o SGBD? . [En línea] CAVSI, 2004. [Citado el: 20 de enero de 2010.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

27. Acerca de PostgreSQL. [En línea] [Citado el: 28 de enero de 2010.] http://www.postgresql.cl/acerca_de.

28. Metodología de desarrollo de software. [En línea] [Citado el: 3 de febrero de 2010.] http://www.scribd.com/doc/12983474/Metodologia-de-Desarrollo-de-Software..

29. Universidad Politécnica de Valencia (RUP). [En línea] mayo de 2007. [Citado el: 10 de febrero de 2010.]https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc.

30. Lenguaje de Modeldo. [En línea] Vico.org, 2002. [Citado el: 10 de febrero de 2010.] http://www.vico.org/FormMentorOutsourcingUML.pdf.

31. Visual_Paradigm. [En línea] [Citado el: 15 de febrero de 2010.] http://ucipedia.uci.cu/index.php/Visual_Paradigm.

32. Kaplan-Moss, Adrian Holovaty y Jacob. *Django Book.* 2008.

33. *Rational Enterprise Edition, Ayuda extendida.* 2003.

34. Especificaciones De Requerimientos. [En línea] [Citado el: 20 de febrero de 2010.] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

35. Ingienería de Requerimientos. [En línea] [Citado el: 23 de febrero de 2010.] http://www.mitecnologico.com/Main/ProcesosDeLaIngenieriaDeRequerimientos.

36. Modelo de Implementación. [En línea] [Citado el: 20 de febrero de 2010.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

37. Diagrama de Componentes. [En línea] [Citado el: 25 de febrero de 2010.] http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml.

38. Gestión de la Calidad de Software. [En línea] [Citado el: 30 de noviembre de 2010.] http://softqm.blogspot.com/2006/11/gestin-de-la-calidad-del-software.html.

39. Universidad Teconológica de Pereira. [En línea] [Citado el: 12 de noviembre de 2009.] http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/111233326-331.pdf.

40. Cuerda. Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto . [En línea] 29 de noviembre de 2004. [Citado el: 15 de noviemmbre de 2009.] http://mosaic.uoc.edu/2004/11/29/introduccion-a-los-sistemas-de-gestion-de-contenidos-cms-de-codigo-abierto/.

41. Calidad del Software. [En línea] [Citado el: 15 de noviembre de 2009.] http://www.monografias.com/trabajos59/calidad-software/calidad-software2.shtml .

42. *Guía de los Fundamentos de la Dirección de Proyectos.* 2004.

43. Definicion Calidad De Software. [En línea] [Citado el: 25 de noviembre de 2009.] http://www.mitecnologico.com/Main/DefinicionCalidadDeSoftware.

44. Sistemas e Gestión de la Calidad. [En línea] [Citado el: 25 de noviembre de 2009.] http://www.monografias.com/trabajos52/gestion-calidad/gestion-calidad3.shtml.

45. AENOR. Las nuevas normas de Gestión de la calidad UNE-EN ISO 9001:2000 Y UNE-EN ISO 9004:2000. [En línea] 2002. [Citado el: 26 de noviembre de 2009.] http://www.femenp.net/PDF/5.pdf.

46. Herramientas de pruebas. [En línea] [Citado el: 2 de enero de 2009.] http://www.als-es.com/home.php?location=herramientas/entorno-pruebas.

47. Kaplan-Moss, Adrian Holovaty y Jacob. *The Definitive Guide to Django.* 2009.

48. Alchin, Marty. *Pro Django.* 2009.

49. Django. [En línea] 2007. [Citado el: 7 de enero de 2009.] http://www.linperial.com/communities/forums/developers/?q=taxonomy/term/11.

50. *Rational Unified Process (RUP).* Universidad Politecnica de Valencia : s.n.

51. MeRinde 2010 . [En línea] [Citado el: 10 de enero de 2010.] http://www.merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=137&Itemid=192.

52. Ingienería de Sistemas. [En línea] marzo de 24. [Citado el: 15 de enero de 2010.] http://athenea.ort.edu.uy/publicaciones/ingsoft/ortsf/areas/IntroduccionCalidad.pdf.

53. Ejemplo de desarrollo software utilizando la metodología RUP . [En línea] [Citado el: 15 de enero de 2010.] http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/Analisis-Disenyo.html#mod_datos.

54. Ejemplo de desarrollo software utilizando la metodología RUP. [En línea] [Citado el: 15 de enero de 2010.] http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/Analisis-Disenyo.html#mod_datos.

55. Comunidad de SOFTWARE factory. [En línea] [Citado el: 20 de enero de 2010.] http://www.newsoftwarefactory.com/implementacion.htm.

56. Comunidad SPARK SYSTEMS. [En línea] [Citado el: 5 de febrero de 2010.] http://www.sparxsystems.com.ar/resources/tutorial/use_case_model.html..

57. Comunidad SPARK SYSTEMS. [En línea] [Citado el: 12 de febrero de 2010.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

GLOSARIO DE TÉRMINOS

Herramienta Case: Aplicación informática destinada a aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en tiempo y dinero, se utiliza para el modelado del sistema.

Framework: Es una estructura de soporte definida en la cual otro proyecto puede ser organizado y trabajado. Un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

IDE: Entorno Integrado de Desarrollo, es un programa compuesto por un conjunto de herramientas para un programador. Puede ser exclusivo de un lenguaje o puede ser utilizado en más de uno.

MVC: Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica del control en tres componentes distintos. Es ampliamente utilizado en aplicaciones Web.

Calidad de Software: se considera calidad de software a la concordancia entre los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente.

Gestión de la Calidad de Software: es un conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades. El propósito de la gestión de la calidad del software es entender las expectativas del cliente en términos de calidad, y poner en práctica un plan proactivo para satisfacer esas expectativas.

Procesos de la calidad de software: de gestión de calidad de software se definen los principios de la gestión de calidad presentados en la norma ISO 9004:2000, los cuales se enfocan a una estructura que permite que las organizaciones mejoren su rendimiento. Estos principios se originan en las mejores prácticas y la experiencia de numerosas compañías e instituciones internacionales.

