Universidad de las Ciencias Informáticas

Facultad 4

Título: “Herramienta para informatizar los procesos en el Laboratorio de Pruebas de Liberación.”

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autoras: Delmys Pozo Zulueta.
Nilien Cruz Palenzuela.

Tutores: Ing. Violena Hernández Aguilar.
Ing. Roig Calzadilla Díaz.
Ing. Delvis Echeverría Pérez.
Ing. Asnier Góngora Rodríguez.

Ciudad de la Habana, julio 2008.

“Año del 50 Aniversario del triunfo de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Delmys Pozo Zulueta.

_____
Firma del autor

Nilien Cruz Palenzuela.

_____
Firma del autor

Ing. Violena Hernández Aguilar.

_____
Firma del tutor

Ing. Roig Calzadilla Díaz.

_____
Firma del tutor

Ing. Delvis Echeverría Pérez.

_____
Firma del tutor

Ing. Asnier Góngora Rodríguez.

_____
Firma del tutor

"La alegría más profunda del corazón

es como un imán que señala el camino de la vida.

Uno debe seguirlo,

aunque se entre en una senda plagada de dificultades".

Madre Teresa de Calcuta.

DATOS DE CONTACTO

Tutora: Ing. Violena Hernández Aguilar

Tel: (07) 2024588 (Ciudad Habana, Playa) **e-mail:** violena@uci.cu

- Ingeniera Informática del Instituto Superior Politécnico José Antonio Echeverría (2005).
- Profesora de la Universidad de las Ciencias Informáticas, en la Disciplina de Ingeniería y Gestión de Software desde el año 2005.
- Cuenta con 3 años de trabajo en la Educación Superior.
- Cursa la Maestría en Gestión de Proyectos Informáticos.
- Se desempeña laboralmente como Especialista General de la Dirección de Calidad de la infraestructura Productiva de la UCI.

Tutora: Ing. Roig Calzadilla Díaz

Tel: (07) 835 8877 (UCI) **e-mail:** rcalzadilla@uci.cu

- Ingeniero en Ciencias Informáticas. UCI 2007
- Profesor de la Universidad de las Ciencias Informáticas, en la Disciplina de Ingeniería y Gestión de Software desde el año 2007.
- Cursa la Maestría en Calidad de Software.
- Se desempeña laboralmente como Especialista General de la Dirección de Calidad de la infraestructura Productiva de la UCI.

Tutor: Ing. Delvis Echeverría Perez.

Tel: (07) 8378877 (UCI) **e-mail:** decheverria@uci.cu

- Ingeniera en Ciencias Informáticas de la Universidad de las Ciencias Informáticas.
- Profesora de la Universidad de las Ciencias Informáticas, en las asignaturas de Gestión de Software y Base Datos
- Cuenta con 10 meses de trabajo en la Educación Superior.
- Es asegurador de calidad de software en el proyecto MINPPAL-CNBA.

Tutor: Ing. Asnier Góngora Rodríguez.

Tel: (07) 8378877 (UCI) **e-mail:** agongora@uci.uci

- Ingeniero en Ciencias Informáticas graduado de la Universidad de las Ciencias Informáticas (2007).
- Profesor de la Universidad de las Ciencias Informáticas, en la Disciplina de Ingeniería y Gestión de Software desde el año 2007.
- Cuenta con 9 meses de trabajo en la Educación Superior.
- Cursa la Maestría de Calidad de Software.
- Se desempeña laboralmente como Asegurador de la calidad del proyecto Mapeo Cerebral Humano Cubano de la Facultad 6.

AGRADECIMIENTOS GENERALES

Han sido 5 años llenos de momentos inolvidables, a nuestro lado siempre hemos tenido personas que nos han ayudado a salir de los malos momentos, y de festejar juntos los buenos. Esta es una parte difícil ya que tememos a que se nos quede alguien sin mencionar y eso sería un error imperdonable, pero si eso sucediera nos gustaría antes decir algo. "Tu fuiste, eres y serás, el más hermoso regalo de la vida".

Agradecemos a todos los profesores que de una forma u otra aportaron su grano de arena a la preparación que hemos adquirido en los años de estudiante.

A nuestros tutores, por apoyarnos y guiarnos para llegar al final.

A nuestras amigas Maylen, Carmen, Yuri, Yudy y Giselle por haber tenido la suerte de compartir con ellas todos estos años.

A Caridad, Alionuska, y Hermes por la ayuda brindada.

A la Revolución Cubana por habernos dado la oportunidad de estudiar y por brindarnos esta gran oportunidad.

A la Universidad de las Ciencias Informáticas por ser la institución donde vivimos por 5 años y nos ha dado la formación académica para esta profesión.

Delmys:

Quiero agradecerle primero que a nadie a Dios, por otorgarme la sabiduría y la salud para lograrlo.

También le agradezco a todos los que de una forma u otra me han ayudado a lo largo de estos 5 años y me han brindado su apoyo para que yo pudiera terminar con éxito este trabajo de diploma, especialmente:

A mis padres por ser los principales responsables de que yo haya cumplido este gran sueño, sin su apoyo incondicional, amor y entrega completa para conmigo no hubiera podido llegar hasta donde estoy hoy, gracias por ser los mejores padres del mundo, todas las palabras del mundo no serían suficientes para expresarles cuantos los amo y admiro, gracias por estar ahí, justo donde y cuando más los necesito.

A mi novio Eduardo por ser mi consejero, mi guía y mi vida.

A mi hermana, mis abuelos (Erasmo y Orlando), mis tías queridas (Tita, Irma y Raiza), mi tío (Erasmito) y a mis primos (Lídice, Felito y Froilán) por siempre estar presentes incondicionalmente cuando los necesito.

A mi suegra por haberme acogido como una hija.

A Lili por ser mi amiga en estos 5 años y mi compañera de tesis.

A Olguita muchísimas gracias por todo.

A Violena a quien aprecio y admiro muchísimo.

Nilien:

En el transcurso del camino la vida me ha puesto pruebas difíciles, problemas en los que si no hubiera contado con el apoyo y cariño de personas especiales, no habría podido llegar al final. Hoy, a sólo unos pasos de la meta, quiero expresar de todo corazón mis más sincero agradecimientos a esas personas que marcaron mi vida para siempre.

A las personas que más amo en este mundo: A mi mamá linda, por darme el apoyo que tanto necesité en muchos momentos, que me dio fuerza para seguir hasta el final. Al mejor: mi papá, que confió en mí con los ojos cerrados, por su amor, comprensión, por darme fuerzas y apoyo en todo, por ser mi guía y enseñarme el sacrificio, que es el verdadero camino hacia el futuro. Estaré en deuda eternamente por todo lo que han hecho por mí. Les doy mil gracias.

A mi niño Raidelito por ser todo lo que esperé de él. Hoy te pido que no me defraudes.

A mi hermana y sus pequeños, gracias por estar presentes, ustedes hacen mis días más lindos.

A mis tías Mirella, Marisol y Zenaida, gracias por su apoyo incondicional, por hacer suyos mis grandes momentos de tristezas y decisiones difíciles. Sólo me queda decirles que hoy, mi corazón estará abierto para ustedes y podrán contar conmigo para siempre.

Al desastre de mi tío Alonso, a mi tío Félix y Geño. Hoy quiero que sepan que más que una sobrina, quiero que encuentren en mí a una hija que los adora.

A mis primitas lindas Raimarys y Raimy, y los demás que no se pongan celosos porque saben que los quiero mucho.

A toda mi familia, a mis amistades que de una forma u otra con su cariño, amistad y apoyo han colaborado para que se vieran realizados mis sueños. A ustedes gracias por existir.

A mi gran amiga Dayana gracias por los tantos momentos de alegrías, de incertidumbres y de situaciones complicadas. Siempre estarás presente.

A Nidia, por sopórtame y por sobre todas las cosas, por cuidarme a lo que más Amo en la vida.

A mi amiga en el trascurso de estos 5 años, y compañera de tesis. Gracias por tu inmensa ayuda.

A mi novio por ser tan atento, dedicado, comprensivo y hacerme vivir tan buenos momentos a su lado. A su familia en especial a su tía Nancy por enseñarme y hacerme sentir que estaba presente para lo que necesitara.

A Eduardo y a Lucia gracias por acogerme en su casa como si fuera otro miembro de la familia.

Al destino por darme el privilegio de haberlos conocidos.

A todos de corazón muchísimas gracias…..

DEDICATORIA

Le dedico este trabajo con mucho amor y cariño
a las personas que más amo: mi familia.
En especial a mis padres por su ayuda incondicional,
a mis sobrinos (Eramys y Anthony) y a mis primos (Dairene y Aché)
para que concienticen que su futuro está en sus manos
y que siempre que se quiere se puede
todo está en proponerselo.
Delmys Pozo Zulueta.

A mi mamá y mi papá los ángeles de mis sueños
por ser los dos seres que me dieron la vida,
que poseen una sonrisa agradable y sincera
que me brindan la calma, el amor y la comprensión.
Por apoyarme con dulces palabras
para alcanzar todas mis metas,
en fin a esas dos personas que adoro
y son sin dudas,
lo más importante para mí.
Nilien Cruz Palenzuela

RESUMEN

La producción de software está estrechamente vinculada al control de la calidad en cada una de las etapas de su desarrollo. Con este objetivo, fue creado el Laboratorio Central de pruebas de liberación de la UCI, siendo su principal tarea la realización de diferentes pruebas para determinar la calidad de los productos desarrollados. En el laboratorio actualmente el proceso del control de la calidad de software se realiza de forma manual, debido a esto existen dificultades, por lo que el presente trabajo de diploma pretende dar respuesta a la siguiente interrogante: ¿Cómo optimizar los procesos de planificación, gestión y seguimiento de las No Conformidades y los recursos en el laboratorio de pruebas de Liberación de la UCI? Para el logro de este objetivo se implementó una herramienta para informatizar estos procesos. Esta aplicación permitirá, a los Asesores de Calidad y al personal autorizado, interactuar a través de la red con la información almacenada en la aplicación así como hacer reportes de seguimiento para llevar un control de todo lo referente a las pruebas de calidad de software.

Palabras claves: Software, Calidad, Pruebas.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Dado el alto grado de desarrollo tecnológico, el campo de desarrollo de software [1]va en aumento cada día. Actualmente estos software están sometidos a un conjunto de exigencias relacionadas con la producción y la calidad, para compensar estas exigencias se hace necesario la ejecución de una renovada perspectiva en la creación de productos de software que sea lo más cercana posible a una disciplina de ingeniería y no a las prácticas y modos artesanales que desafortunadamente se ha estado aplicando en varias ocasiones.

En la actualidad diversas empresas de diferentes países luchan por estar en la cima del mercado de software; desarrollando sistemas que tengan una excelente calidad. La calidad y el proceso de desarrollo de software están fuertemente relacionados, ya que el producto que presente una calidad óptima presentará una mayor demanda en el mercado del software.

La industria cubana de software está ascendiendo diariamente, debido a que Cuba se ha visto en la necesidad de informatizar las esferas de la sociedad como lo son la educación, la salud, las Fuerzas Armadas Revolucionarias (FAR), etc., así como la realización de software para su comercialización nacional e internacional. El término **desarrollo de software** en Cuba como en la Universidad de las Ciencias Informáticas (UCI) está en notable aumento por lo que se requiere el máximo esfuerzo de los desarrolladores para que desde su inicio el software posea la calidad[2] exigida y requerida.

La UCI fue creada hace 6 años por el Comandante en Jefe Fidel Castro Ruz, entre sus tareas tiene el fin de incursionar en el desarrollo del software ayudando además en la informatización de la sociedad cubana. Al surgir la UCI aumenta en gran por ciento la producción de software a nivel nacional y en medio de este desarrollo se observó que la calidad de sus productos no era la mejor, por lo que en el año 2005 es creado el Laboratorio de Pruebas de Liberación [3]en la UCI con el fin de certificar[4] con un sello de calidad los productos que estén listos para ser comercializados internacionalmente y conjuntamente medir la calidad de los productos que se están produciendo en la universidad. Además se creó en cada facultad grupos de calidad con el objetivo de garantizar que exista documentación, lograr mejoras en el proceso de desarrollo de software y garantizar que se prueben las aplicaciones que realizan en cada facultad.

La principal actividad del Laboratorio de Pruebas de Liberación está el proceso de pruebas a las aplicaciones de software. En el laboratorio actualmente el proceso de pruebas al software se realiza de forma manual, debido a esto existen dificultades como:

- No se realiza una planificación[5] real de los recursos.
- No existe un control total de los recursos que se encuentren disponibles, ni del trabajo realizado, ni del estado real de los medios computarizados disponibles para realizar el trabajo.
- La planificación actual trae consigo pérdida de tiempo.
- La asignación máquina–estudiante no se realiza de forma eficiente.
- No se cumple con el tiempo asignado para la revisión de la calidad del producto a prueba.
- No existen reportes que muestren el seguimiento[6] del trabajo realizado en cada una de las iteraciones en la revisión del producto.
- Existen limitaciones en el empleo del tiempo real por parte del estudiante para realizar las diferentes pruebas de software por las dificultades que se presentan con la planificación del mismo.
- Pérdida de tiempo y derroche de los recursos (debido a esto no se puede hacer una buena gestión de las No Conformidades[7] y de los reportes de las mismas).
- La accesibilidad, por parte de los usuarios, a la información de los reportes de las No Conformidades de manera rápida es insuficiente pues no existe una publicación de la misma en la red, ni un buen intercambio de información entre los asesores de calidad y el quipo de desarrollo.
- No hay seguridad ni integridad de la documentación que se entrega por parte del equipo de desarrollo, pues la misma no cuenta con ninguna protección.

Por la situación anterior se llegó a la conclusión que el **problema a resolver** está dado por:

¿Cómo optimizar procesos de planificación, gestión y seguimiento de las No Conformidades y los recursos en el laboratorio de pruebas de Liberación de la UCI?

Se trazó como **objetivo** realizar el análisis, diseño e implementación de un sistema que permita planificar, gestionar y dar seguimiento a las No conformidades y a los recursos del Laboratorio de Pruebas de Liberación de la UCI.

Por tanto el **objeto de estudio** del presente trabajo es: Procesos de planificación, gestión y seguimiento de las No conformidades y los recursos en un Laboratorio de Pruebas y su informatización.

El **campo de acción** está determinado por Sistemas que informaticen los procesos de Planificación, gestión de No Conformidades generados por las pruebas y seguimiento de los recursos.

Para dar cumplimiento al objetivo expuesto anteriormente se propone obtener como **posible resultado** una herramienta que informatice procesos en el laboratorio de pruebas de liberación de la UCI, para lo cual se realizan las siguientes **tareas**:

- Estudiar los Sistemas automatizados existentes vinculados al campo de acción.
- Realizar un análisis de las herramientas y tecnologías actuales para llevar a cabo el proceso de implementación.
- Realizar el flujo de trabajo de análisis y diseño.
- Realizar la Implementación de la herramienta informática que se propone.
- Realizar las pruebas al sistema.

La investigación está estructurada por: Introducción, tres Capítulos, Conclusiones, Recomendaciones, Referencia bibliográfica, Bibliografía, Glosario de términos y Anexos.

Capítulo 1. Fundamentación Teórica: En el presente capítulo se plasma una breve explicación del flujo actual de los procesos a informatizar. Se realiza una descripción de algunos de los sistemas automatizados que existen vinculados al campo de acción, así como las tendencias y tecnologías

actuales relacionadas con el objeto de estudio. Se presentan todos los elementos teóricos que sustentan la investigación realizada.

Capítulo 2. Análisis y Diseño del Sistema: El presente capítulo refleja las clases del análisis de los casos de uso más significativos, se representan los diagramas de secuencia del diseño, además diagramas de clases de diseño Web, propios de este tipo de aplicación, se tiene en cuenta, diagrama de clases persistentes para una mejor comprensión, de qué parte de la información se almacena.

Capítulo 3. Implementación y Prueba: En el presente capítulo se construye el modelo necesario para desarrollar el proceso de implementación del sistema con los diagramas de componentes definidos. También se elabora el diagrama de despliegue donde se representan los nodos necesarios, en los que se distribuye la aplicación. Además se abordarán los temas relacionados con las pruebas aplicadas al software.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo se plasma una breve explicación del flujo actual de los procesos a informatizar. Se realiza una descripción de algunos de los sistemas automatizados que existen vinculados al campo de acción, así como tecnologías actuales relacionadas con el objeto de estudio. Se presentan todos los elementos teóricos que sustentan la investigación realizada.

1.2 Situación actual de los procesos en el laboratorio de pruebas.

1.2.1 Proceso de Planificación.

El proceso de planificación tiene la misión de garantizar una eficaz planificación de los recursos de un laboratorio de calidad. Del mismo se debe obtener como resultado la planificación de todas las pruebas a realizar a un software en el laboratorio de pruebas de liberación. Dentro de las actividades existentes en el mismo están:

- **Decidir las pruebas que se le aplicarán a un software.**

Para la realización de esta tarea los especialistas de calidad luego de aceptar la revisión de un software y en dependencia de lo que se quiera probar del software, decide cuales tipos de prueba se realizarán. Se obtiene como resultado el listado de las pruebas que se aplicarán a un software.

- **Determinar las métricas[8] que se le aplicarán a la prueba.**

En esta actividad el especialista de calidad una vez que decida las pruebas que se le aplicaran a un software decide que métricas utilizar para medir el posible tiempo de duración de las pruebas, estas métricas son por cada tipo de prueba y complejidad de la misma. Las métricas se utilizarán a la hora de la planificación de las pruebas. Se obtiene como resultado las métricas a aplicar a cada tipo de prueba.

- **Determinar los recursos involucrados en la prueba.**

En esta actividad el jefe de prueba evalúa los recursos con que cuenta para la realización de las pruebas, ya sean recursos humanos o tecnológicos, o sea determina la cantidad de estudiantes que tiene disponible, así como la cantidad de computadoras con que cuenta en ese momento.