ANEXOS

Anexo 1

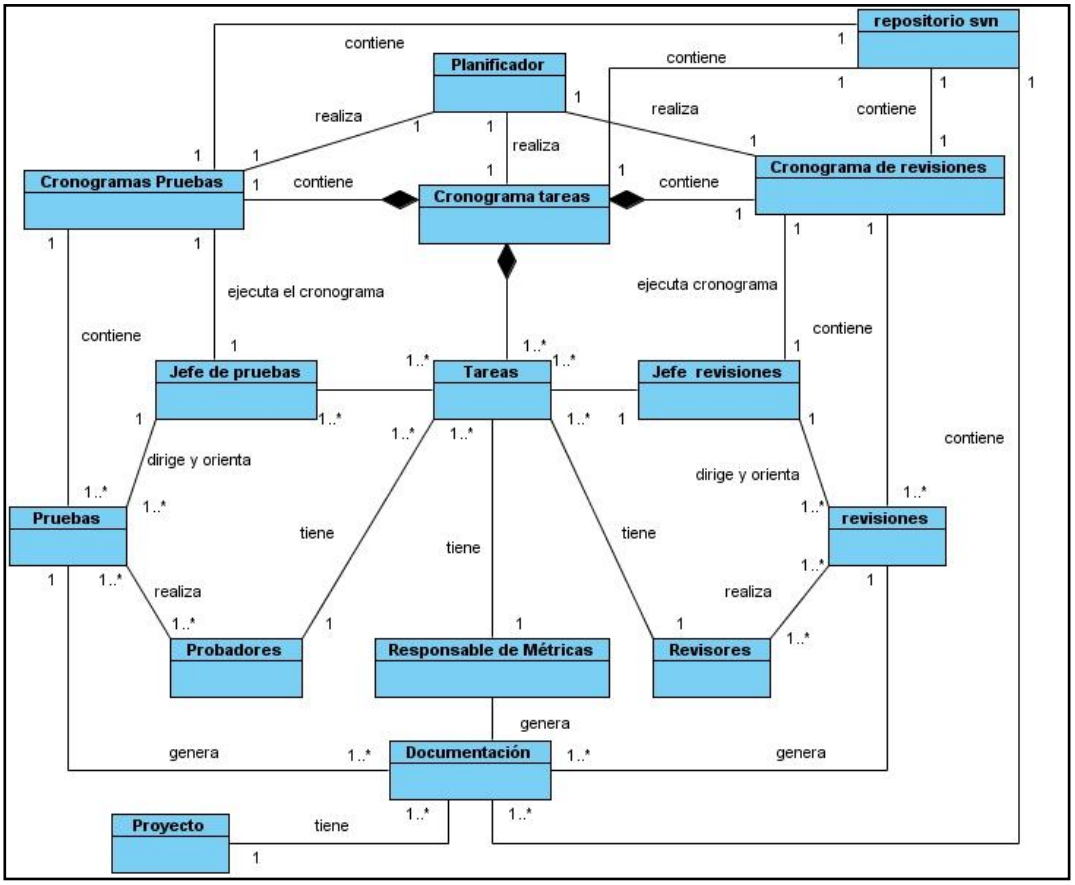

Figura 1. Modelo de dominio.

Anexo 2. Descripciones extendidas de los casos de uso.

Caso de Uso	
Caso de Uso	Autenticar usuario
Propósito	Permitir o denegar el acceso a la aplicación.
Actores: Usuario	
Resumen: EL Caso de Uso se inicializa cuando el usuario decide autenticarse, para ello ingresa su identificador y la contraseña, el sistema determina si puede acceder o no a la aplicación. El Caso de Uso finaliza cuando el usuario se autentica en el sistema.	
Referencias	RF 1
Acción del actor	***Respuesta del sistema***

	1. Muestra la vista para la realización de la autenticación: • Usuario • Contraseña
2. El usuario introduce el usuario y la contraseña correspondiente. Oprime el botón “Aceptar”.	3. Verifica que todos los datos hayan sido proporcionados. 4. Verifica la existencia del usuario y la correspondencia con la contraseña suministrada. 5. Permite el acceso a la aplicación acorde a los permisos que posee el usuario autenticado.
Flujo Alterno 1	
Acción del actor	***Respuesta del sistema***
	3. Se muestra un mensaje informando que su contraseña es incorrecta.
Flujo Alterno 2	
Acción del actor	***Respuesta del sistema***
	4. Muestra un mensaje de error informando de un error en la autenticación.
	5. Permite volver a realizar la operación.

Caso de uso		
CU 2	Gestionar Usuarios	
Propósito	Permitir realizar la gestión de los usuarios.	
Actores: Administrador de calidad.		
Resumen: EL Caso de Uso se inicializa cuando el Administrador decide realizar acciones como agregar, modificar o eliminar un usuario, además puede ver los detalles de un usuario previamente seleccionado. EL Caso de Uso finaliza cuando se culmina la acción seleccionada previamente (agregar, eliminar, modificar).		
Referencias	***RF 2***	
Acción del actor		***Respuesta del sistema***
1. El Administrador la opción “Gestionar Usuarios”.		2. Muestra un listado de los usuarios existentes.
3. Escoge una de las opciones: a) Agregar un nuevo usuario ***(Ver Sección “Adicionar Usuario”)***. b) Modificar un usuario ***(Ver Sección “Modificar Usuario”)***.		

c) Eliminar un usuario ***(Ver Sección "Eliminar Usuario")***.	
Sección "Adicionar Usuario"	
Acción del actor	***Respuesta del sistema***
	1. Muestra los campos para la inserción de los datos necesarios: • Nombre • Apellidos • Rol • Identificador • Contraseña • Dirección de correo
2. Entra los datos especificados y escoge la opción "Aceptar".	2.1. Verifica que los datos necesarios están completos y son correctos. 2.2. El usuario nuevo es agregado a los usuarios del sistema. 2.3. Se muestra un mensaje informando del éxito de la operación. 2.4. Los campos de entrada de datos se muestran vacíos.
Flujo Alterno 1	
	2.1. Muestra un mensaje informando del error en los datos proporcionados.
3. Cancela el mensaje	4. Permite volver a realizar la operación
Flujo Alterno 2	
2. Entra los datos especificados y escoge la opción "Cancelar".	2.1. El usuario nuevo no es agregado. 2.2. Los campos de entrada de datos se muestran limpios.
Sección "Modificar Datos del Usuario"	
Acción del actor	***Respuesta del sistema***
1. Selecciona un usuario del listado	2. Muestra los datos del usuario seleccionado. • Nombre • Apellidos • Rol • Identificador • Contraseña • Dirección de correo
3. Modifica los datos y escoge la opción	3.1. Verifica que los datos necesarios están