- **Realizar la planificación de la prueba.**

Una vez que se tengan todos los recursos, las métricas y la prueba que se va a aplicar se procede a la planificación de la prueba donde se determina por turnos de trabajo la fuerza necesaria a intervenir y los recursos tecnológicos así como una posible fecha de comienzo y fin de la prueba [1] .

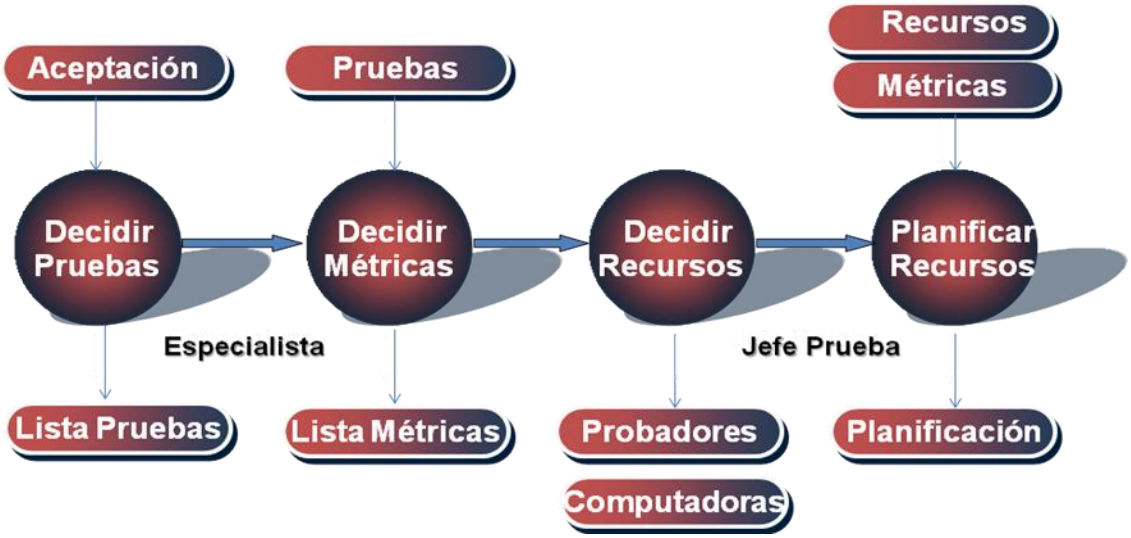


Figura 1: Esquema del proceso de Planificación.

1.2.2 Proceso de Gestión de las No Conformidades.

Este proceso tiene como finalidad garantizar que se realicen las pruebas a un software y obtener los reportes correspondientes. Del mismo se debe alcanzar como resultado las no conformidades a un software así como las respuestas a las mismas y los reportes correspondientes a ellas. Dentro de las actividades existentes del mismo se encuentran:

- **Comunicar la revisión del software.**

Después que se decide comenzar la revisión del software por la dirección de calidad, los Especialistas de calidad, seleccionan a un Jefe de prueba y le comunican al mismo que tiene que comenzar la revisión de un software. El Especialista de calidad le entrega al Jefe de prueba toda la documentación que hace falta para la revisión del software en cuestión.

- **Repartir el trabajo entre los probadores.**

Una vez que el Jefe de prueba tenga toda la documentación necesaria para la revisión del software, este comienza con la distribución del trabajo entre los probadores. Se comenzará a repartir los elementos de prueba entre los probadores en dependencia de la complejidad del caso de uso y de la cantidad de turnos de trabajo con que se cuente. Así queda elaborado un informe con la asignación del trabajo de los probadores.

- **Confección de los diseños de caso de prueba.**

Cuando el probador sepa los elementos de prueba que tiene que revisar, este comienza la confección del diseño de caso de prueba que le ayudará, para encontrar las no conformidades al software. Por tanto se confecciona un diseño de caso de prueba por cada elemento de prueba.

- **Revisión del software.**

El Probador con el diseño de caso de prueba terminado procede a la revisión del software. En ella se irán encontrando no conformidades, con las cuales se confeccionará un documento que contendrá las no conformidades encontradas al software. Una vez que el Probador termine la revisión el Jefe de prueba revisará las no conformidades encontradas para dar culminación a esta iteración de la revisión del software.

- **Comunicar a los desarrolladores la culminación de una iteración de la revisión del software.**

El Jefe de prueba le comunica al Especialista de calidad que se terminó una iteración de la revisión del software y al mismo tiempo se lo comunica a los Desarrolladores, entregándoles las no conformidades encontradas de esta revisión en un informe de no conformidades.

❖ **Responder las no conformidades.**

Los Desarrolladores leen las no conformidades encontradas por los Probadores y comienzan a elaborar un documento dándole respuestas a estas no conformidades. Una vez que terminan de darle repuestas a las no conformidades le entregan este documento al Jefe de prueba que esté al frente de la revisión del software.

❖ **Comunicar la culminación de la revisión del software al solicitante.**

Cuando el Jefe de prueba le informe al Especialista de calidad que en una iteración de la revisión del software no se encontraron no conformidades o que terminó con la revisión de un software, este elabora un informe de culminación de la revisión del software y se lo informa el solicitante, dando por terminado la revisión del software [2].

Figura 2: Esquema del proceso de Gestión de las No Conformidades**.**

1.2.3 Proceso de Seguimiento.

El seguimiento consiste en el análisis y recopilación sistemáticos de información a medida que avanza un proyecto. Su objetivo es mejorar la eficiencia y efectividad de un proyecto y organización. Se basa en metas establecidas y actividades planificadas en las distintas fases del trabajo. Ayuda a que se siga una línea de trabajo, y además, permite a la administración conocer cuándo algo no está funcionado.

Si se lleva a cabo adecuadamente, es una herramienta de incalculable valor para una buena administración. Proporciona la base para la evaluación permitiendo determinar si los recursos disponibles son suficientes y están bien administrados. Además controla si la capacidad de trabajo es eficiente, adecuada y si se está haciendo lo que está planificado.

Este proceso puede considerarse como indispensable para la realización de cualquier actividad, su elemento más representativo es la retroalimentación, entendida como la información que resulta de comparar los objetivos y las metas de cada eje estratégico con los resultados obtenidos al momento de la evaluación, lo que permite conocer la brecha recorrida y el camino por recorrer entre el estado actual del sistema y la visión al futuro del mismo, las acciones que conviene mantener y las rectificaciones que sea necesario realizar [3].

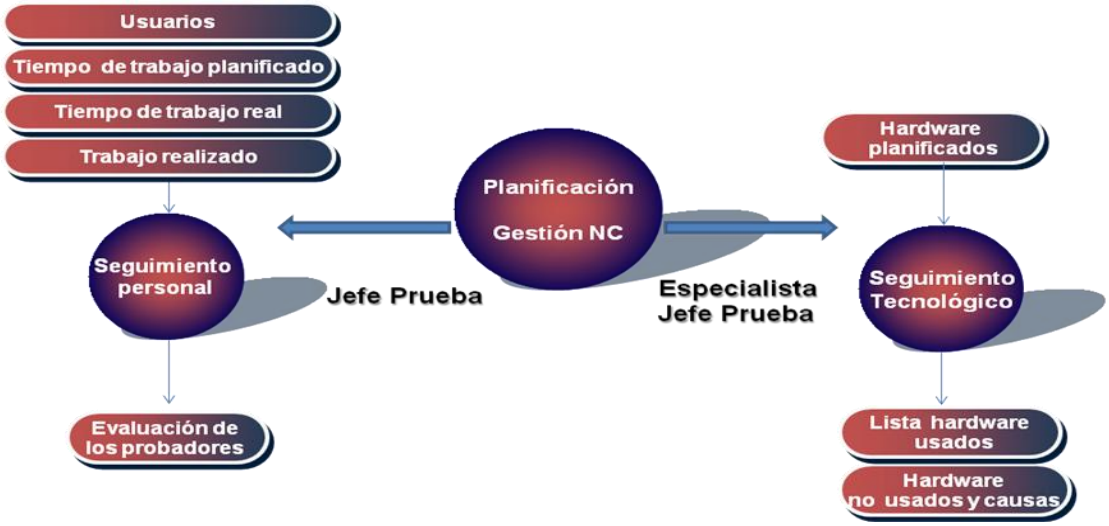


Figura 3: Esquema del proceso de Seguimiento.

1.3 Sistemas automatizados existentes vinculados al campo de acción.

En la actualidad existen herramientas que informatizan los procesos de Planificación, Gestión de la No Conformidades y Seguimiento de los recursos, entre las que se pueden mencionar, el Sistema para la Gestión de No Conformidades (SIGNO), el Gestor de Proyectos, el Project KickStart, entre otras. A continuación se muestra una descripción de cada una de estas herramientas.

1.3.1 SIGNO - Sistema para la Gestión de No Conformidades.

SIGNO permite registrar toda la información de una no-conformidad, asignar responsables (automática o manualmente), reportar la respuesta remedial, reportar el problema y sus causas, definir el plan de acción (acciones correctivas/preventivas), visualizar mediante semáforos la posible demora en ejecutar acciones, reportar la ejecución de acciones, reportar la auditoría realizada, cerrar no-conformidades y realizar consultas específicas por diferentes criterios. Adicionalmente permite todo el manejo de usuarios y motivos para hacer una fácil adaptación a las necesidades de su organización.

SIGNO es una aplicación que se encuentra en funcionamiento en empresas colombianas, es totalmente basado en la web; por lo tanto, para su manejo sólo se requiere de un navegador o browser reciente y una conexión al servidor donde se instale la aplicación[4].

1.3.2 Gestor de proyectos.

El Gestor de Proyectos es un programa que brinda muchas facilidades, ahorra tiempo, permite planificar y administrar los proyectos empresariales, organizando y efectuando un seguimiento de las tareas de forma efectiva para impedir demoras y ajustarse al presupuesto. Mediante el uso del gestor se puede calcular previsiones precisas de gastos y hacer un seguimiento de los tiempos y la facturación, así como de las personas que ejecutan las operaciones. Brinda la posibilidad de diseñar, gestionar y evaluar los proyectos de la organización, facilitando la distribución de la información entre las diversas áreas de la empresa y auditando que las labores asignadas a cada trabajador se efectúen de forma correcta y conforme a los plazos establecidos.

- El software de gestión de proyectos garantiza la seguridad y confidencialidad de la información.
- Controla proyectos y tareas.
- Gestiona su avance, plazos, costes, esfuerzos, recursos (grupos de personas, perfiles y persona).
- Controla los cambios que se producen en los proyectos.
- Modifica fácilmente los plazos, recursos y esfuerzos.

1.3.3 Project KickStart.

Es un programa de gestión de proyectos de fácil uso. Con este sistema de gestión de proyectos, es muy cómodo y rápido planear un proyecto utilizando el calendario del menú descendente y la gráfica de Gantt que especifiquen desiguales detalles como las fechas y responsabilidades. El Project KickStart ayuda a concentrar la atención en la estructura del proyecto, los objetivos, los recursos, los riesgos y las cuestiones estratégicas que son significativas para el éxito del proyecto. Establece un "enlace dinámico" de su información con Microsoft Project, Word, Outlook, PowerPoint ó Excel. Posee diversas características que hace reconocer sus ventajas las cuales se plasman a continuación.

- Se puede utilizar en proyectos de cualquier tamaño.
- Siete tipos de informes preestablecidos.
- Listado de sugerencias en los archivos de Objetivos, Fases y Obstáculos.

1.3.4 Software de seguimiento y control de tareas.

Es un software diseñado para mantener el control y el estado de las tareas y documentos que circulan por una oficina desde que se originan hasta que se cierran. Esta magnífica herramienta da la posibilidad de poder incluir documentos anexos a cualquier tarea y de manejar un número ilimitado de las mismas. Cuenta con un módulo de seguridad para restringir accesos al sistema. Está diseñado para utilizarse en Internet, su interfaz es intuitiva y fácil de usar. (Software 2006). Permite la clasificación de los asuntos mostrados por cualquier columna de la ventana. También, permite mantener el control de los asuntos y documentos que llegan, así como los que son delegados a un determinado tipo de usuario, manteniendo informado al personal de las fechas de inicio, vencimiento y terminación que estos guardan.

1.3.5 Generación, control y seguimiento de recursos propios (GCSRP).

El GCSRP es una plataforma colaborativa que admite acceder a las diferentes fuentes de información relacionada con la generación de ingresos propios, buscando la integración de las actividades y procesos administrativos a los sustantivos de la entidad, con una visión de calidad y servicio. Este software permite brindar el conocimiento de los ingresos en una empresa desde sus mismas fuentes generadoras, además posibilita la comparación en distintos períodos de tiempo, establece estrategias

y da seguimiento a las mismas, a través de un monitoreo constante. Entre sus características más trascendentes se destaca que está desarrollado en software libre lo que implica una reducción considerable de costos, una alta confidencialidad y gran capacidad de adaptación.

Como resultado del análisis de las herramientas expuestas anteriormente se llegó a la conclusión que muchas de estas herramientas a pesar de ser muy eficaces, son propietarias, o su utilización no es aplicable en Cuba ni en el Laboratorio de Pruebas de Liberación, puesto que tienen precios muy costosos y no cubren todas las necesidades o simplemente no se enmarcan dentro de las características del los procesos de planificación, gestión y seguimiento de la no conformidades y los recursos en el Laboratorio.

No existe ninguna herramienta que informatice estos procesos que haya sido realizada en Cuba y tampoco en la universidad es por esto que se decide implementar una herramienta para la planificación, la gestión de las no conformidades y el seguimiento de los recursos adaptada a nuestro entorno y necesidades, integrando estos tres procesos en una sola aplicación.

1.4 Tecnologías actuales.

1.4.1 Lenguajes de programación Web.

PHP

PHP (PHP Hypertext Pre-processor) es un lenguaje de programación del lado del servidor gratuito rápido, con una gran librería de funciones, mucha documentación y fácil de aprender. Este lenguaje es usado generalmente para el desarrollo de sitios Web, además presenta un estilo clásico, es decir, está contenido por bucles, variables, sentencias condicionales, funciones, entre otros. Este lenguaje de programación no requiere de definición de tipos de variables y manejo de excepciones. Además se puede utilizar PHP para generar archivos PDF, Flash, así como imágenes en diferentes formatos etc. Es un lenguaje multiplataforma que disfruta de una amplia documentación en su página oficial. Puede usarse con la mayoría de los sistemas operativos, ejemplo Windows o basados en Unix. Presenta una gran capacidad de conexión con una marcada cantidad de gestores de base de datos que son utilizados actualmente (ejemplo: PostgreSQL, Oracle, dbm, filePro, interbasem), aunque su mayor conectividad, la base de datos con la que mejor trabaja es con MySQL. El mismo lee y manipula datos

desde disímiles fuentes, incluyendo datos que pueden inscribir los usuarios desde formularios HTML. Permite crear los formularios para la Web, así como las técnicas de Programación Orientada a Objetos. Es Libre, por lo que se presenta como una alternativa de fácil acceso para todos.

Es un lenguaje que posee muchas ventajas a su favor; aunque en ocasiones pueden mezclarse las sentencias HTML y PHP por lo que se ve afectada la claridad del código.

Ventajas

- Software libre.
- Multiplataforma.
- Fácil de utilizar.
- Fácil de integrar con MySQL.
- Permite crear los formularios.
- Código potable, de fácil interpretación y rápida ejecución.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

JavaScript

Java Script es un lenguaje que no requiere compilación, con capacidades elementales orientadas a objeto. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML. Está diseñado para controlar la apariencia y manipular los eventos dentro de la ventana del navegador Web y es soportado por la gran mayoría de los navegadores, aunque no está diseñado para grandes aplicaciones.

Sin que conste una comunicación a través de la red, una página HTML con JavaScript incrustado puede interpretar, y anunciar al usuario con una ventana de diálogo, de que las entradas de los formularios no son válidas. O bien efectuar algún tipo de acción como ejecutar un fichero de sonido o un Applet de Java.

Java

Está inspirado en C++ y se proyectó con la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos ya sea a nivel de código fuente como a nivel de código binario.

Entre sus principales características se encuentra que es un lenguaje de propósito general y orientado a objetos. Su sintaxis ha sido trabajada mejorando la de C++ logrando mayor sencillez y legibilidad. Presenta mayor robustez al simplificar la gestión de memoria y eliminar las complejidades del manejo explícito de punteros. Presenta capacidades avanzadas de ejecución multihilo y proporciona mecanismos de carga dinámica de clases en tiempo de ejecución. Se puede compilar y ejecutar en cualquier plataforma de sistema operativo por ejemplo en Windows, Solaris o Linux gracias a su máquina virtual. Su desarrollo ha sido rápido y exitoso debido a la gran cantidad de grandes empresas colaboradoras que han dado su aporte para enriquecerlo.

La opinión de la mayoría de los programadores es que se torna más potente en el desarrollo de aplicaciones Web y ha sido mayormente orientado en ese sentido por sus promotores.

La ejecución de programas escritos en Java suele comportarse más lenta que la de aplicaciones de otro lenguaje y hace un uso voraz de recursos como memoria y procesador. Esto se hace más notorio si la ejecución se basa en cálculos matemáticos complejos o si la aplicación presenta un diseño cargado de componentes visuales.

¿Por qué PHP y JavaScript?

Se utilizan estos dos lenguajes porque PHP además de ser uno de los más universales que existe en la actualidad presenta gran velocidad a la hora de procesar los datos y es soportado por varias plataformas, conjuntamente con JavaScript que se encargaría de las validaciones en la entrada de datos del sistema.

1.4.2 Metodología de desarrollo de software.

En todo proceso de desarrollo de software siempre puede existir el riesgo que este sea difícil de controlar y arriesgado, de ahí surge la necesidad de utilizar una metodología de desarrollo de software, para así evitar resultados impredecibles y detección tardía de errores. En un proyecto de desarrollo de software la metodología define Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo [5]. Utilizar una metodología de desarrollo de software es de gran importancia para los desarrolladores ya que permite entre muchas otras cosas mantener un orden y estructura sobre cómo hacer el software. Entre las metodologías más utilizadas en la actualidad se encuentran RUP y XP, a continuación se menciona en qué consiste y algunas características de cada una de ellas y además se alude cuál de estas utilizará y ¿por qué?

RUP (Rational Unified Process)

RUP es una metodología de desarrollo de software que permite asignar de forma disciplinada tareas y responsabilidades dentro de un equipo de desarrollo, asegurando, si es correctamente aplicado la confección de software con calidad dentro de plazos y presupuestos predecibles.

El Proceso Unificado es un proceso de desarrollo de software (conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software). Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos.

Las características que diferencian el ciclo de vida de RUP de otras metodologías son que es:

- Iterativo e Incremental.
- Centrado en la Arquitectura.
- Guiado por Casos de Uso [5].

Extreme Programing (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo y pequeños equipos de trabajo. Consiste en una programación rápida, cuya

particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Se basa principalmente en pruebas unitarias, re-fabricación y programación en pares.

Entre sus principales características se destacan:

- La comunicación entre los usuarios y los desarrolladores.
- La simplicidad al desarrollar y codificar los módulos del sistema.
- La retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

¿Por qué RUP?

Decidimos utilizar RUP ya que este proceso está dividido en 4 fases, las mismas se dividen en iteraciones, los artefactos son el objetivo principal de cada actividad, es basado en roles, posee mucha documentación, muy organizativo. Viendo así de una forma más descriptiva el papel que juega cada objeto en el sistema. Por todo lo antes dicho y basándonos en lo mismo podemos decir que este proceso de desarrollo es el idóneo para llevar a cabo la descripción de este software.

1.4.3 Herramienta de modelado visual.

Las herramientas de modelado son muy útiles en la creación de modelos software que se desarrollarán, además permiten crear un esbozo a bajo precio y riesgo mínimo. Seguidamente se mencionarán las herramientas de modelado visual que están entre las más utilizadas hoy en día.

Rational Rose

Es una herramienta de entorno visual**,** que provee el modelado basado en UML, es decir permite construir un sistema antes de comenzar a construirlo. Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros de equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las amplias ventajas de Rose es que utiliza la notación estándar en la arquitectura de software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común, además los diseñadores

pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto.

Visual Paradigm

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas.

¿Por qué Racional Rose?

Decidimos utilizar Rose porque admite como notaciones: UML, COM, OMT y Booch. Realiza Chequeo semántico de los modelos. Ingeniería “de ida y vuelta”: Rose permite generar código a partir de modelos y viceversa. Permite un desarrollo multiusuario, integración con modelado de datos. Genera documentación. Tiene un lenguaje de script para poder ampliar su funcionalidad.

1.4.4 Lenguaje para la modelación.

UML (Unified Modeling Languaje).

UML (Unified Modeling Languaje) o Lenguaje de Modelación Unificado es un lenguaje gráfico para especificar, construir, visualizar y documentar las partes o artefactos. (Información que se utiliza o produce mediante un proceso de software).

Pueden ser artefactos: un modelo, una descripción que comprende el desarrollo de software que se basen en el enfoque Orientado a Objetos, utilizándose también en el diseño Web. UML usa procesos

de otras metodologías, aprovechando la experiencia de sus creadores, eliminó los componentes que resultaban de poca utilidad práctica y añadió nuevos elementos.

UML es un lenguaje más expresivo, claro y uniforme que los anteriores definidos para el diseño Orientado a Objetos, que no garantiza el éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios.

Entre sus principales características tenemos que posee tecnología orientada a objetos, algo muy importante es que el cliente participa en todas las etapas del proyecto, es aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor.

UML como solución:

La elección del lenguaje para modelar depende de muchos factores. Muchos han creído ver el UML como solución para todos sus problemas, sin conocer en muchos casos de lo que se trataba en realidad. UML es una notación, es decir, una serie de reglas y recomendaciones para representar modelos. Nos da la posibilidad de permitir documentar y especificar los elementos creados mediante un lenguaje común describiendo modelos. UML surge como respuesta al primer problema identificado para contar con un lenguaje estándar y poder escribir planos de software.

1.4.5 Arquitectura

La arquitectura está orientada a una aplicación Web, específicamente el patrón 3 capas. Este modelo se basa fundamentalmente en una aplicación, la cual está dividida en tres capas que tienen diferentes propósitos e interactúan entre sí pero de forma independiente, de forma tal que al hacer cambios en una no se afectan los otros dos , básicamente estos módulos o capas reciben el nombre de:

- **Capa Cliente:** Es la capa encargada de hacer el pedido y mostrar la infamación al usuario, es donde va la interfaz de usuario.
- **Capa intermedia o de negocio:** Esta capa es la encargada de recibir el pedido del cliente, extrae los datos al interactuar con la capa de datos y le da una respuesta al cliente tal y como él la pidió, es donde van implementadas todas las reglas de negocio.

- **Capa servidor o de datos**: Esta capa es la encargada de guardar los datos que serán utilizados por la aplicación.

1.4.6 Sistemas de Gestión de Bases de Datos (SGBD).

Los **Sistemas de gestión de base de datos** son un prototipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan facilitándole al usuario las herramientas que le permitan manipular o manejar de manera clara, sencilla y ordenada un conjunto de datos. Este sistema tiene la **ventaja** de permitir la facilidad de manejo de grandes volúmenes de información, que alcancé gran velocidad en muy poco tiempo, no hay probabilidad de duplicidad de información, nos permite la comprobación de información en el momento de introducir la misma. Seguridad de información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta. Y otro de los beneficios que nos aporta este gestor es independencia del tratamiento de información.

MySQL

MySQL Database Server es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones, es la base de datos de código fuente abierto más usada del mundo. Su ingeniosa y astuta arquitectura lo hace extremadamente rápido y fácil de personalizar.

La extensiva reutilización del código dentro del software, ha dado lugar a un sistema de administración de la base de datos incomparable en:

1. Velocidad.
2. Compactación.
3. Estabilidad.
4. Facilidad de despliegue.

Una de las principales **ventajas** de MySQL es que es multiplataforma, posee una estabilidad comprobada, gran rapidez a la hora de ejecutar las consultas, conectividad segura.

Como sus trascendentales **características** podemos señalar que soporta hasta 32 índices por tabla, soporta gran cantidad de tipos de datos para las columnas, aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo, gran portabilidad entre sistemas.

PostgreSQL

PostgreSQL es un servidor de base de datos objeto relacional libre, es un motor de base de datos.

Una de las principales **ventajas** de PostgreSQL es que es multiplataforma, diseñado para ambientes de alto volumen. Está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows, usa una estrategia de almacenamiento de filas para conseguir una mejor respuesta en ambientes de grandes volúmenes.

Podemos mencionar como una **desventaja** de este gestor de Base de datos que consume bastante recursos y carga más el sistema.

Como sus trascendentales **características** podemos señalar que posee alta concurrencia, Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (foreign keys), disparadores, vistas, integridad transaccional, herencia de tablas.

¿Por qué MySQL?

Utilizamos la versión 5.0.45 ya que su principal objetivo de diseño fue la velocidad, consume muy pocos recursos, tanto de CPU como de memoria, además de ser rápido, robusto, multitarea, multiusuario tiene un enfoque relacional. Es seguro y usa Listas de Control de acceso para todas las conexiones, consultas y otras operaciones. Trabaja en diferentes plataformas, soporta múltiples idiomas. Está completo y optimizado. Tiene una excelente integración con PHP, también tiene un control de acceso de los de los usuarios bastante amplio. No tiene límites en el tamaño de los registros. Posee extraordinarias utilidades de administración como backup y recuperación de errores.

1.4.7 Servidor Web.

Un servidor Web es un programa que le brinda grandes servicios a un navegador escuchando la petición que este le hace y devolviendo siempre algún resultado HTML. Este programa utiliza el protocolo HTTP(protocolo de transferencia de hipertexto) esperando las solicitudes que le realicen desde cualquier puerto TCP, cuando recibe alguna petición busca el recurso y lo envía a su lugar de origen, luego posteriormente se mantiene en la espera de alguna otra solicitud.

Apache

Apache es un servidor Web de código fuente abierto el cual funciona en plataformas virtuales muy utilizadas. Es el servidor Web de código fuente abierto más eficaz del mundo. Este servidor trabaja con la mayoría de lenguajes de la Web de la actualidad, como CGI, Perl, PHP, y otros lenguajes Script. En estos momentos ocupa un lugar importante entre los servidores a escala mundial. Un 49.57 % de los desarrolladores internacionalmente utilizan el servidor Apache [6]**.** Esta popularidad entre los desarrolladores de sitios web se debe a sus diversas ventajas las cuales son:

- **Modular :**

  Es un servidor altamente configurable, con diseño modular. Presenta una arquitectura modular ya que consta de diversos módulos que ofrecen mucha de las funcionalidades que podrían considerarse primordial para un servidor Web. Estos módulos permiten la construcción del propio servidor mediante paquetes de pequeño tamaño.

- **Open Source:**

  Al ser un software de código fuente abierto permite que cualquier persona que posea conocimientos en la programación de C o Perl pueda modificarlo para agregar o modificar una determinada función. Además Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto, sin ninguna puerta trasera[7].

- **Multi-plataforma :**

  Se puede utilizar tanto en sistemas operativos Unix (Linux, Solaris, Mac-OS), como en sistemas Windows, existiendo versiones diferentes para cada uno de los sistemas.

- **Gratuito :**

  Como es un servidor Web de código abierto, por lo tanto puede ser modificado y distribuido gratuitamente.

1.4.8 DreamWeaver 8

Dreamweaver 8 es un editor de HTML visual, diseñado para desarrolladores profesionales. Dreamweaver hace muy fácil y cómodo el crear complejas páginas Web dinámicas, con la conocida técnica de "arrastrar y soltar", permitiendo que los diseñadores puedan crear entornos Web y animaciones sofisticadas sin tener que escribir una sola línea de código.

Cumple perfectamente el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, además muy fáciles de usar:

- Hojas de estilo y capas.
- Java script para establecer efectos e interactividades.
- Implantación de archivos multimedia.

Es un programa que se puede actualizar con componentes, que fabrica tanto Macromedia como otras compañías, para realizar otras acciones más avanzadas. Permiten agregar ágilmente diseño y funcionalidades a las páginas, sin la necesidad de programar manualmente el código HTML. Se puede introducir o crear tablas, editar marcos, trabajar con capas, insertar comportamientos JavaScript, de una forma muy natural, sencilla y visual.

1.4.9 WAMP 5

Es un paquete de instalación que puede instalar en conjunto Apache, PHP 5, MySQL y PHPMyAdmin. W.A.M.P. son las siglas de Windows + Apache + MySQL + PHP, es decir, un instalador paso a paso que contiene en un solo paquete estas tres tecnologías para montar tu propio servidor bajo Windows.

El software que se instala con WAMP5 contiene los siguientes servidores y programas:

- Apache 2.2.6 El servidor de páginas Web más extendido del mercado.
- PHP5. El motor renovado del lenguaje.
- MySQL. La base de datos más extendida para utilizar con PHP.
- PHPmyadmin. Un software que permite administrar una base de datos a través de una interfaz Web.
- SQLitemanager. Un sistema para administrar una base de datos a partir de sentencias SQL [8].

1.5 Artefactos de entrada.

Este trabajo es continuidad de tres trabajos de diplomas realizados anteriormente en los cuales se definen los Procesos de Planificación, Gestión de la No Conformidades y Seguimiento de los recursos en el laboratorio de pruebas de liberación de la UCI. Además se define el modelo de casos de usos del negocio (Ver Anexo 1), los trabajadores que desarrollan las actividades del negocio y otras personas involucradas. Se identifican los requisitos funcionales y no funcionales, los diagramas de casos de usos del sistema (Ver Anexo 2, 3 y 4) y las descripciones de estos casos de uso. A continuación mostramos los requisitos funcionales que soportará nuestra aplicación.

1.5.1 Requisitos Funcionales.

1. Gestionar Pruebas.
2. Gestionar Métrica.
3. Insertar datos del proyecto.
4. Gestionar datos del proyecto.
5. Planificar pruebas.
6. Gestionar planificación.
7. Gestionar Usuario.

8. Autentificar Usuario.
9. Buscar tipo de prueba.
10. Buscar elemento de prueba.
11. Gestionar asignación de trabajo.
12. Gestionar Probador.
13. Gestionar elemento de prueba.
14. Mostrar datos de los elementos de prueba.
15. Gestionar No Conformidades.
16. Gestionar las respuestas de las No Conformidades.
17. Gestionar Interrupción del trabajo.
18. Mostrar estados de las pruebas.
19. Visualizar Reporte.
20. Gestionar la visualización de las No Conformidades.
21. Gestionar datos de Jefe de prueba.
22. Gestionar datos de Probador.
23. Evaluar Probador.
24. Seguimiento PC.
25. Insertar cursos optativos.

1.6 Conclusiones.

En este capítulo se abordó el estado del arte relacionado a la informatización de los procesos de planificación, gestión y seguimiento de las no conformidades y los recursos, además se estudio y se analizaron las tendencias y tecnologías actuales, brindando las definiciones necesarias y precisas para comprender el por qué de las tecnologías que serán utilizadas en el desarrollo de la herramienta software para automatizar los procesos en el Laboratorio de pruebas de Liberación.

- Se ha escogido el lenguaje de programación PHP por las posibilidades multiplataforma que este brinda, además por su posibilidad de uso gratuito.
- El editor seleccionado para la programación de este fue DreamWeaver, por la facilidad en su uso.

- El lenguaje utilizado para la metodología fue el de modelación UML y la herramienta CASE empleada fue Racional Rose.

En este trabajo de diploma se realizará una aplicación la cual será utilizada en el Laboratorio de Pruebas de Liberación pues como bien se muestra en los requisitos funcionales permitirá insertar los datos de un software al cual se le aplicarán la pruebas, brindará la planificación de los recursos involucrados en la realización de las pruebas según los datos introducidos, insertar los datos de la no conformidades y darle respuesta a las misma por parte de los desarrolladores del software además de otras funcionalidades.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL SISTEMA.

2.1 Introducción

El presente capítulo refleja las clases del análisis de los casos de uso más significativos, se representan los diagramas de secuencia del diseño, además diagramas de clases de diseño Web, propio de este tipo de aplicación, se tiene en cuenta, el diagrama de clases persistentes, para una mejor comprensión, de qué parte de la información se almacena.

2.2 *Análisis*

2.2.1 Modelo de análisis.

Este modelo es usado para representar la estructura global del sistema, describe la realización de casos de uso, sirve como una abstracción del Modelo de Diseño y se centra en los requerimientos funcionales.
El Modelo de Análisis puede contener: las clases y paquetes de análisis, las realizaciones de los casos de uso, las relaciones y los diagramas.

2.2.2 Diagrama de clases del análisis.

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas [9]. En el presente capítulo se muestran los diagramas de clases del análisis de los casos de usos más significativos, a los cuales se le dará seguimiento en el diseño e implementación.

Caso de Uso "Autenticarse"

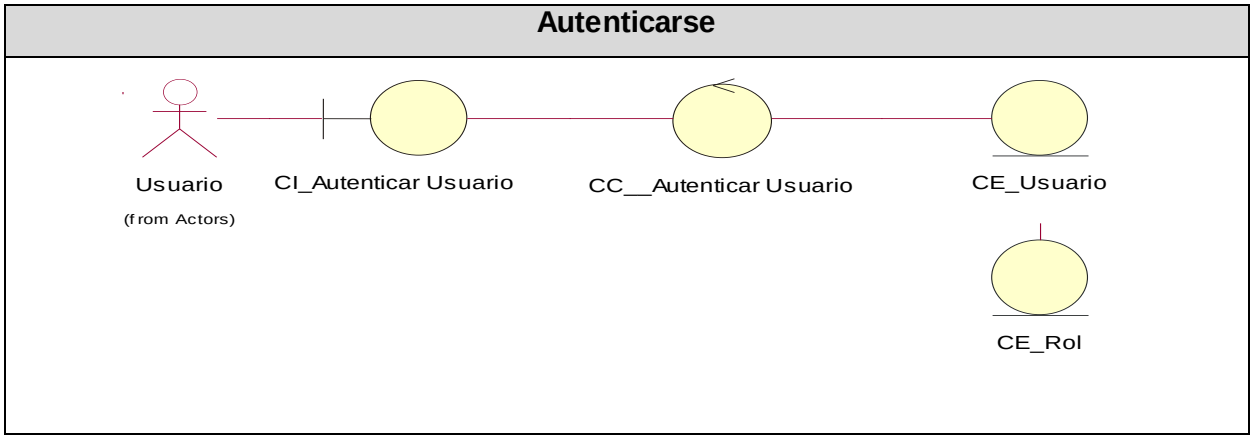


Figura 4: Diagrama de clases del análisis del Caso de Uso Autenticarse.