"Aceptar".	completos y son correctos. 3.2. Los datos del usuario son modificados. 3.3. Se muestra un mensaje informando del éxito de la operación. 3.4. Los campos de entrada de datos se muestran vacíos.
Flujo Alterno 1	
	3.1. Muestra un mensaje informando del error en los datos proporcionados.
4. Acepta el mensaje	5. Permite volver a realizar la operación
Flujo Alterno 2	
3. Modifica los datos y escoge la opción "Cancelar".	3.1. Los datos del usuario no son modificados. 3.2. Los campos de entrada de datos se muestran limpios.
Sección "Eliminar Usuario"	
Acción del actor	***Respuesta del sistema***
1. Selecciona un usuario del listado 2. Escoge la opción "Eliminar".	
	3. Muestra un mensaje para confirmar la acción.
4. En el mensaje escoge "Aceptar".	5. El usuario es eliminado del sistema.
Flujo Alterno 1	
3. En el mensaje escoge "Cancelar".	4. El usuario no es eliminado del sistema.

Caso de Uso	
Caso de Uso	Gestionar Roles
Propósito	Permitir realizar la gestión de los roles para cada usuario.
Actores: Administrador	
Resumen: EL Caso de Uso se inicializa cuando el Administrador decide realizar acciones como agregar o eliminar un usuario y finaliza cuando se culmina la acción seleccionada previamente (agregar, eliminar).	
Referencias	***RF4, RF4.1, RF4.2, RF4.3***
Acción del actor	***Respuesta del sistema***
1. El administrador selecciona opción ***Modificar*** usuario.	2. El sistema muestra un vínculo a la página de administración de Django, específicamente a la sección de los Grupos.

3 Escoge una de las opciones: ***a)*** Agregar un nuevo rol ***(Ver Sección "Adicionar Rol").*** b) Eliminar un rol ***(Ver Sección "Eliminar Rol")***.	4 Selecciona el usuario y lo agregas al grupo al que se desee.
Sección "Adicionar Rol"	
Acción del actor	***Respuesta del sistema***
	1. Se adiciona el nombre del rol y se seleccionan los permisos.
	1.1. El sistema verifica que los datos necesarios estén correctos y completos. 1.2. El rol nuevo es agregado a los roles del sistema. 1.3. Se muestra un mensaje informando del éxito de la operación.
Flujo Alterno 1	
	2.1. Muestra un mensaje informando del error en los datos proporcionados.
3. Acepta el mensaje	4. Permite volver a realizar la operación
Flujo Alterno 2	
2. Entra los datos especificados y escoge la opción "Cancelar".	2.1. El usuario nuevo no es agregado. Los campos de entrada de datos se muestran limpios.
Sección "Eliminar Rol"	
Acción del actor	***Respuesta del sistema***
1. Selecciona un rol del listado 2. Escoge la opción "Eliminar".	
	3. Muestra un mensaje para confirmar la acción.
4. En el mensaje escoge "Aceptar".	5. El rol es eliminado del sistema.
Flujo Alterno 1	
4. En el mensaje escoge "Cancelar".	5. El usuario no es eliminado del sistema.

<table>
<tr><td colspan="3">Caso de uso</td></tr>
<tr><td>Caso de Uso</td><td colspan="2">CU Gestionar Proyecto</td></tr>
<tr><td>Propósito</td><td colspan="2">Permite realizar la gestión de los proyectos.</td></tr>
<tr><td colspan="3">Actores: Administrador de Calidad.</td></tr>
<tr><td colspan="3">Resumen: El caso de uso se inicia cuando el actor desea adicionar, modificar o eliminar algún proyecto, el sistema permite realizar una de las opciones deseadas, el caso de uso termina.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF3.1,RF3.2,RF3.3</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1. El caso de uso se inicia cuando el Administrador desea adicionar, modificar o eliminar algún proyecto.</td><td colspan="2">2. Brinda la posibilidad de modificar o eliminar el proyecto.</td></tr>
<tr><td>3 Selecciona la opción Aceptar o Cancelar.</td><td colspan="2">4 Si selecciona la opción Aceptar ,modifica o elimina en dependencia de lo que se desee hacer
Si se selecciona la opción cancelar se cancela toda la operación.</td></tr>
<tr><td colspan="3">Sección “Adicionar Proyecto”</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1 El caso de uso se inicia cuando el actor accede a la opción Adicionar Proyecto.</td><td colspan="2">2 Permite Introducir los datos
• Nombre
• Fase
• Fecha fin fase
• Nombre líder
• usuario líder
• Teléfono
• Departamento</td></tr>
<tr><td>3 Introduce los datos.</td><td colspan="2"></td></tr>
<tr><td>4 Selecciona la opción Aceptar o Cancelar.</td><td colspan="2">5 Si selecciona Aceptar.
• Existen campos vacios, se muestra un mensaje
“Este campo es obligatorio”
Si opción Cancelar
• Se cancela toda la operación.</td></tr>
</table>

	7 El caso de uso termina.
Flujo Alterno 1	
	1. Muestra un mensaje informando que los campos son obligatorios.
Flujo Alterno 2	
	3. Si selecciono la opción Cancelar se regresa ala vista anterior.
	4. Se termina el caso de uso.
Sección "Modificar Proyecto.	
Acción del actor:	***Respuesta del sistema***
1 El caso de uso se inicia cuando el actor accede a la opción ***Modificar Proyecto.***	2. Permite modificar los datos • Nombre • Fase • Fecha fin fase • Nombre líder • usuario líder • Teléfono • Departamento
3. Modifica los datos deseados.	
4. Selecciona Opción Modificar O Cancelar.	***1.*** Si selecciona la opción ***Modificar***: • . Si existen campos vacios: lanza un mensaje :"Existen campos vacios." Si selecciona la opción ***Cancelar***: Se cancela la operación.
Flujo Alterno 1	
	1 Muestra un mensaje informando que los campos son obligatorios. "Estos campos son obligatorios."
Flujo Alterno 2	
	2 Si se cancela la operación se regresa a la vista anterior.
	3 Se termina el caso de uso.