Caso de Uso "Gestionar pruebas"

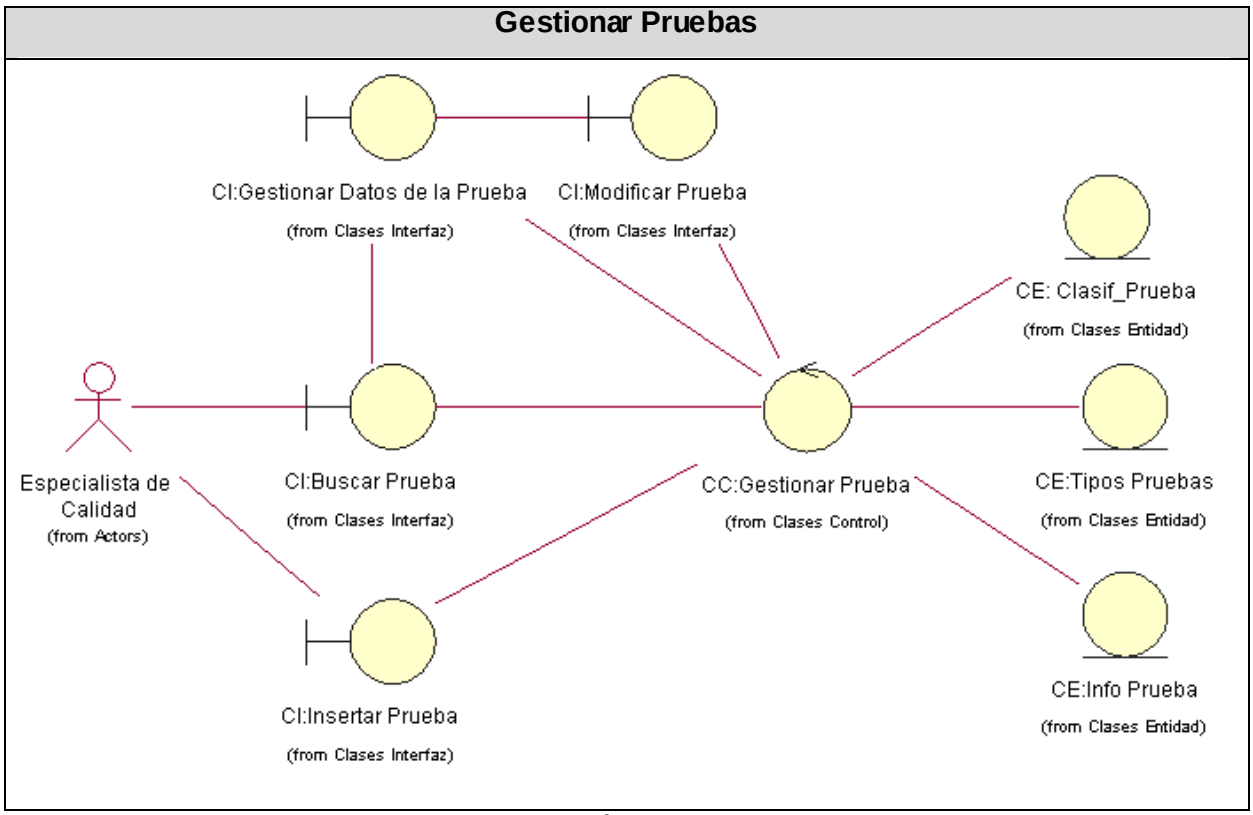


Figura 5: Diagrama de clases del análisis del Caso de Uso Gestionar Prueba.

Caso de uso "Planificar pruebas"

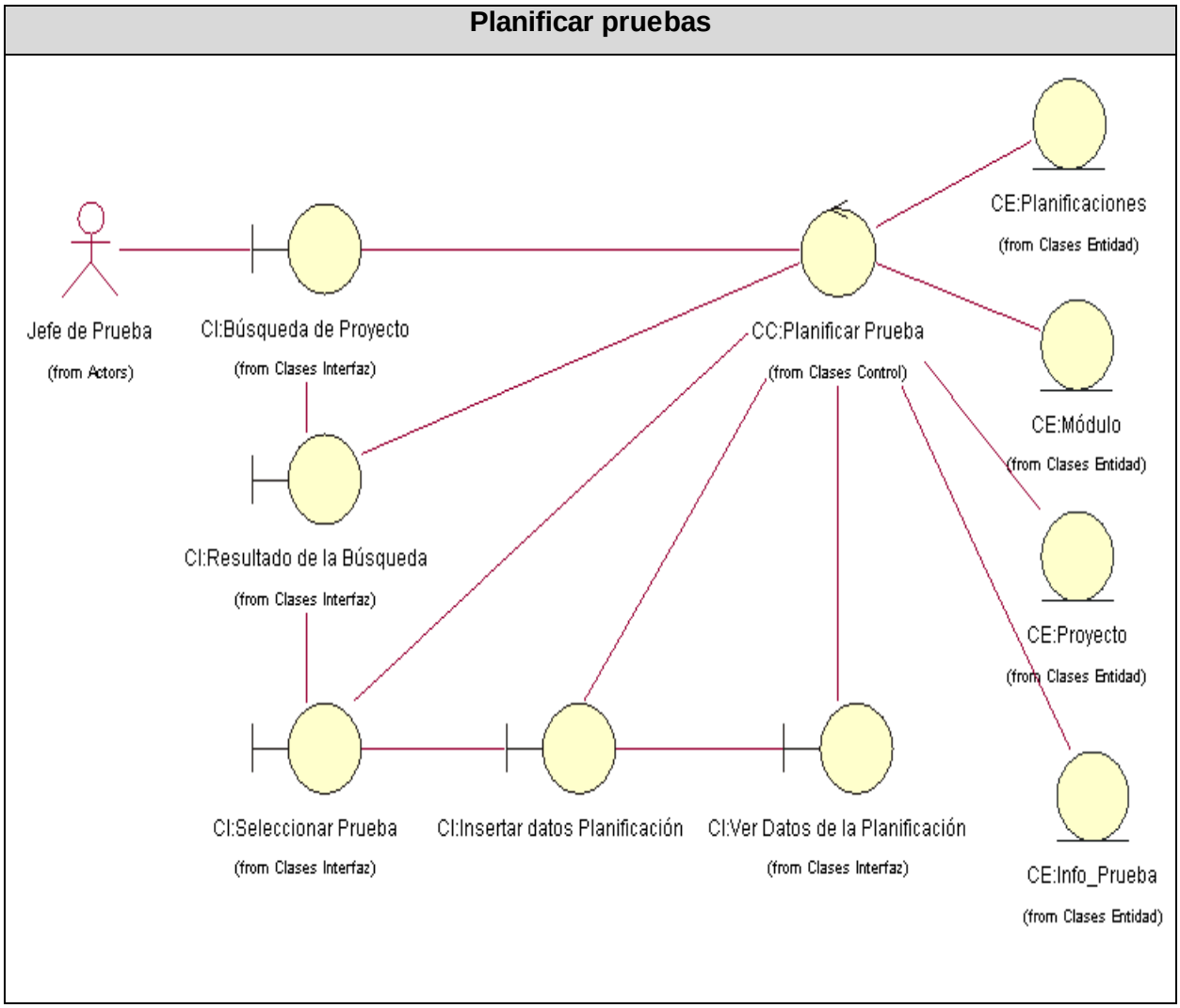


Figura 6: Diagrama de clases del análisis del Caso de Uso Planificar Pruebas.

Caso de uso "Insertar No Conformidades"

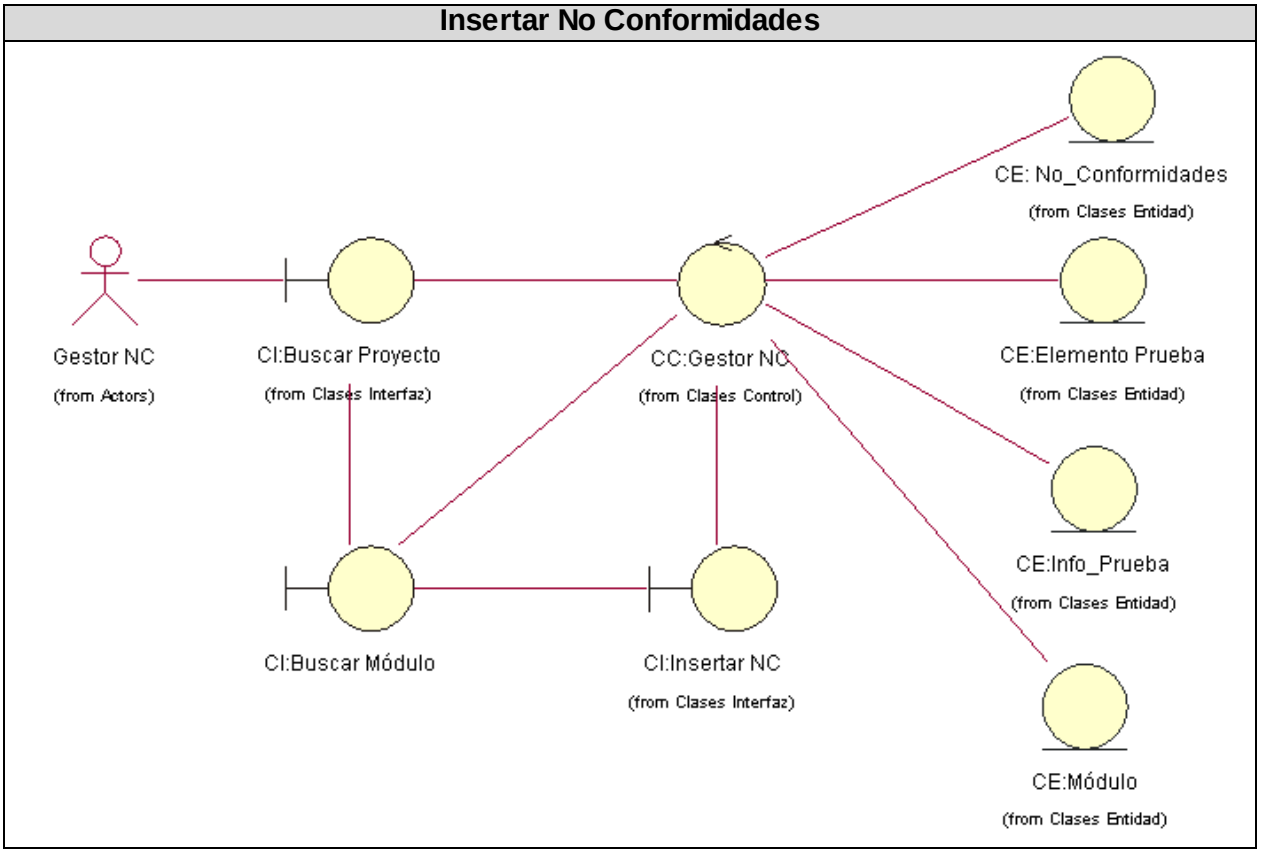


Figura 7: Diagrama de clases del análisis del Caso de Uso Insertar No Conformidades.

Caso de uso "Seguimiento PC"

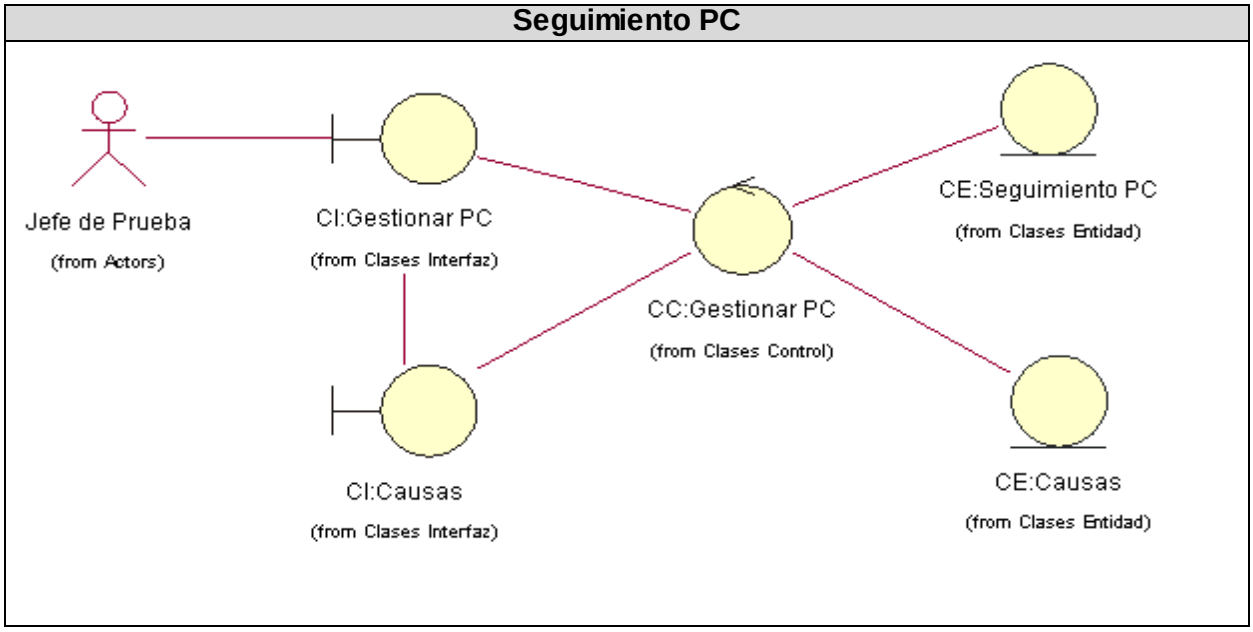


Figura 8: Diagrama de clases del análisis del Caso de Uso Seguimiento PC.

2.3 Diseño

En el diseño modelamos el sistema y encontramos su forma para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema [10].

2.3.1 Modelo de Diseño.

El modelo de diseño es una abstracción de la implementación del sistema. Es usado para concebir un documento del diseño del sistema de SW. Es abarcador, compuesto por artefactos que engloban todas las clases del diseño, subsistemas, paquetes, colaboraciones, y las relaciones entre ellos [10].

2.3.2 Diagrama de Secuencia del Diseño.

Caso de uso "Autenticarse"

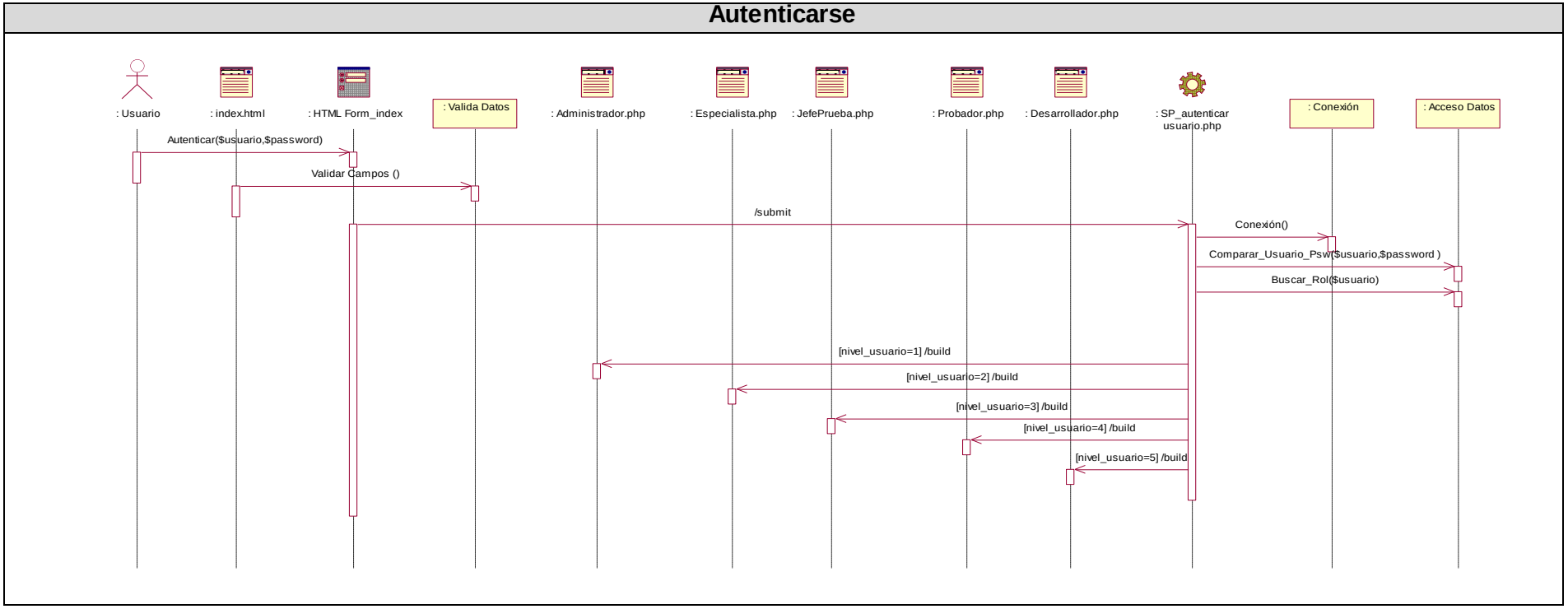


Figura 9: Diagrama de secuencia del diseño, del Caso de Uso Autenticarse.

Caso de Uso "Gestionar pruebas"

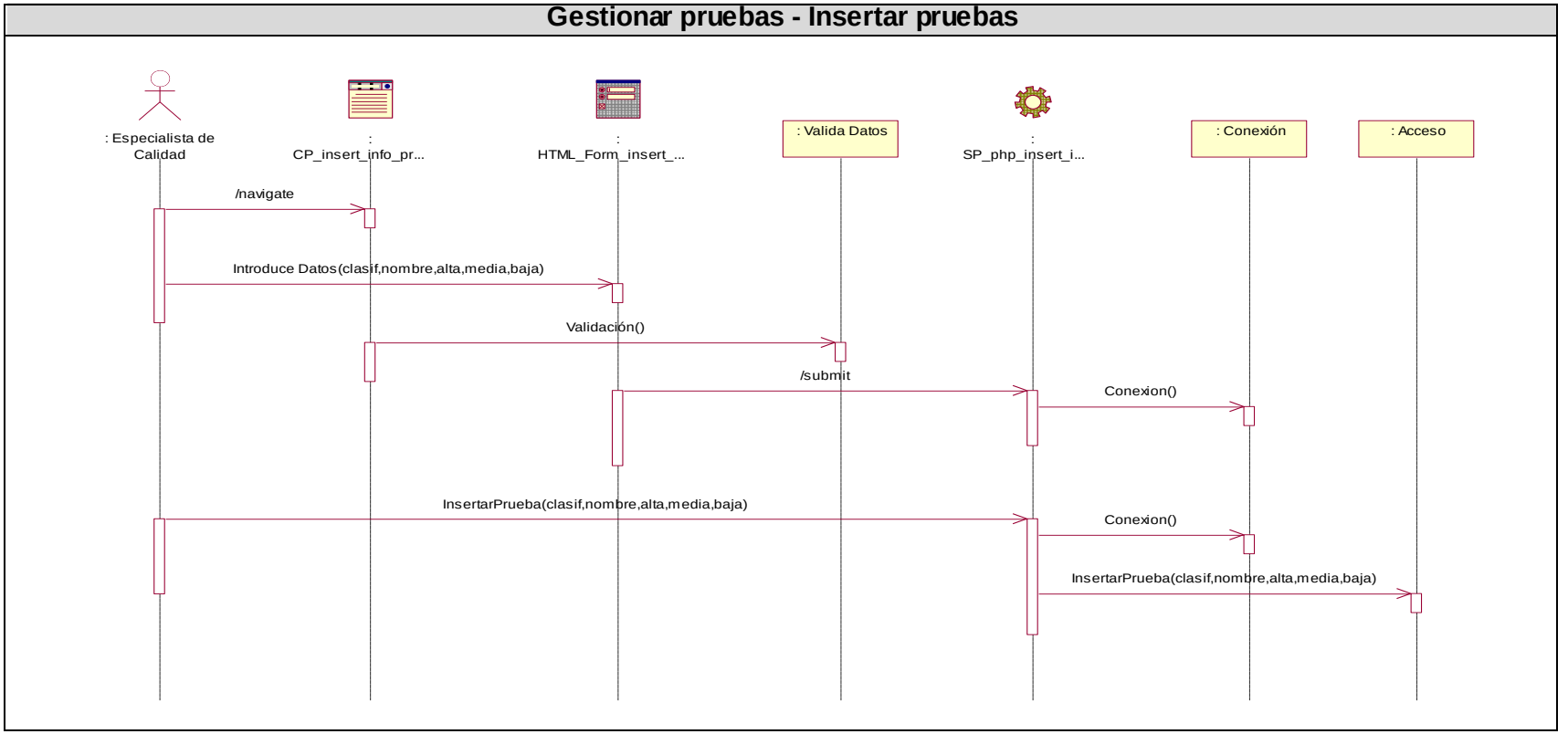


Figura 10: Diagrama de secuencia del diseño, escenario del Caso de Uso Gestionar Prueba (Insertar Prueba).

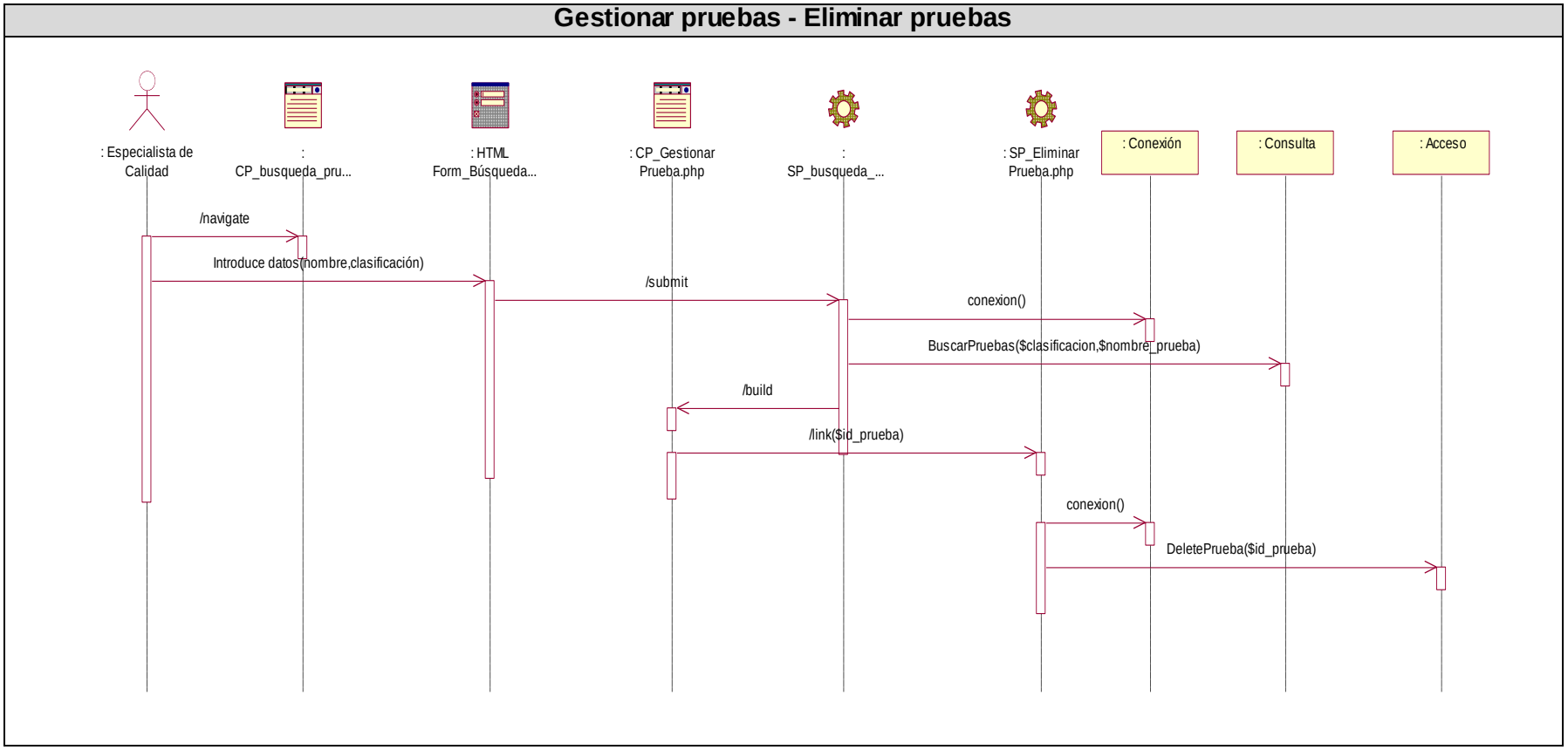


Figura 11: Diagrama de secuencia del diseño, escenario del Caso de Uso Gestionar Prueba (Eliminar Prueba).

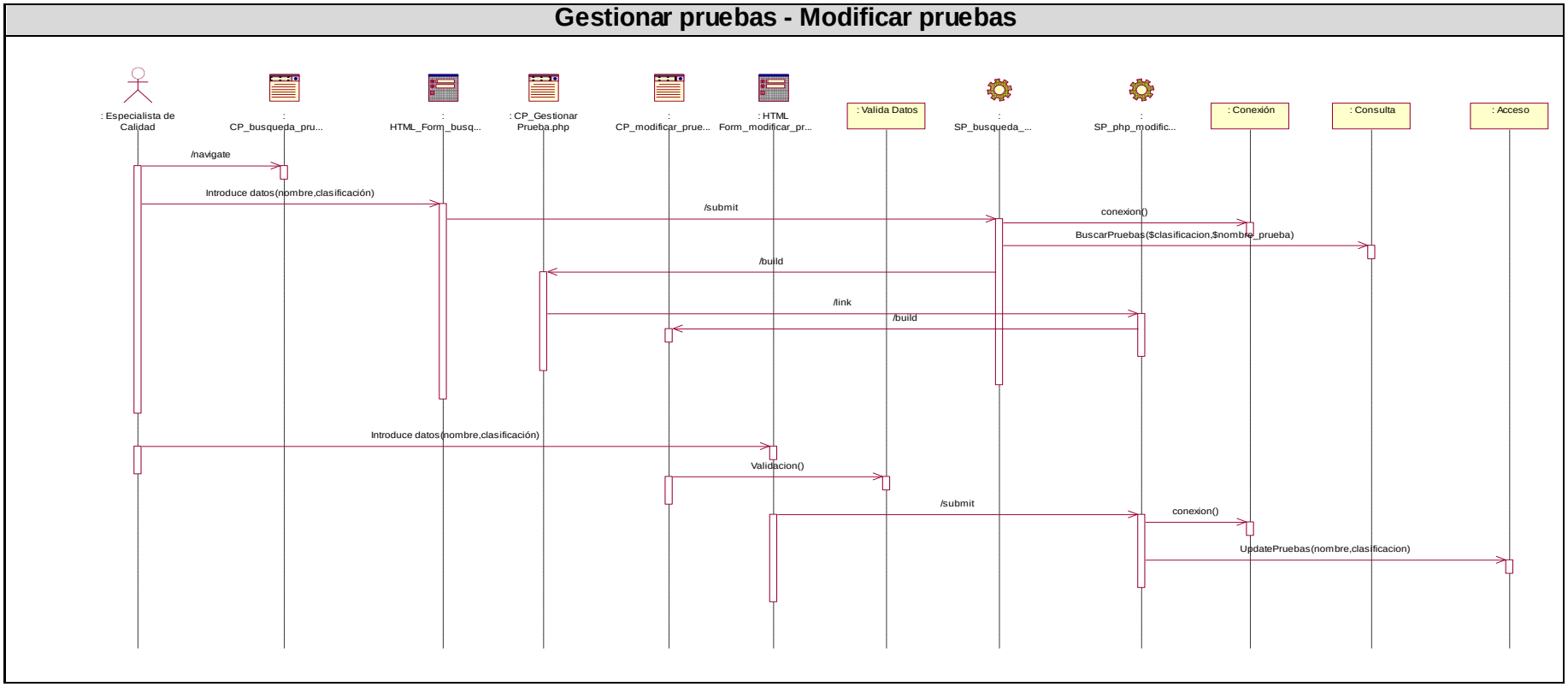


Figura 12: Diagrama de secuencia del diseño, escenario del Caso de Uso Gestionar Prueba (Modificar Prueba**).**

Caso de uso "Planificar pruebas"

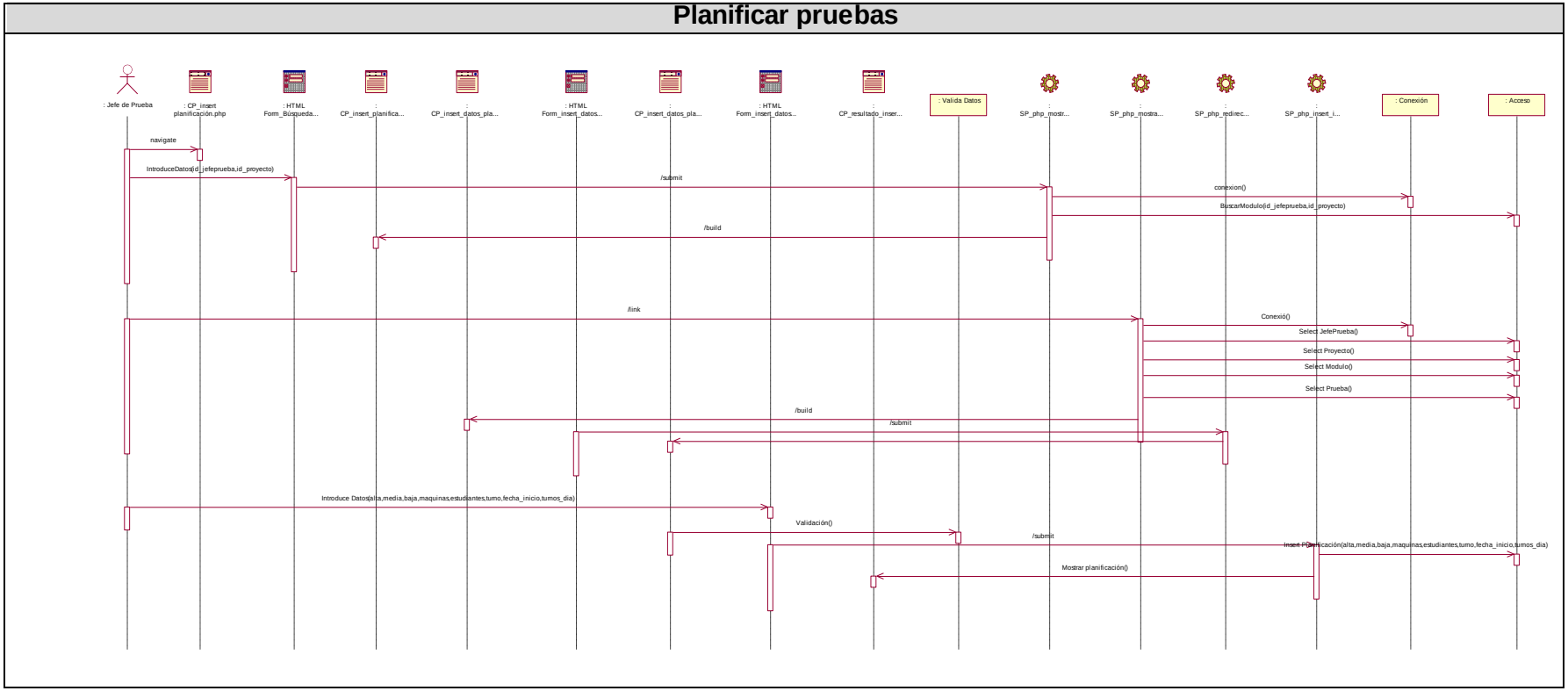


Figura 13: Diagrama de secuencia del diseño, del Caso de Uso Planificar Pruebas.

Caso de uso "Insertar No Conformidades"

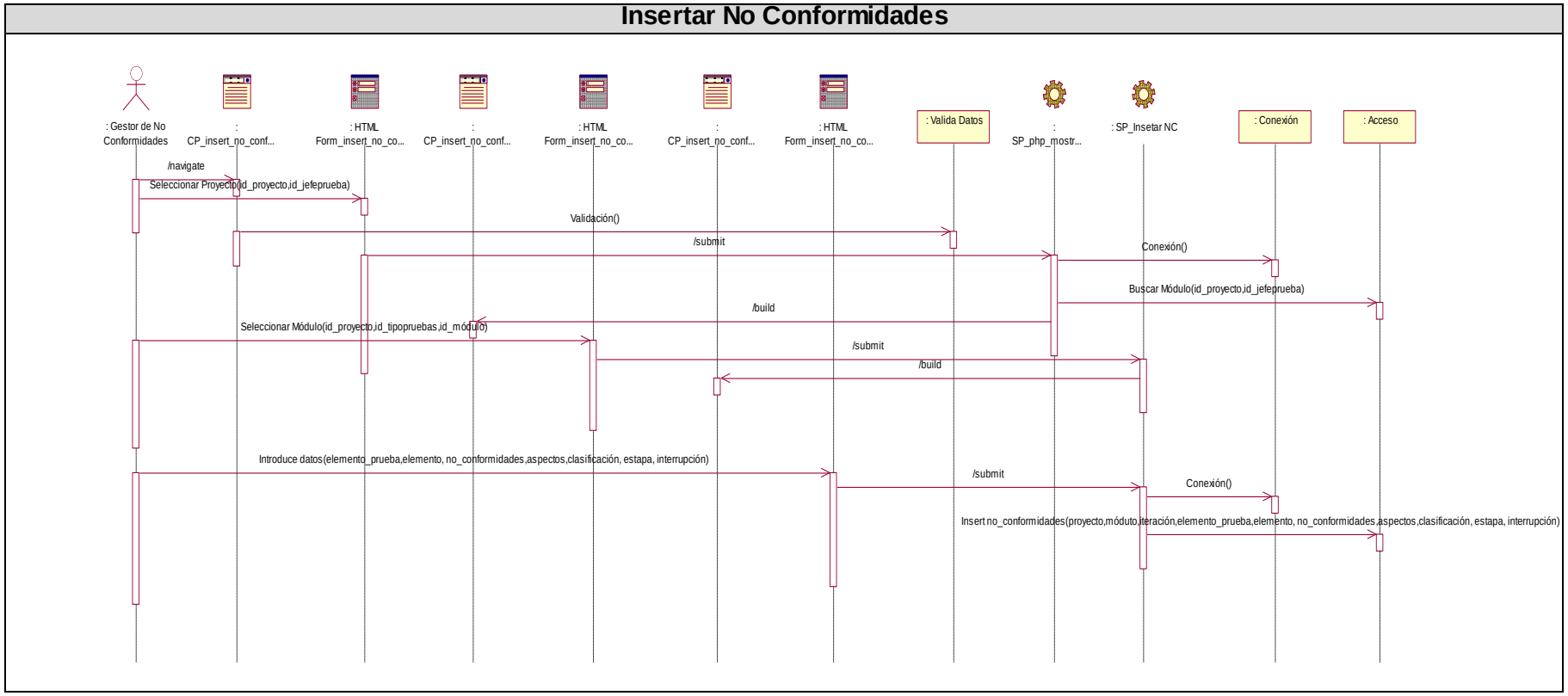


Figura 14: Diagrama de secuencia del diseño, del Caso de Uso Insertar No Conformidades.