Sección "Eliminar Proyecto"	
Acción del actor	***Respuesta del sistema***
1 El caso de uso se inicia cuando el actor accede a la opción ***Cerrar Proyecto***	2 Muestra los datos predeterminados • Nombre • Fase • Fecha fin fase • Nombre líder • usuario líder • Teléfono • Departamento
3 Se selecciona la opción Aceptar o Cancelar.	4 Si selecciona la opción ***Cerrar***: • Se cierra el proyecto Si selecciona ***Cancelar*** "Cancelar Operación
Flujo Alterno 1	
	1. Muestra un mensaje informando que los campos son obligatorios. "Estos campos son obligatorios
Flujo Alterno 2	
	Se selecciona la opción Cancelar. Se cancela la operación.

Caso de Uso	
Caso de Uso	Gestionar Tareas para probadores
Propósito	Permitir realizar la gestión de las tareas para probadores por parte de sus respectivos líderes.
Actores: Jefe de Pruebas	
Resumen: El Caso de Uso se inicializa cuando el Jefe de Pruebas desea realizar acciones como agregar, modificar o eliminar una tarea.	

<table>
<tr><td>Referencias</td><td colspan="2">RF5 , RF5.1, RF5.2, RF5.3</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El caso de uso se inicia cuando el Jefe de Prueba desea asignar, modificar o eliminar alguna tarea.</td><td>2. Brinda la posibilidad de asignar, modificar o eliminar la tarea.</td></tr>
<tr><td colspan="3">Sección “Asignar Tarea”</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1 Escoge la opción “Asignar tarea”.</td><td>1. Muestra los campos correspondientes.
• Fecha_inicio
• Fecha_fin
• Descripción
• Tipo_tarea
• General
• Responsable
• Por_Ciento_Cumplimiento
• Dependencia
• Seguidor
• Documento
• Estado
• Evaluación</td></tr>
<tr><td colspan="2">1. Escoge la opción “Aceptar o Cancelar”.</td><td>2. Si selecciona Aceptar
• Verifica si existen campos vacios de los obligatorios, se muestra un mensaje

“Este campo es obligatorio”

Si opción Cancelar

• Se cancela toda la operación.

3. Termina el caso de uso.</td></tr>
<tr><td colspan="3">Flujo Alterno 1</td></tr>
<tr><td colspan="2"></td><td>1. Muestra un mensaje informando que los campos son obligatorios.</td></tr>
<tr><td colspan="3">Flujo Alterno 2</td></tr>
<tr><td colspan="2"></td><td>2. Si selecciono la opción Cancelar se regresa ala vista anterior.</td></tr>
<tr><td colspan="2"></td><td>3. Se termina el caso de uso.</td></tr>
<tr><td colspan="3">Flujo Alterno 3</td></tr>
<tr><td colspan="3">Sección “Modificar Tarea”</td></tr>
</table>

Acción del actor	***Respuesta del sistema***
	1. Muestra todas las tareas que han sido asignadas a casos de pruebas que no han sido terminadas.
2. Selecciona una tarea. 3. Escoge la opción "Modificar Tarea Asignada"	4. Muestra todos los casos de prueba que se encuentran excepto aquellos que han sido terminados. 5. Muestra todos los usuarios que poseen el rol de Probador.
4. Modifica el caso de prueba y/o el usuario que participará en ella, escoge la opción "Aceptar".	1.1. La nueva tarea es registrada en el sistema. 1.2. Se muestra un mensaje informando del éxito de la operación.
Flujo Alterno 1	
	1. Muestra un mensaje informando que no existen tareas para casos de prueba en estado de ejecución o comenzadas.
Flujo Alterno 2	
	4. Muestra un mensaje informando que no existen casos de prueba en estado de ejecución o comenzadas.
Flujo Alterno 3	
	5. Muestra un mensaje informando que no existen usuarios con el rol de Probador.
Flujo Alterno 4	
5. Selecciona el caso de prueba y el usuario que participará en ella, escoge la opción "Cancelar".	1.1. La nueva tarea no es agregada.
Sección "Eliminar Tarea"	
Acción del actor	***Respuesta del sistema***
	1. Muestra todas las tareas que han sido asignadas a casos de prueba que no han sido terminadas
2. Selecciona una tarea. 3. Escoge la opción "Eliminar Tarea Asignada".	4. Se muestra un mensaje para confirmar la acción.
5. En el mensaje escoge "Aceptar".	5.1. La tarea es eliminada del sistema.
Flujo Alterno 1	

	1. Muestra un mensaje informando que no existen tareas para casos de prueba en estado de ejecución o comenzadas.
Flujo Alterno 2	
5. En el mensaje escoge “Cancelar”.	5.1. La tarea no es eliminada del sistema.

Caso de uso	
Caso de Uso.	Listar Tarea
Actores: Jefe Prueba, Jefe Revisiones.	
Resumen: El caso de uso se inicia cuando el actor desea listar las tareas. Estas se pueden modificar o eliminar según el sistema permite realizar una de las opciones deseadas, el caso de uso termina.	
Referencias	***RF6,RF6.1,,RF6.2***

Acción del actor	***Respuesta del sistema***
1. El caso de uso se inicia cuando el Jefe de Pruebas o el Jefe de Revisiones desea listar las tareas.	2. Brinda la posibilidad de modificar o eliminar las tareas.
3. Selecciona la opción ***Aceptar*** o ***Cancelar***.	4. Si selecciona la opción ***Aceptar.*** • Modifica o elimina en dependencia de lo que se desee hacer. Si se selecciona la opción ***Cancelar.*** • Se cancela toda la operación.
Sección “Modificar Tarea.	
Acción del actor:	***Respuesta del sistema***
1. El caso de uso se inicia cuando el actor accede a la opción ***Modificar Tarea.***	2. Permite modificar los datos.
3. Modifica los datos deseados.	
4. Selecciona Opción Modificar O Cancelar.	5. Si selecciona la opción ***Modificar***: • . Si existen campos vacios que son obligatorios, lanza un mensaje :”Existen campos vacios.” Si selecciona la opción ***Cancelar***: Se cancela la operación.