Caso de uso "Seguimiento PC"

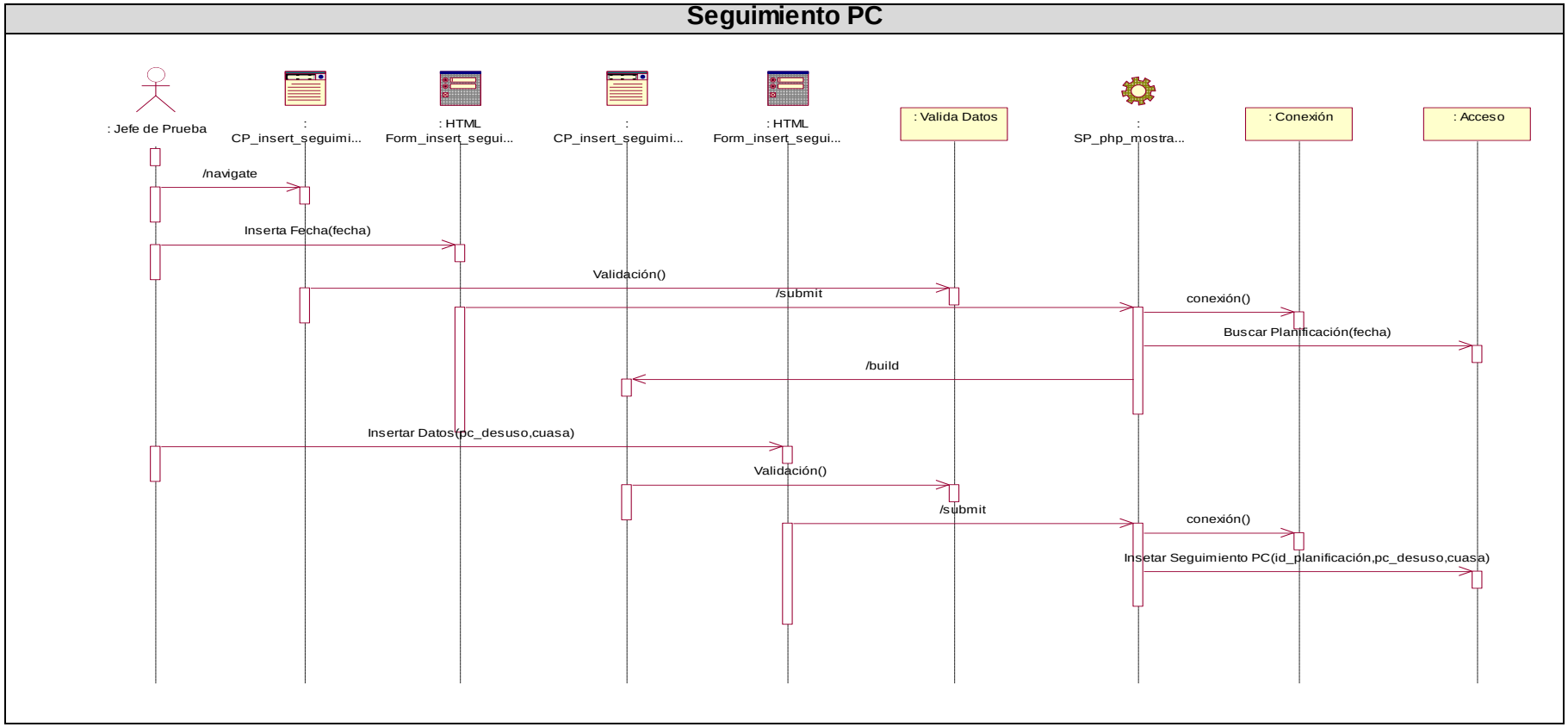


Figura 15: Diagrama de secuencia del diseño, del caso de uso Seguimiento PC.

2.3.3 Diagrama de clase del Diseño

Una clase de diseño es una construcción similar en la implementación del sistema:

- El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación. Las operaciones, atributos, tipos, visibilidad (public, protected y private) se pueden especifican con la sintaxis del lenguaje elegido.
- Las relaciones entre clases de diseño se traducen de manera directa al lenguaje:
  - Generalización: herencia
  - Asociaciones, agregaciones: atributos
- Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.
- Se pueden postergar algunos requisitos a implementación (por ejemplo: manera de nombrar los atributos y operaciones).

Una clase de diseño puede proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación [11].

Los diagramas de clases Web se realizaron empleando estereotipos Web (formularios, páginas clientes, páginas servidoras), para un mayor entendimiento y lógica de los mismos. Se realizó un diagrama de clases de diseño por cada caso de uso.

Caso de Usos "Autenticarse"

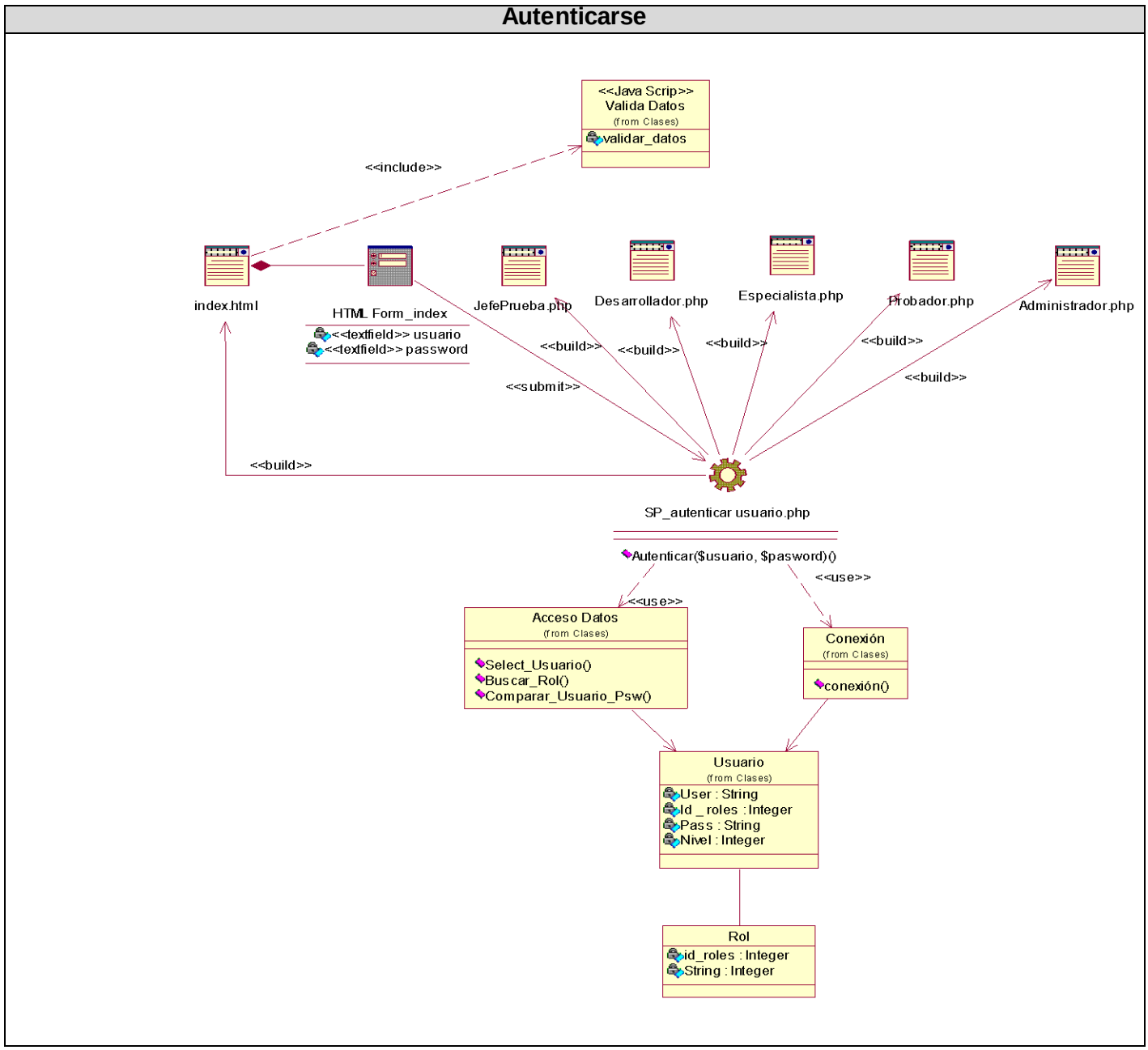


Figura 16: Diagrama de clases del diseño Caso de Uso Autenticarse.

Caso de Uso "Gestionar pruebas"

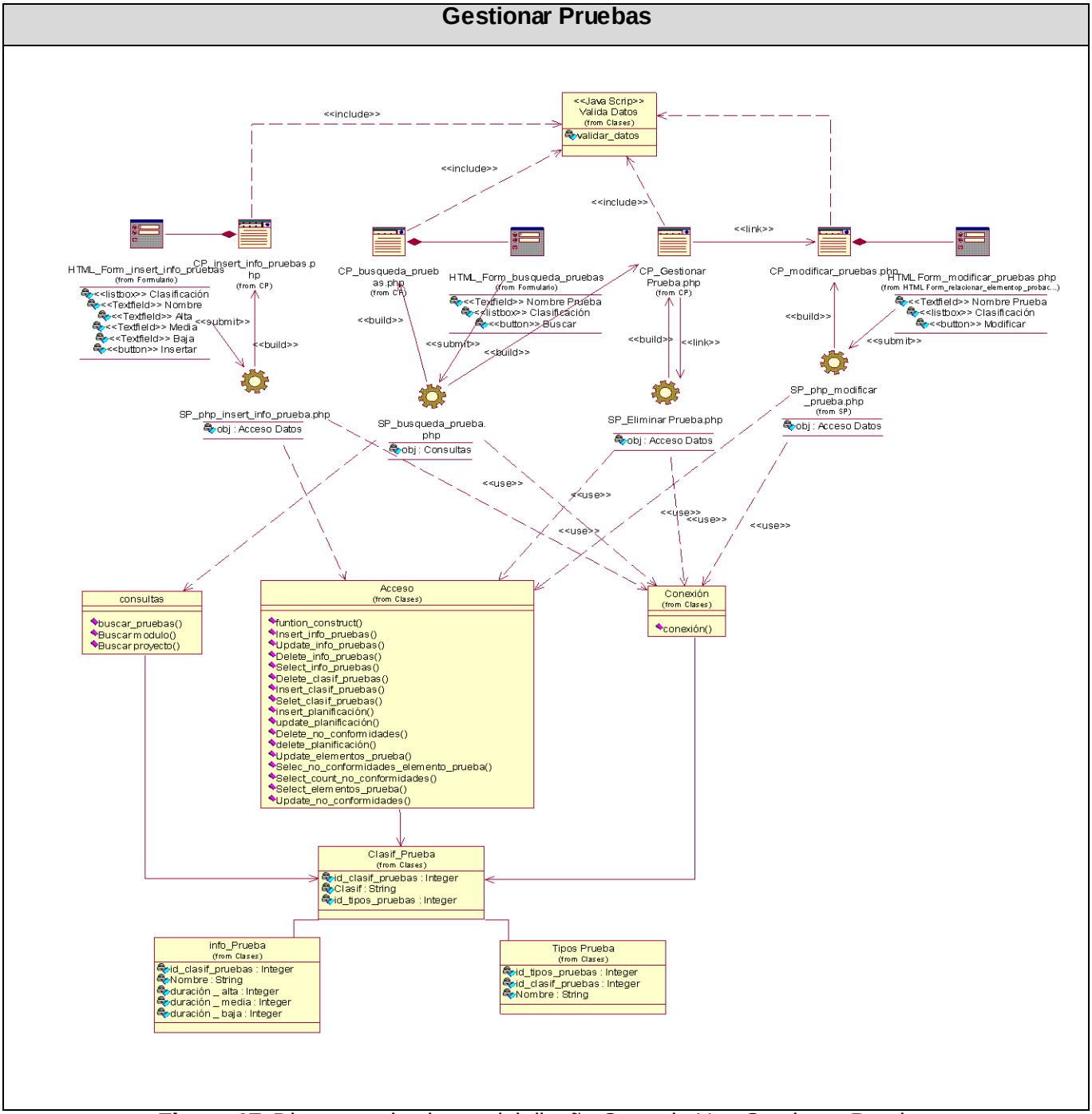


Figura 17: Diagrama de clases del diseño Caso de Uso Gestionar Prueba.

Caso de uso "Planificar pruebas"

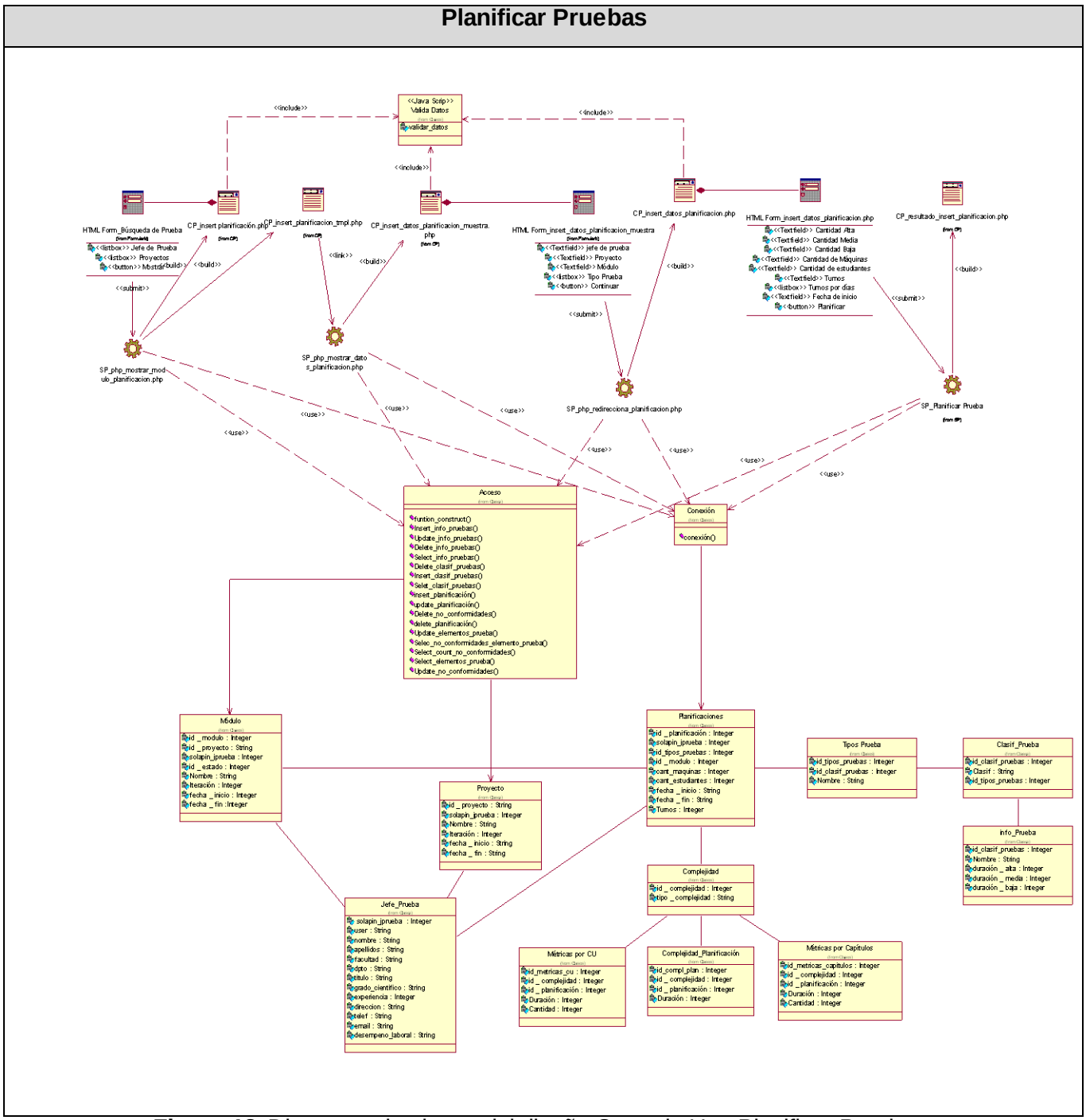


Figura 18: Diagrama de clases del diseño Caso de Uso Planificar Pruebas.

Caso de uso "Insertar No Conformidades"

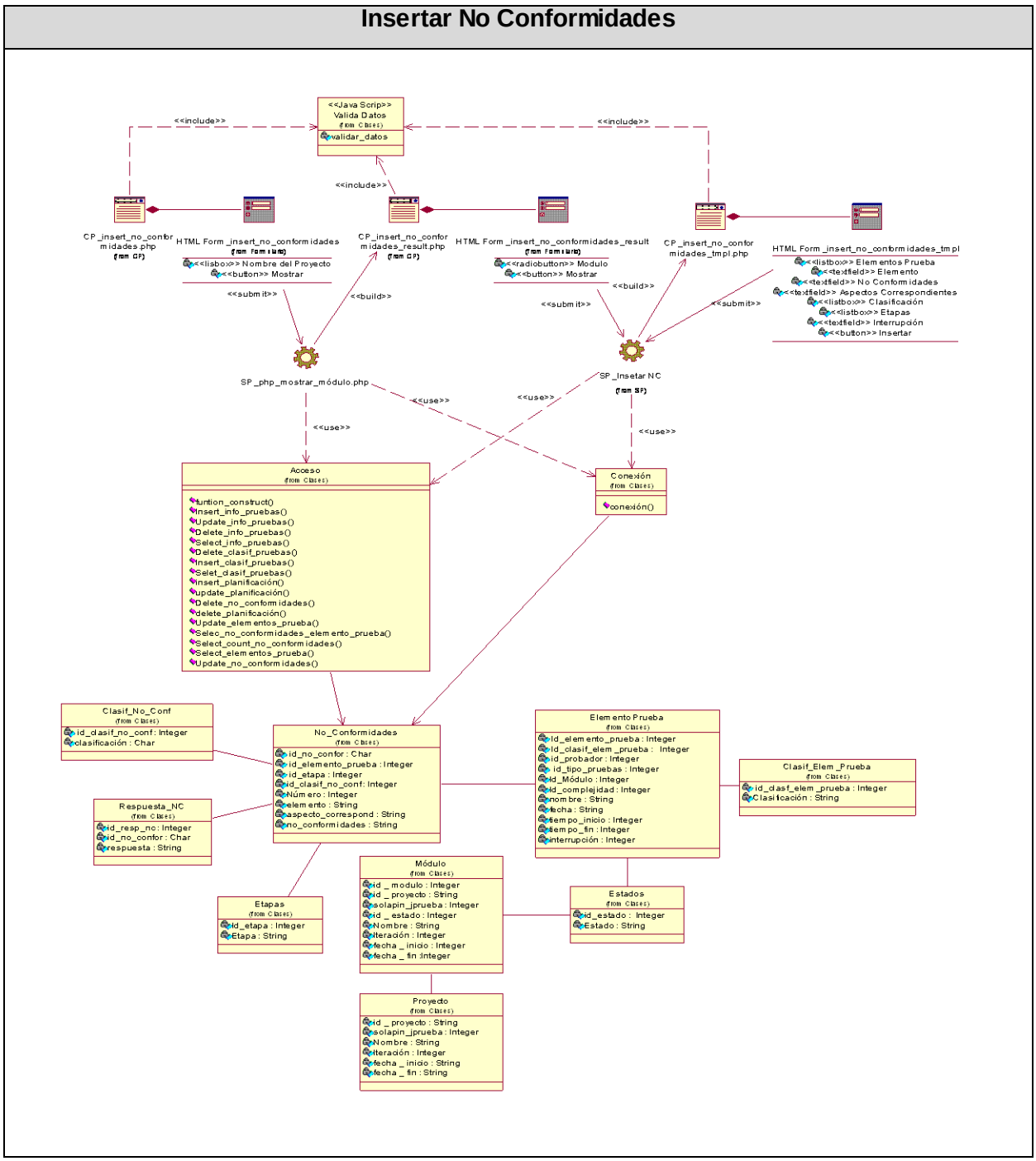


Figura 19: Diagrama de clases del diseño Caso de Uso Insertar No Conformidades.

Caso de uso "Gestionar Datos de Jefe de Prueba"

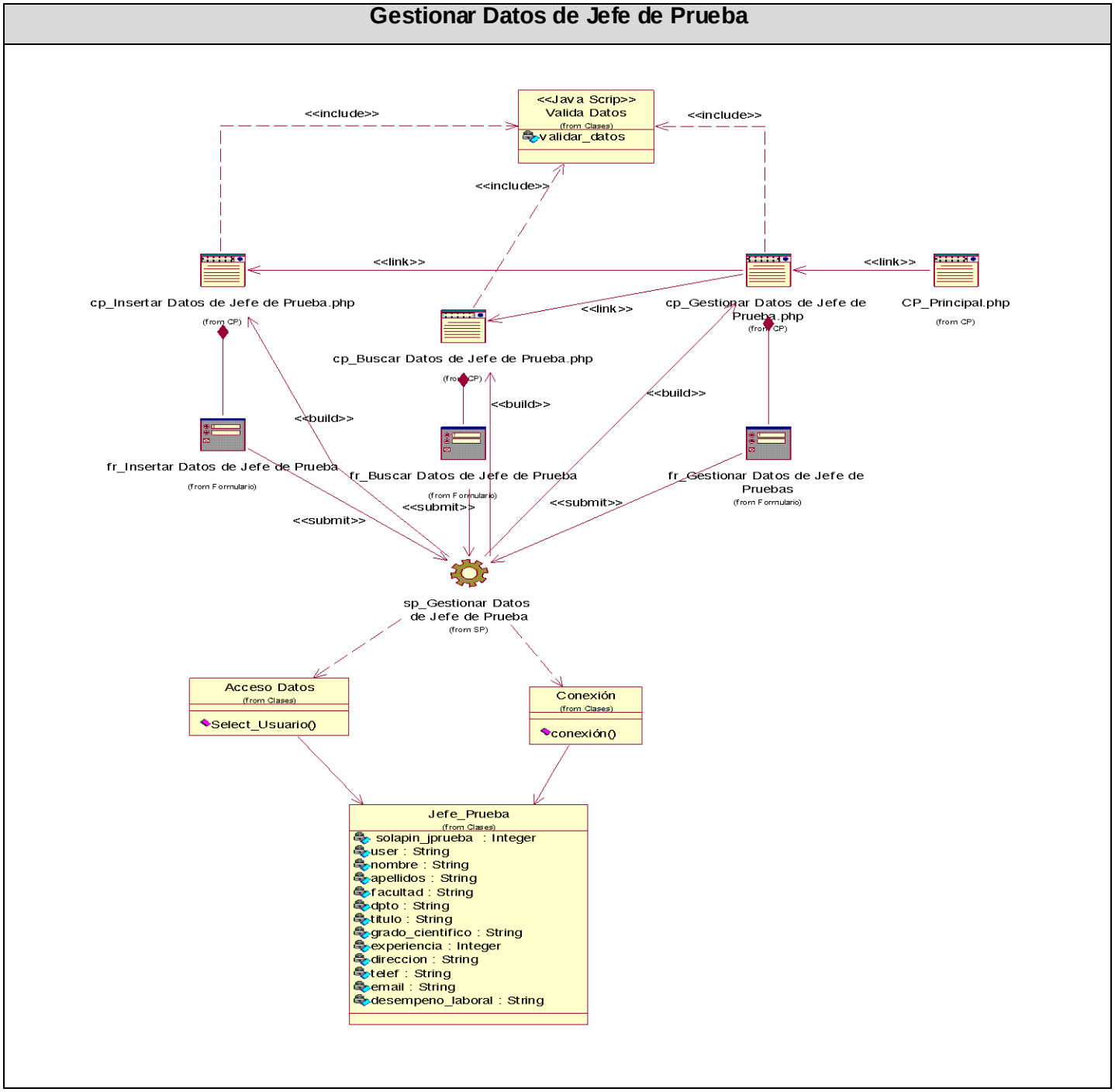


Figura 20: Diagrama de clases del diseño Caso de Uso Gestionar Datos de Jefe de Prueba.

Caso de uso "Seguimiento PC"

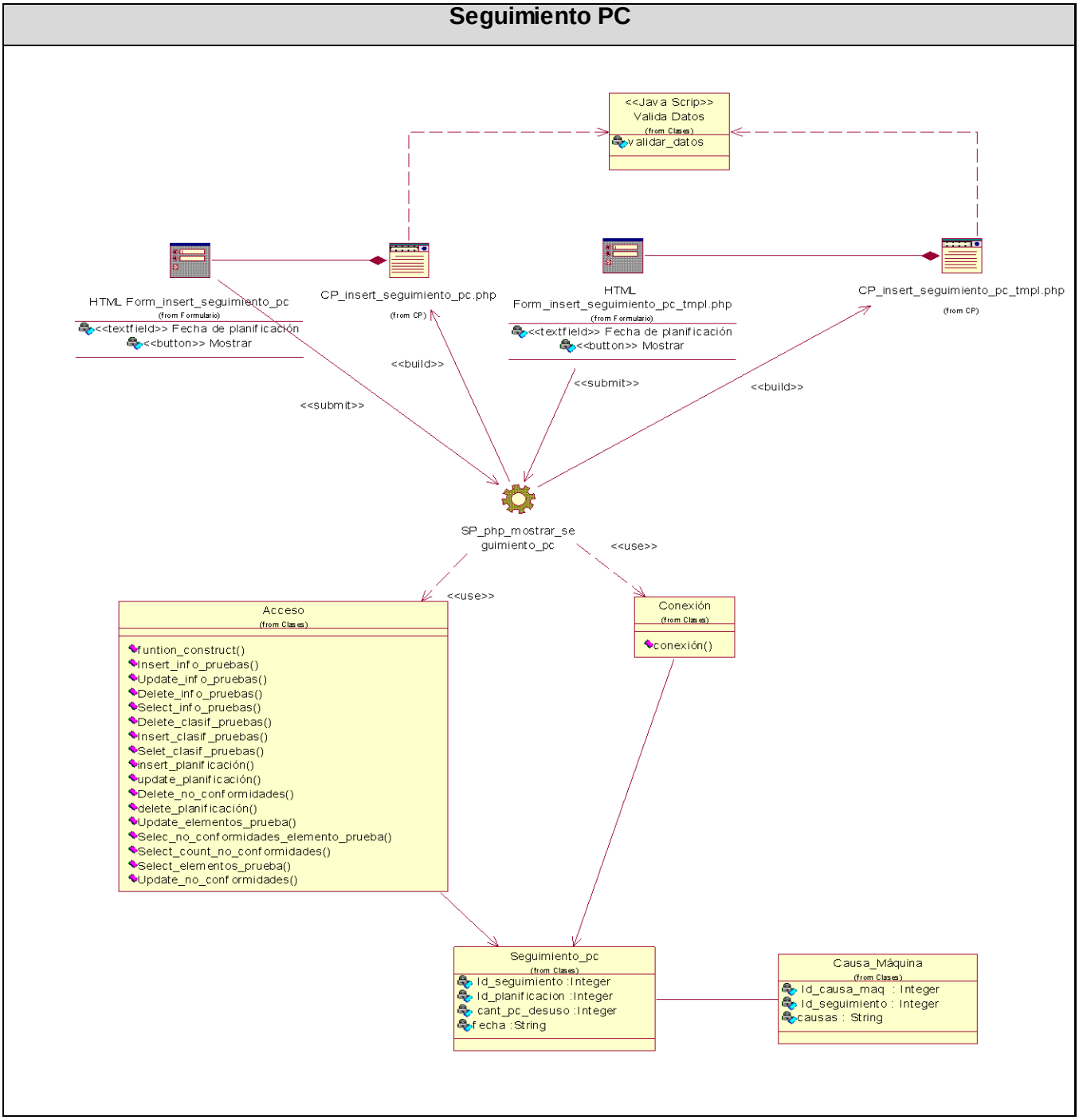


Figura 21: Diagrama de clases del diseño Caso de Uso Seguimiento PC.

2.4 Diseño de la Base de Datos.

2.4.1 Diagrama Entidad Relación de la BD.

Diagrama Entidad Relación

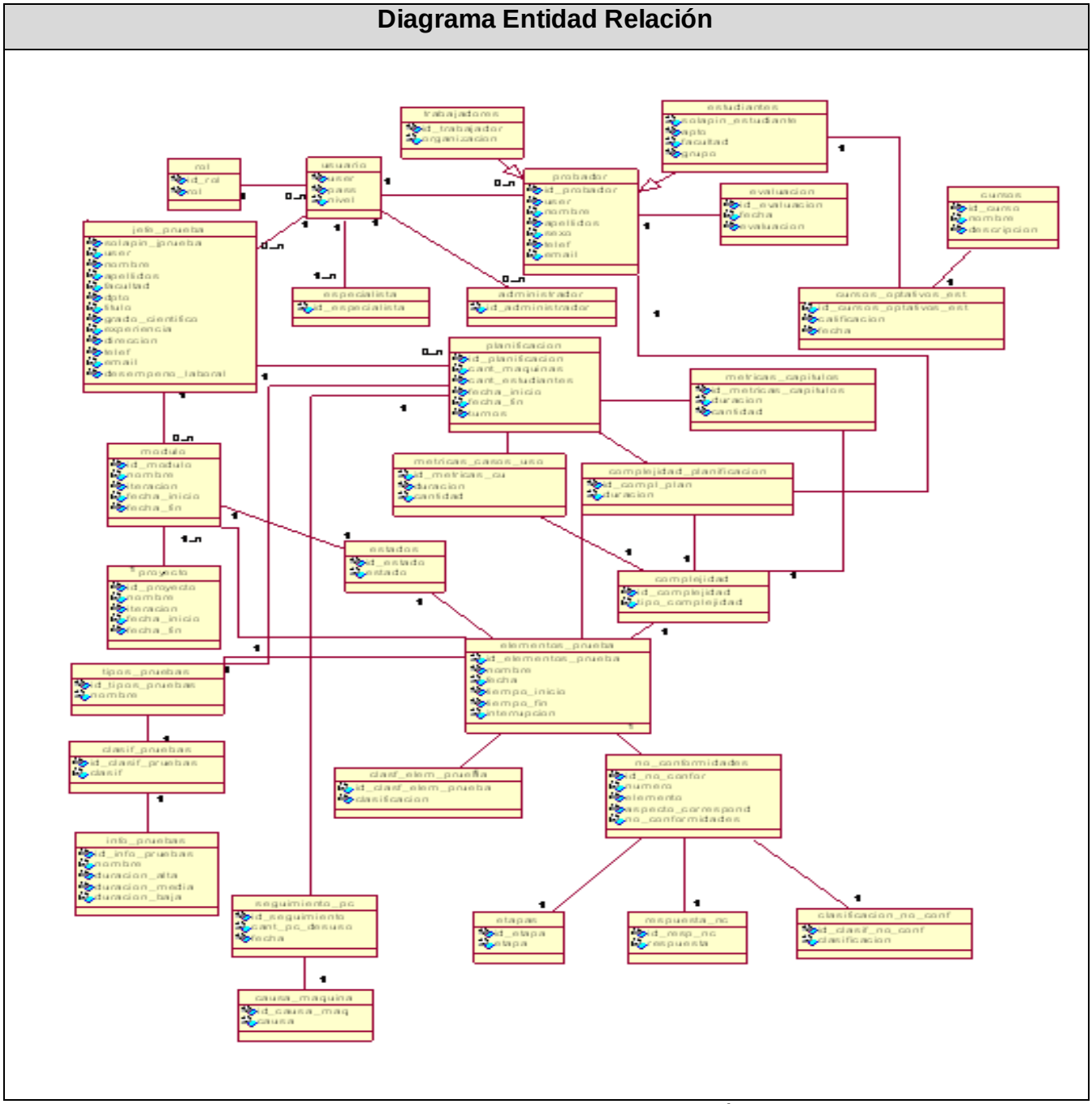


Figura 22: Diagrama Entidad Relación.

2.4.2 Descripción de las Tablas.

Nombre: Info_Pruebas		
Descripción: Esta tabla almacena las duraciones por complejidad de los tipos de pruebas.		
Atributo	**Tipo**	**Descripción**
Id_info_pruebas	Integer	Este campo es llave primaria y almacena el identificador que distingue la información de la duración de las pruebas
id_clasif_pruebas	Integer	Este campo es una llave extranjera y almacena el identificador que distingue la clasificación de la pruebas.
Nombre	String	Este campo almacena los nombres de las pruebas.
duración _ alta	Integer	Este campo almacena la duración para complejidad alta de una prueba.
duración _ media	Integer	Este campo almacena la duración para complejidad media de una prueba.
duración _ baja	Integer	Este campo almacena la duración para complejidad baja de una prueba.

Tabla 1: Descripción de la tabla de la base de dato "Info_Pruebas"

Nombre: clasif_pruebas		
Descripción: En esta tabla se almacenan las clasificaciones de pruebas.		
Atributo	**Tipo**	**Descripción**
id_clasif_pruebas	Integer	Este campo es una llave primaria y almacena el identificador que distingue la clasificación de la pruebas.
Clasif	String	Este campo almacena los nombres de las clasificaciones de pruebas.

Tabla 2: Descripción de la tabla de la base de dato "clasif_pruebas"

Nombre: tipos _ pruebas		
Descripción: En esta tabla se almacenan todos los tipos de prueba.		
Atributo	**Tipo**	**Descripción**
id_tipos_pruebas	Integer	Este campo es llave primaria y almacena el identificador que distingue a los tipos de prueba.
id_clasif_pruebas	Integer	Este campo es llave extranjera y almacena el identificador que distingue la clasificación de la pruebas.
Nombre	String	Este campo almacena los nombres de los tipos de pruebas.

Tabla 3: Descripción de la tabla de la base de dato “tipos_pruebas”

Nombre: Proyecto		
Descripción: En esta tabla se almacena los datos del proyecto.		
Atributo	**Tipo**	**Descripción**
id _ proyecto	String	Este campo es llave primaria y almacena el identificador que distingue a los proyectos.
solapin_jprueba	Integer	Este campo es llave extranjera y almacena el número de solapín del jefe de prueba.
Nombre	String	Este campo almacena los nombres de los proyectos.
Iteración	Integer	Este campo almacena las iteraciones de pruebas de un proyecto.
fecha _ inicio	String	Este campo almacena la fecha en que el proyecto entra en revisión.
fecha _ fin	String	Este campo almacena la fecha en que el proyecto termino de ser revisado.

Tabla 4: Descripción de la tabla de la base de dato “Proyecto”

Nombre: módulo		
Descripción: En esta tabla se almacenan los datos de los módulos de un proyecto.		
Atributo	**Tipo**	**Descripción**
id _ módulo	Integer	Este campo es llave primaria y almacena el identificador que distingue a los módulos.
id _ proyecto	String	Este campo es llave extranjera y almacena el identificador que distingue a los proyectos.
solapin_jprueba	Integer	Este campo es llave extranjera y almacena el número de solapín del jefe de prueba.
id _ estado	Integer	Este campo es llave extranjera y almacena el identificador que distingue el estado en el que se encuentra un módulo.
Nombre	String	Este campo almacena los nombres de los módulos.
Iteración	Integer	Este campo almacena las iteraciones de pruebas de un módulo.
fecha _ inicio	String	Este campo almacena la fecha en que el módulo entra en revisión.
fecha _ fin	String	Este campo almacena la fecha en que el módulo termino de ser revisado.

Tabla 5: Descripción de la tabla de la base de dato “módulo”

Nombre: planificación		
Descripción: En esta tabla se almacena las planificaciones realizadas.		
Atributo	**Tipo**	**Descripción**
id _ planificación	Integer	Este campo es una llave primaria y almacena el identificador que distingue las planificaciones.
solapin_jprueba	Integer	Este campo es llave extranjera y almacena el número de solapín del jefe de prueba.
id_tipos_pruebas	Integer	Este campo es llave extranjera y almacena el identificador que distingue a los tipos de prueba.
id _ módulo	Integer	Este campo es llave extranjera y almacena el identificador que distingue a los módulos.
cant_maquinas	Integer	Este campo almacena la cantidad de máquinas que intervienen en una planificación.
cant_estudiantes	Integer	Este campo almacena la cantidad de estudiantes que intervienen en una planificación.
fecha _ inicio	String	Este campo almacena la fecha en que debe comenzar el trabajo.
fecha _ fin	String	Este campo almacena la fecha en que debe terminar el trabajo.
Turnos	Integer	Este campo almacena los turnos de trabajo en una planificación.

Tabla 6: Descripción de la tabla de la base de dato “planificación”

Nombre: métricas _ capítulos		
Descripción: En esta tabla se almacenan las métricas de las pruebas que utilizan capítulos.		
Atributo	**Tipo**	**Descripción**
id_metricas_capitulos	Integer	Este campo es una llave primaria y almacena el identificador que distingue a las métricas de los capítulos.
id _ complejidad	Integer	Este campo es una llave extranjera y almacena el identificador que distingue a las complejidades de pruebas.
id _ planificación	Integer	Este campo es una llave extranjera y almacena el identificador que distingue las planificaciones.
Duración	Integer	Este campo almacena la duración de las métricas por capitulo.
Cantidad	Integer	Este campo me almacena la cantidad de capítulos que utilizaran esa métrica.