Flujo Alterno 1	
	1 Muestra un mensaje informando que los campos son obligatorios. “Estos campos son obligatorios.”
Flujo Alterno 2	
	4 Si se cancela la operación se regresa a la vista anterior.
	5 Se termina el caso de uso.
Sección “Eliminar Tarea”	
Acción del actor	***Respuesta del sistema***
1. El caso de uso se inicia cuando el actor accede a la opción ***Eliminar Tarea.***	2. Si se oprime la opción Eliminar. Aparece un mensaje: • “Está seguro que desea eliminar.”
3. Se selecciona la opción ***Aceptar*** o ***Cancelar***.	4. Si selecciona la opción ***Aceptar.***: • Se elimina la tarea. Si selecciona ***Cancelar.*** • “Cancelar Operación”.
Flujo Alterno 1	
	2. Muestra un mensaje informando si desea eliminar. “Está seguro que desea eliminar”.
Flujo Alterno 2	
	Se selecciona la opción ***Aceptar.*** • Se elimina la tarea. Si se selecciona la opción ***Cancelar.*** • Se cancela la operación.

Caso de uso	
Caso de Uso	Realizar Reporte
Propósito	Permitir realizar un reporte de las tareas que fueron desarrolladas.
Actores: Profesor	
Resumen: El Caso de Uso se inicializa cuando uno de los actores necesita visualizar un reporte con los resultados de las tareas que fueros realizadas.	
Referencias	***RF7***

Acción del actor	***Respuesta del sistema***
1. Se escoge la opción Realizar Reporte.	2. Se muestran las opciones: • Responsable • Estado Se muestra el botón Buscar.
3. Escoge el responsable y el estado de la tarea y presiona el botón Buscar.	4. Muestra los elementos
	5. Se genera el Reporte.
Flujo Alterno 1	
Acción del actor	***Respuesta del sistema***
	1. Muestra un mensaje informando: "No existe la tarea solicitada".
Flujo Alterno 2	
	1. Muestra un mensaje informando que "No existen datos para generar el reporte solicitado".

Caso de uso	
CU 8	Gestionar Fase
Propósito	Permitir realizar la gestión de las fases en las que se encuentra un proyecto.
Actores: Administrador de calidad.	
Resumen: EL Caso de Uso se inicializa cuando el Administrador decide adicionar un proyecto y tiene que especificar en que fase se encuentra dicho proyecto.	
Referencias	***RF 8***

Acción del actor	***Respuesta del sistema***
4. El Administrador la opción "Gestionar Proyectos".	5. Muestra un listado de datos y entre ellos la Fase.
6. Escoge la opción :	

d) Adicionar una nueva Fase ***(Ver Sección "Adicionar Fase")***. e) Modificar un usuario ***(Ver Sección "Modificar Fase")***. f) Eliminar un usuario ***(Ver Sección "Eliminar Fase")***.	
Sección "Adicionar Fase"	
Acción del actor	***Respuesta del sistema***
	3. Muestra los campos para la inserción de la Fase • Nombre
4. Entra los datos especificados y escoge la opción "Aceptar".	4.1. Se muestra un mensaje informando del éxito de la operación. 4.2. Los campos de entrada de datos se muestran vacíos.
Flujo Alterno 1	
Sección "Modificar Fase"	
Acción del actor	***Respuesta del sistema***
	4. Muestra los datos del usuario seleccionado. • Nombre
5. Modifica los datos y escoge la opción "Aceptar".	5.1. Verifica que el nombre este completo y correctos. 5.2. Los datos son modificados. 5.3. Se muestra un mensaje informando del éxito de la operación. 5.4. Los campos de entrada de datos se muestran vacíos.
Sección "Eliminar Fase"	
Acción del actor	***Respuesta del sistema***
6. Selecciona del listado la Fase 7. Escoge la opción "Eliminar".	
	8. Muestra un mensaje para confirmar la acción.
9. En el mensaje escoge "Aceptar".	10. El usuario es eliminado del sistema.
Flujo Alterno 1	

5. En el mensaje escoge “Cancelar”.	6. El usuario no es eliminado del sistema.

Anexo 3. Diagramas de clases del análisis.

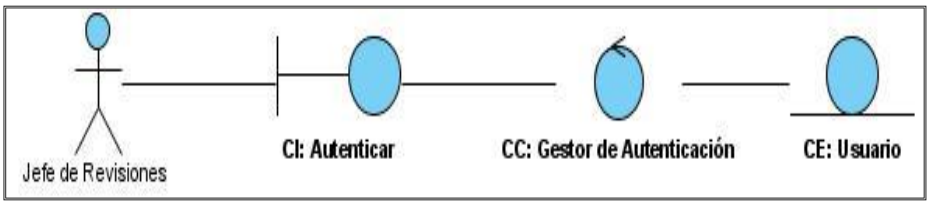

Figura 1. DCA Autenticar Usuario.

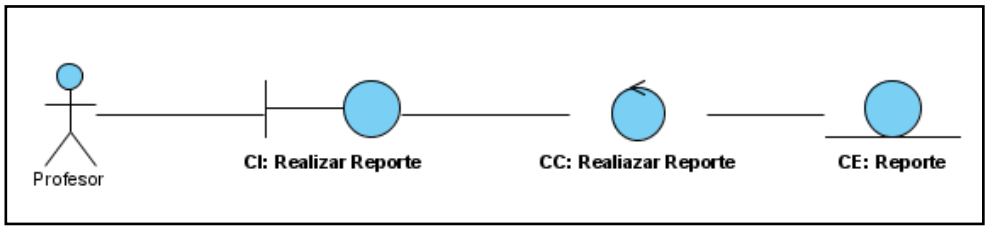

Figura 2. DCA Realizar Reporte.

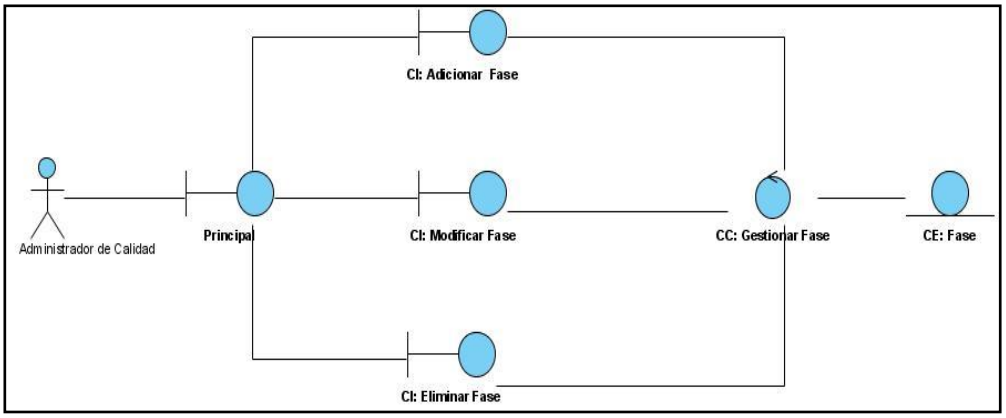

Figura 3. DCA Gestionar Fase.

Anexo 4. Diagrama de clases persistentes

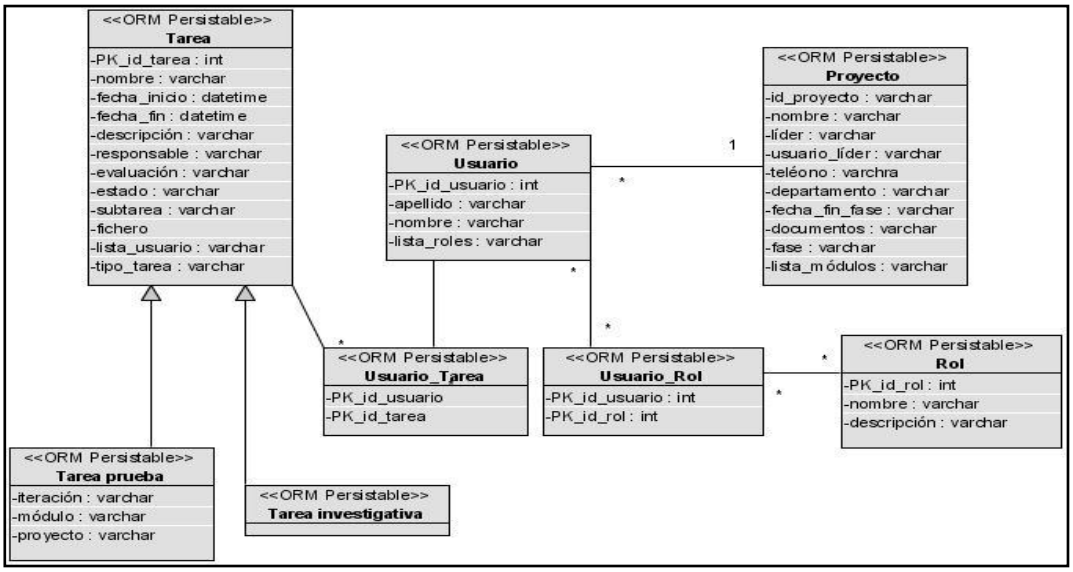

Figura 4. Diagrama de clases persistentes.

Anexo 5. Modelo de datos

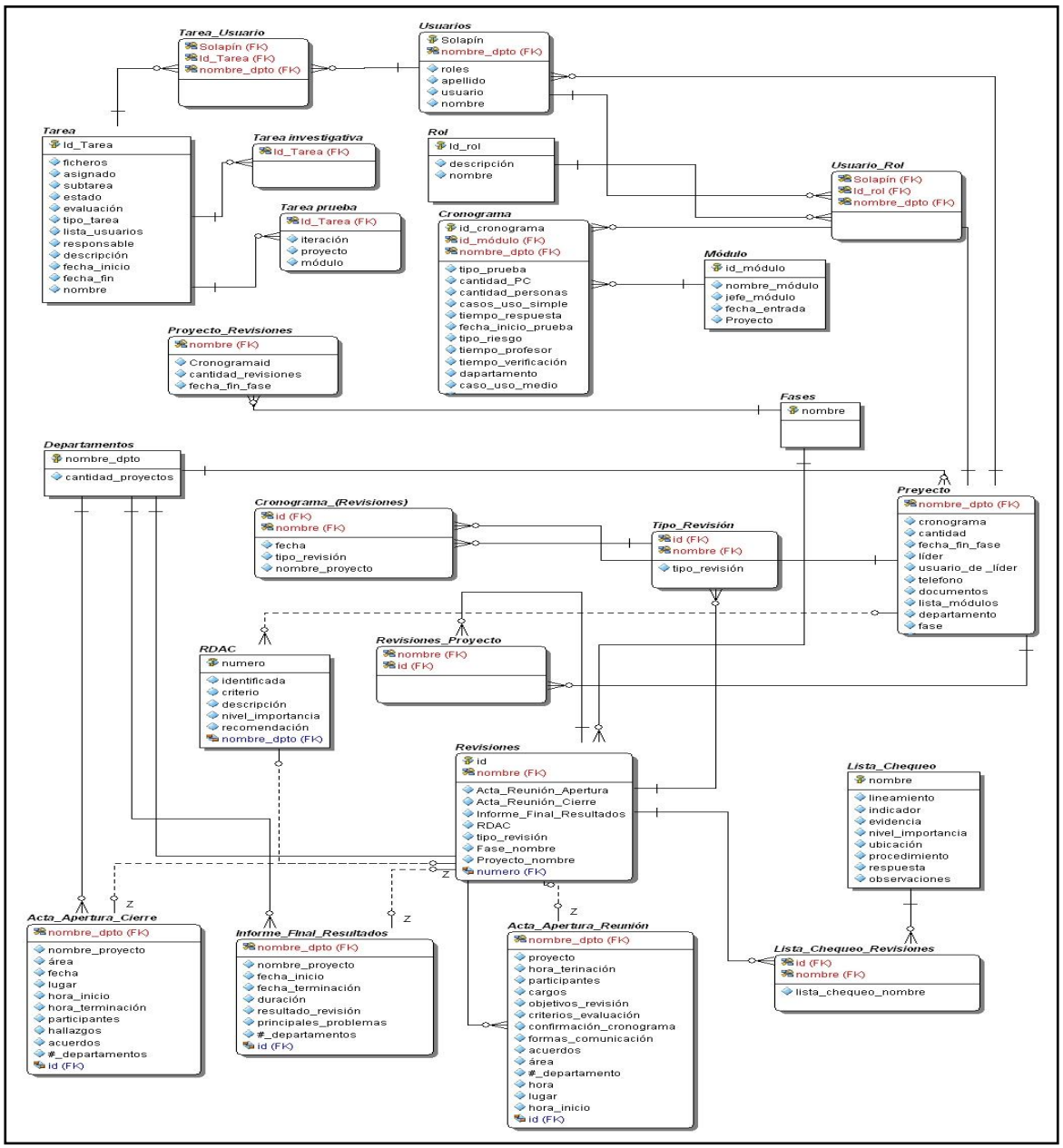

Figura 5. Diagrama Entidad-Relación.

Anexo 6. Descripción de las clases del diseño.

Nombre: CC Gestor de Autenticación	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	Autenticar Usuario()
Descripción:	Permite verificar si el usuario es correcto.

Nombre: CC Gestionar Usuario	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	Adicionar Usuario()
Descripción:	Permite adicionar un nuevo usuario.
Nombre:	Modificar Usuario()
Descripción:	Modificar el usuario.
Nombre:	Eliminar Usuario()
Descripción:	Elimina el usuario deseado.

Nombre: CC Gestionar Rol	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	Asignar Rol()
Descripción:	Permite adicionar un nuevo rol
Nombre:	Modificar Rol()
Descripción:	Modificar el rol deseado.
Nombre:	Eliminar Rol()
Descripción:	Eliminar el rol deseado

Nombre: CC Gestionar Tarea para probadores
Tipo de clase: controladora

Para cada responsabilidad:	
Nombre:	Asignar Tarea()
Descripción:	Permite adicionar una nueva tarea
Nombre:	Modificar Tarea()
Descripción:	Modificar la tarea deseada.
Nombre:	Eliminar Tarea()
Descripción:	Elimina la tarea deseada.

Nombre: CC Gestionar Proyecto	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	Asignar Proyecto()
Descripción:	Permite adicionar un nuevo proyecto.
Nombre:	Modificar Proyecto()
Descripción:	Modificar el proyecto deseado.
Nombre:	Eliminar Proyecto()
Descripción:	Elimina Proyecto.

Nombre: CC Listar Tarea	
Tipo de clase: controladora	
Atributo	***Tipo***
Para cada responsabilidad:	
Nombre:	Modificar Tarea()
Descripción:	Modificar la tarea deseada.
Nombre:	Eliminar Tarea()
Descripción:	Elimina la Tarea.

Nombre: CC Realizar Reporte	
Tipo de clase: controladora	
Atributo	***Tipo***
Para cada responsabilidad:	
Nombre:	Realizar Reporte()
Descripción:	Permite realizar el reporte de la tarea asignada.

Nombre: CC Gestionar Fase	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	Adicionar Fase()
Descripción:	Permite adicionar una nueva fase al proyecto.
Nombre:	Modificar Fase()
Descripción:	Modificar la Fase.
Nombre:	Eliminar Fase()
Descripción:	Elimina la Fase.

Nombre: CI Autenticar	
Tipo de clase: interfaz	
Atributo	***Tipo***
Id_User	TextBox
Contraseña	TextBox
Nombre	TextBox
Apellido	TextBox
Rol	ComboBox
Aceptar	Booton
Cancelar	Booton
Para cada responsabilidad:	
Nombre:	Asignar Rol
Funcionalidad:	Brindará la funcionalidad de asignar el rol deseado.

Nombre: CI Asignar Rol	
Tipo de clase: interfaz	
Atributo	***Tipo***
Id_Rol	TextBox

Nombre	TextBox
Descripción	TextBox
Roles	ComboBox
Aceptar	Booton
Cancelar	Booton
Para cada responsabilidad:	
Nombre:	Asignar Rol
Funcionalidad:	Brindará la funcionalidad de asignar el rol deseado.

Nombre: CI Modificar Rol	
Tipo de clase: interfaz	
Atributo	***Tipo***
Id_Rol	TextBox
Nombre	TextBox
Descripción	TextBox
Roles	ComboBox
Aceptar	Booton
Cancelar	Booton
Para cada responsabilidad:	
Nombre:	Modificar Rol
Funcionalidad:	Brindará la funcionalidad de modificar el rol deseado.

Nombre: CI Eliminar Rol	
Tipo de clase: interfaz	
Atributo	***Tipo***
Eliminar	Booton
Cancelar	Booton
Para cada responsabilidad:	
Nombre:	Eliminar_Rol
Funcionalidad:	Brindará la funcionalidad de eliminar el rol deseada.
Nombre:	Cancelar_Eliminar_Rol
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Asignar Tarea para probadores

Tipo de clase: interfaz	
Atributo	***Tipo***
Id_Tarea	TextBox
Nombre	TextBox
Fecha_inicio	Calendario
Fecha_fin	Calendario
Aceptar	Booton
Cancelar	Booton
Para cada responsabilidad:	
Nombre:	Asignar_Tarea
Funcionalidad:	Brindará la funcionalidad de Asignar la tarea.
Nombre:	Cancelar_ Asignar_Tarea
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Modificar Tarea para probadores	
Tipo de clase: interfaz	
Atributo	***Tipo***
Id_Tarea	TextBox
Nombre	TextBox
Fecha_inicio	Calendario
Fecha_fin	Calendario
Aceptar	Booton
Cancelar	Booton
Para cada responsabilidad:	
Nombre:	Modificar_Tarea
Funcionalidad:	Brindará la funcionalidad de modificar la tarea deseada
Nombre:	Cancelar_ Modificar_Tarea
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Eliminar Tarea para probadores	
Tipo de clase: interfaz	
Atributo	***Tipo***

Eliminar	Booton
Cancelar	Booton
Para cada responsabilidad:	
Nombre:	Eliminar_Tarea
Funcionalidad:	Brindará la funcionalidad de eliminar la tarea deseada
Nombre:	Cancelar_Eliminar_Tarea
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Adicionar Proyecto	
Tipo de clase: interfaz	
Atributo	***Tipo***
Id_Proyecto	TextBox
Nombre	TextBox
Fase	ComboBox
Fecha_Fin	ComboBox
Líder	TextBox
Usuario	TextBox
Teléfono	TextBox
Departamento	TextBox
Documentos	TextBox
Lista_Módulos	ComboBox
Para cada responsabilidad:	
Nombre:	Adicionar_Proyecto.
Funcionalidad:	Brindara la funcionalidad de Adicionar un proyecto.
Nombre:	Cancelar_ Adicionar Proyecto.
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Modificar Proyecto	
Tipo de clase: interfaz	
Atributo	***Tipo***
Id_Proyecto	TextBox
Nombre	TextBox
Fase	ComboBox
Fecha_Fin	ComboBox
Líder	TextBox

Usuario	TextBox
Teléfono	TextBox
Departamento	TextBox
Documentos	TextBox
Lista_Módulos	ComboBox
Para cada responsabilidad:	
Nombre:	Modificar_Proyecto.
Funcionalidad:	Brindará la funcionalidad de Modificar_Proyecto.
Nombre:	Cancelar_ Modificar Proyecto.
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Eliminar Proyecto	
Tipo de clase: interfaz	
Atributo	***Tipo***
Id_Proyecto	TextBox
Nombre	TextBox
Fase	ComboBox
Fecha_Fin	ComboBox
Líder	TextBox
Usuario	TextBox
Teléfono	TextBox
Departamento	TextBox
Documentos	TextBox
Lista_Módulos	ComboBox
Para cada responsabilidad:	
Nombre:	Eliminar Proyecto.
Funcionalidad:	Brindara la funcionalidad de Eliminar un proyecto.
Nombre:	Cancelar_ Eliminar Proyecto.
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Modificar Fase

Tipo de clase: interfaz	
Atributo	***Tipo***
Nombre	TextBox
Para cada responsabilidad:	
Nombre:	Modificar_Fase.
Funcionalidad:	Brindará la funcionalidad de Modificar_Fase.
Nombre:	Cancelar_ Modificar Fase.
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Eliminar Fase	
Tipo de clase: interfaz	
Atributo	***Tipo***
Nombre	TextBox
Para cada responsabilidad:	
Nombre:	Eliminar Fase.
Funcionalidad:	Brindará la funcionalidad de Eliminar una Fase.
Nombre:	Cancelar_ Eliminar Fase.
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Realizar Reporte	
Tipo de clase: interfaz	
Para cada responsabilidad:	
Nombre:	Realizar Reporte.
Funcionalidad:	Brindará la funcionalidad de Realizar un reporte de las tareas.
Nombre:	Cancelar_ Realizar Reporte.
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CI Adicionar Fase	
Tipo de clase: interfaz	
Atributo	***Tipo***
Nombre	TextBox
Para cada responsabilidad:	
Nombre:	Adicionar_Fase.

Funcionalidad:	Brindará la funcionalidad de Adicionar una Fase.
Nombre:	Cancelar_ Adicionar Fase.
Funcionalidad:	Brindará la funcionalidad de cancelar la operación y regresar a la vista anterior.

Nombre: CE Usuario	
Tipo de clase: entidad	
Atributo	***Tipo***
Usuario	String

Nombre: CE Rol	
Tipo de clase: entidad	
Atributo	***Tipo***
Id_Rol	Integer
Nombre	String
Descripción	String
Roles	String

Nombre: CE Tareas	
Tipo de clase: entidad	
Atributo	***Tipo***
Id_Tarea	Integer
nombre	String
Fecha_inicio	Datatime
Fecha_fin	Datatime

Nombre: CE Proyecto	
Tipo de clase: entidad	
Atributo	***Tipo***
Id_proyecto	TextBox
Nombre	TextBox
Fase	ComboBox
Fecha_Fin	ComboBox
Líder	TextBox

Usuario	TextBox
Teléfono	TextBox
Departamento	TextBox
Documentos	TextBox
Lista_Módulos	ComboBox

Nombre: CE Fase	
Tipo de clase: entidad	
Atributo	***Tipo***
Nombre	String

Anexo 7. Descripción de las tablas de la base de datos

Nombre: Usuario		
Atributo	***Tipo***	***Descripción***
Id_usuario	integer	Identificador de la tabla.
Nombre	varchar(255)	Contiene el (los) nombre(s) del usuario.
Apellido	varchar(255)	Contiene el (los) apellido(s) del usuario.
Lista_roles	varchar(255)	Contiene todos los roles que le pueden ser asignados a los usuarios de acuerdo a sus necesidades dentro de la aplicación.

Nombre: Tarea		
Atributo	***Tipo***	***Descripción***
Id_tarea	integer	Identificador de la entidad.
Nombre	varchar(255)	Contiene el (los) nombre(s) de la tarea.
Responsable	varchar(255)	Es el encargado de de verificar el

		estado de las tareas que fueron asignadas.
Evaluación	varchar(255)	Se obtiene una evaluación de la tarea realizada.
Descripción	varchar(255)	Es una breve descripción de la tarea.
Estado	varchar(255)	Se especifica el avance de la tarea si fue o no cumplida o se encuentra en ejecución.
Subtarea	varchar(255)	
Fecha_inicio	date	Fecha para la cual la tarea se empieza a desarrollar.
Fecha_fin	date	Fecha en la cual la tarea será terminada.
Tipo_tarea	varchar(255)	El tipo de tarea asignada ya sea investigativa o de pruebas.
Asignado	varchar(255)	Usuario que va a ser el encargado de realizar la tarea encomendada.
Fichero	varchar(255)	Se encuentran los documentos que serán utilizados para el desarrollo de la tarea.
Lista_usuarios	varchar(255)	Lista de todos los usuarios del proyecto.

Nombre: Rol		
Atributo	***Tipo***	***Descripción***
Id_rol	integer	Identificador de la tabla.
Nombre	varchar(255)	Contiene el (los) nombre(s) del rol.
Descripción	varchar(255)	Breve descripción del rol que se desempeña.

Nombre: Proyecto		
Atributo	***Tipo***	***Descripción***
Id_proyecto	varchar(255)	Identificador de la tabla.
Fecha_fin_fase	Datatime	Fecha para la cual termina la fase en que se encuentra el proyecto.
líder	varchar(255)	Es el encargado de dirigir el proyecto.
usuario_líder	varchar(255)	Es el usuario del líder del proyecto.
teléfono	varchar(255)	Es el teléfono del líder del proyecto.

documentos	varchar(255)	Documentos pertenecientes al proyecto.
departamentos	varchar(255)	Son los departamentos en los que se encuentran los proyectos.
lista_módulos	varchar(255)	Lista de todos los módulos perteneciente al proyecto.
fase	varchar(255)	Fase en la que se encuentra el proyecto.
nombre	varchar(255)	El nombre del líder del proyecto.

Nombre: Fase		
Atributo	***Tipo***	***Descripción***
Nombre	varchar(255)	Contiene el (los) nombre(s) del usuario.