Tabla 7: Descripción de la tabla de la base de dato “métricas_capítulos”

Nombre: metricas_casos_uso		
Descripción: En esta tabla se almacenan las métricas de las pruebas que utilizan casos de uso.		
Atributo	**Tipo**	**Descripción**
id_metricas_cu	Integer	Este campo es una llave primaria y almacena el identificador que distingue a las métricas de los casos de uso.
id _ complejidad	Integer	Este campo es una llave extranjera y almacena el identificador que distingue a las complejidades de pruebas.
id _ planificación	Integer	Este campo es una llave extranjera y almacena el identificador que distingue las planificaciones.
Duración	Integer	Este campo almacena la duración de las métricas por casos de uso.
Cantidad	Integer	Este campo me almacena la cantidad de capítulos que utilizaran esa métrica.

Tabla 8: Descripción de la tabla de la base de dato “metricas_casos_uso”

Nombre: complejidad _ planificación		
Descripción: En esta tabla se almacenan la complejidad de las pruebas que utilizan en la planificación.		
Atributo	**Tipo**	**Descripción**
id_compl_plan	Integer	Este campo es una llave primaria y almacena el identificador que distingue a las métricas de la complejidad de prueba.
id _ complejidad	Integer	Este campo es una llave extranjera y almacena el identificador que distingue a las complejidades de pruebas.
id _ planificación	Integer	Este campo es una llave extranjera y almacena el identificador que distingue las planificaciones.
Duración	Integer	Este campo almacena la duración de las métricas por complejidad de pruebas.

Tabla 9: Descripción de la tabla de la base de dato “complejidad_planificación”

Nombre: Complejidad		
Descripción: En esta tabla se almacena las complejidades.		
Atributo	**Tipo**	**Descripción**
id _ complejidad	Integer	Este campo es una llave primaria y almacena el identificador que distingue a las complejidades de pruebas.
tipo _ complejidad	String	Este campo almacena los nombres de los tipos de complejidad.

Tabla 10: Descripción de la tabla de la base de dato “Complejidad”

Nombre: Usuario		
Descripción: Se almacena los datos de los usuarios del sistema.		
Atributo	**Tipo**	**Descripción**
User	String	Es la llave primaria y almacena el nombre de usuario relacionado con los probadores y jefes de pruebas.
Id _ roles	Integer	Es una llave extranjera y almacena el identificador del rol que desempeña el usuario.
Pass	String	Almacena la contraseña que tiene el usuario.
Nivel	Integer	Almacena

Tabla 11: Descripción de la tabla de la base de dato “usuario”

Nombre: roles		
Descripción: Se almacena el rol que desempeñan los usuarios del sistema.		
Atributo	**Tipo**	**Descripción**
Id_rol	Integer	Es la llave primaria y almacena el identificador de un rol que puede desempeñar un usuario.
rol	String	Almacena un rol que puede desempeñar un usuario del sistema.

Tabla 12: Descripción de la tabla de la base de dato “roles”

Nombre: probador		
Descripción: En esta tabla se almacenan los datos que coinciden de los estudiantes y los trabajadores, los probadores del sistema.		
Atributo	**Tipo**	**Descripción**
Id_probador	Integer	Este campo es una llave primaria y almacena el identificador que distingue al Probador.
user	String	Este campo es una llave extranjera y almacena el nombre de usuario relacionado con los probadores.
nombre	String	Este campo almacena el nombre del Probador.
apellidos	String	Este campo almacena los apellidos del Probador.
sexo	String	Este campo almacena el sexo del Probador.
telef	String	Este campo almacena el teléfono del Probador.
email	String	Este campo almacena el correo electrónico del Probador.

Tabla 13: Descripción de la tabla de la base de dato “probador”

Nombre: estudiantes		
Descripción: En esta tabla se almacenan los datos específicos de los estudiantes.		
Atributo	**Tipo**	**Descripción**
solapin_estudiante	Integer	Este campo es una llave primaria que almacena el número del solapín del estudiante.
Id_probador	Integer	Este campo es una llave extranjera y almacena el identificador que distingue al Probador.
apto	String	Este campo almacena el número del apartamento UCI donde reside el estudiante.
facultad	String	Este campo almacena el número de la facultad a la que pertenece el estudiante.
grupo	String	Este campo almacena el grupo al que pertenece el estudiante.

Tabla 14: Descripción de la tabla de la base de dato “estudiantes”

Nombre: trabajadores		
Descripción: En esta tabla se almacenan los datos específicos de los trabajadores.		
Atributo	**Tipo**	**Descripción**
Id_trabajador	Integer	Este campo es una llave primaria y almacena el identificador de cada uno de los trabajadores.
Id_probador	Integer	Este campo es una llave extranjera y almacena el identificador que distingue al Probador.
organizacion	String	Este campo almacena el nombre de la organización a la que pertenece el trabajador.

Tabla 15: Descripción de la tabla de la base de dato “trabajadores”

Nombre: evaluación		
Descripción: En esta tabla se almacenan los datos de las evaluaciones realizadas a los probadores.		
Atributo	**Tipo**	**Descripción**
Id_evaluacion	Integer	Este campo es una llave primaria y almacena el identificador de la evaluación de cada uno de los estudiantes.
Id_probador	Integer	Este campo es una llave extranjera y almacena el identificador que distingue al Probador.
fecha	Date	Este campo almacena la fecha de la evaluación.
evaluacion	String	Este campo almacena el valor de la evaluación realizada al estudiante durante su turno de trabajo.

Tabla 16: Descripción de la tabla de la base de dato “evaluación”

Nombre: cursos_optativos_est		
Descripción: En esta tabla se almacenan los datos específicos de los cursos optativos recibidos por los estudiantes.		
Atributo	**Tipo**	**Descripción**
Id_curso_optativo_est	Integer	Este campo es una llave primaria y almacena el identificador del curso optativo asociado a un estudiante.
solapin_estudiante	Integer	Este campo es una llave extranjera y es donde se almacena el número del solapín del estudiante.
id_curso	Integer	Este campo es una llave extranjera y almacena el identificador del curso que fue recibido por cualquiera de las personas involucradas en el laboratorio, ya sea estudiante o trabajador.
calificacion	Integer	Este campo almacena la calificación obtenida de un estudiante en un curso optativo.
fecha	Date	Este campo almacena la fecha en la que un estudiante recibió un curso optativo.

Tabla 17: Descripción de la tabla de la base de dato “cursos_optativos_est”

Nombre: cursos		
Descripción: En esta tabla se almacenan los datos de todos los cursos optativos.		
Atributo	**Tipo**	**Descripción**
Id_curso	Integer	Este campo es una llave primaria y almacena el identificador del curso que fue recibido por cualquiera de las personas involucradas en el laboratorio, ya sea estudiante o trabajador.
nombre	String	Este campo es una llave primaria y almacena el nombre de los cursos optativos.
descripcion	String	Este campo almacena la descripción de los Cursos optativos.

Tabla 18: Descripción de la tabla de la base de dato “cursos”

Nombre: seguimeinto_pc		
Descripción: En esta tabla se almacenan los datos de las pc sin utilizar.		
Atributo	**Tipo**	**Descripción**
Id_seguimiento	Integer	Este campo es una llave primaria y almacena el identificador de seguimiento.
Id_planificación	Integer	Este campo es una llave extranjera y almacena el identificador de las planificaciones.
cant_pc_desuso	Integer	Este campo almacena el número de pc que no se usan.
fecha	String	Este campo almacena la fecha en que la pc no se utilizó.

Tabla 19: Descripción de la tabla de la base de dato “seguimiento_pc”

Nombre: causa_maquina		
Descripción: En esta tabla se almacenan las diferentes causas por las que las pc no se pueden utilizar.		
Atributo	**Tipo**	**Descripción**
Id_causa_maq	Integer	Este campo es una llave primaria y almacena el identificador de la causa por la que no se utiliza una pc.
Id_seguimiento	Integer	Este campo es una llave extranjera y almacena el identificador de seguimiento.
causas	String	Este campo almacena las diferentes causas por las que las pc no se pueden utilizar.

Tabla 20: Descripción de la tabla de la base de dato “causa_maquina”

Nombre: jefe_prueba		
Descripción: En esta tabla se almacenan los datos específicos de los Jefes de prueba.		
Atributo	**Tipo**	**Descripción**
solapin_jprueba	Integer	Este campo es una llave primaria y es donde se almacena el número del solapín del Jefe de prueba.
user	String	Este campo es una llave extranjera y almacena el nombre de usuario relacionado con el Jefe de prueba.
nombre	String	Este campo almacena el nombre del Jefe de Prueba.
apellidos	String	Este campo almacena los apellidos del Jefe de prueba.
facultad	String	Este campo almacena el número de la facultad a la que pertenece el Jefe de prueba.
dpto	String	Este campo almacena el departamento al que pertenece el Jefe de prueba.
titulo	String	Este campo almacena el título de graduación de cada Jefe de prueba.
grado_cientifico	String	Este campo almacena el grado científico de Cada Jefe de prueba.
experiencia	Integer	Este campo almacena los años de experiencia en el tema.
direccion	String	Este campo almacena la dirección del Jefe de prueba.
telef	String	Este campo almacena el teléfono del Jefe de Prueba.
email	String	Este campo almacena el correo electrónico del Jefe de prueba.
desempeno_laboral	String	Este campo almacena los cursos que ha Pasado cada Jefe de prueba.

Tabla 21: Descripción de la tabla de la base de dato “jefe_prueba”

Nombre: cursos_optativos_prof		
Descripción: En esta tabla se almacenan los datos de los cursos optativos de los jefes de pruebas.		
Atributo	**Tipo**	**Descripción**
Id_cursos_optativos_prof	Integer	Este campo es una llave primaria y almacena el identificador de los cursos recibidos por cada Jefe de prueba.
Solapin_jprueba	Integer	Este campo es una llave extranjera y almacena el número del solapín del Jefe de prueba.
Id_curso	Integer	Este campo es una llave extranjera y almacena el identificador de los cursos recibidos por cada Jefe de prueba.
Calificación	Integer	Este campo almacena la calificación obtenida de un Jefe de prueba en un curso optativo.
Fecha	Date	Este campo almacena la fecha en la que un Jefe de prueba recibió un curso optativo.

Tabla 22: Descripción de la tabla de la base de dato “cursos_optativos_prof”

Nombre: estados		
Descripción: Se almacena el estado en que se encuentra los elementos de prueba y el Módulo.		
Atributo	**Tipo**	**Descripción**
id_estado	Integer	Es la llave primaria y almacena el identificador de un estado.
Estado	String	Almacena el estado en que se encuentra un elemento de prueba y un Módulo.

Tabla 23: Descripción de la tabla de la base de dato “estados”

Nombre: elemento_prueba		
Descripción: Se almacenan los datos correspondientes a los elementos de prueba que se van a revisar.		
Atributo	**Tipo**	**Descripción**
Id_elemento_prueba	Integer	Es la llave primaria y almacena el identificador de un elemento de prueba.
Id_clasif_elem_prueba	Integer	Es una llave extranjera y almacena el identificador de la clasificación de un elemento de prueba.
id_probador	Integer	Es una llave extranjera y almacena el identificador que distingue al probador.
id_tipo_pruebas	Integer	Es una llave extranjera y almacena el identificador del tipo de prueba al que pertenece un elemento de prueba.
Id_Módulo	Integer	Es una llave extranjera y almacena el identificador del Módulo al que pertenece el elemento de prueba.
Id_complejidad	Integer	Es una llave extranjera y almacena el identificador de la complejidad que tiene el elemento de prueba.
nombre	String	Almacena el nombre que lleva el elemento de prueba.
fecha	String	Almacena la fecha en que se reviso el elemento de prueba.
tiempo_inicio	Integer	Almacena el tiempo en que se comienza a revisar el elemento de prueba.
tiempo_fin	Integer	Almacena el tiempo en que se termina de revisar el elemento de prueba.
interrupción	Integer	Almacena el tiempo de interrupción que puede tener un probador al estar revisando el elemento de prueba.

Tabla 24: Descripción de la tabla de la base de dato “elemento_prueba”

Nombre: No_conformidades		
Descripción: Se almacenan los datos correspondientes a las no conformidades encontradas en las revisiones.		
Atributo	**Tipo**	**Descripción**
id_no_confor	Char	Es la llave primaria y almacena el identificador de una no conformidad.
id_elemento_prueba	Integer	Es una llave extranjera y almacena el identificador de un elemento de prueba.
id_etapa	Integer	Es una llave extranjera y almacena el identificador de la etapa en que se encuentra la revisión.
id_clasif_no_conf	Integer	Es una llave extranjera y almacena el identificador de la clasificación de la no conformidad.
Número	Integer	Almacena el Número consecutivo de las no conformidades encontradas
elemento	String	Almacena el elemento de la no conformidad encontrada.
aspecto_correspond	String	Almacena el donde fue que se encontró la no conformidad.
no_conformidades	String	Almacena la no conformidad encontrada.

Tabla 25: Descripción de la tabla de la base de dato “No_conformidades”

Nombre: clasif_elem_prueba		
Descripción: Se almacena la clasificación de las no conformidades.		
Atributo	**Tipo**	**Descripción**
id_clasf_elem_prueba	Integer	Es la llave primaria y almacena el identificador de la clasificación de los elementos de prueba.
Clasificación	String	Almacena la clasificación de un elemento de prueba.

Tabla 26: Descripción de la tabla de la base de dato “clasif_elem_prueba”

Nombre: clasificación_no_conf		
Descripción: Se almacena la clasificación de las no conformidades.		
Atributo	**Tipo**	**Descripción**
id_clasif_no_conf	Integer	Es la llave primaria y almacena el identificador de la clasificación de las no conformidades.
clasificación	Char	Almacena la clasificación de una no conformidad.

Tabla 27: Descripción de la tabla de la base de dato “clasificación_no_conf”

Nombre: respuesta_NC		
Descripción: Se almacena las respuestas de las no conformidades encontradas.		
Atributo	**Tipo**	**Descripción**
id_resp_nc	Integer	Es la llave primaria y almacena el identificador de una respuesta a una no conformidad.
id_no_confor	Char	Es una llave extranjera y almacena el identificador de la no conformidad a la que le da respuesta.
respuesta	String	Almacena la respuesta de a las no conformidades.

Tabla 28: Descripción de la tabla de la base de dato “respuesta_NC”

Nombre: etapas		
Descripción: Se almacena la etapa en que se encontró la no conformidad.		
Atributo	**Tipo**	**Descripción**
Id_etapa	Integer	Es la llave primaria y almacena el identificador de la etapa en que se encuentra una no conformidad.
Etapa	String	Almacena la etapa en que se encuentra una no conformidad.

Tabla 29: Descripción de la tabla de la base de dato “etapas”

2.5 Definiciones de diseño que se aplican.

El diseño debe adaptarse al ambiente de trabajo del usuario por tanto el desarrollo de la aplicación se basa en los 7 Principios del Diseño Universal o Diseño para Todos, estos principios se centran en el diseño de aplicaciones teniendo en cuenta la cultura, el conocimiento, el ambiente, que influyan sobre los usuarios a los que va dirigido el producto y los mismos se presentan a continuación:

1. Igualdad de uso
2. Flexibilidad
3. Simple e intuitivo
4. Información fácil de percibir
5. Tolerante a errores
6. Escaso esfuerzo físico
7. Dimensiones apropiadas

El diseño debe ser fácil de usar y adecuado para todas las personas independientemente de sus capacidades y habilidades. Debe poder adecuarse a un amplio rango de preferencias y habilidades individuales, fácil de entender independientemente de la experiencia, los conocimientos, las habilidades o el nivel de concentración del usuario. El diseño debe ser capaz de intercambiar información con usuario, independientemente de las condiciones ambientales o las capacidades sensoriales del mismo. Debe minimizar las acciones accidentales o fortuitas que puedan tener consecuencias fatales o no deseadas. El diseño debe poder ser usado eficazmente y con el mínimo

esfuerzo posible. Los tamaños y espacios deben ser apropiados para el alcance, manipulación y uso por parte del usuario, independientemente de su tamaño, posición, y movilidad.

2.6 Interfaz Gráfica.

Para que la aplicación sea un éxito absoluto no debe solamente cumplir con todos los requisitos funcionales acordados sino también con muchas otras cosas, por ejemplo algo muy importante, que no se puede dejar de la mano es que la interfaz de la aplicación debe ser amigable, sencilla de fácil entendimiento y manejo para el usuario. Debe además reflejar equilibrio en la organización de la información, mostrando todas las páginas la información en el mismo orden. Se debe optimizar la cantidad de elementos en la pantalla, ayudando a una mejor comprensión de la información mostrada en pantalla. Debe evidenciar que cada elemento de la pantalla siga el mismo patrón de tamaño, color y forma.

Para dar cumplimiento a lo antes expuesto se utilizó el mismo estilo y tamaño de fuente en cada una las páginas para proporcionarle uniformidad al sistema y en caso de determinados contenidos que tienen formato Negrita, se realizó con el objetivo de resaltar o diferenciar una información de otra. Los colores son aquellos que identifican nuestra universidad, azul y blanco, ni fuertes ni brillantes, para que cumpla con las pautas trazadas en nuestro centro.

2.7 Tratamiento de Errores.

Algo fundamental para un buen funcionamiento del sistema es el tratamiento de errores, desde el inicio de desarrollo del mismo se realizan operaciones y se cumplen tareas para evitar la ocurrencia de estos.

Se contará con un sistema de tratamiento de errores para reducir la posibilidad de cometerlos. Para esto se contará con la validación de la información introducida en el sistema por la validación de los formularios utilizando funciones JavaScripts.

Mediante la interfaz Web se impedirá que el usuario asuma un papel activo en la introducción de la información, para esto se contará con cuadros de opción, menú de selección lo cual facilitará la entrada de datos. La información que requiera ser adicionada por el usuario se validará mediante funciones que garanticen que sea válida y que el cuadro de texto no esté vacío si es obligatorio

llenarlo. Si hay un error en la información le saldrá al usuario un mensaje en pantalla indicándole el error, al oprimir Aceptar el mensaje desaparece y el usuario podrá seguir introduciendo los datos en el formulario.

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

A continuación se muestran algunos de estos mensajes:

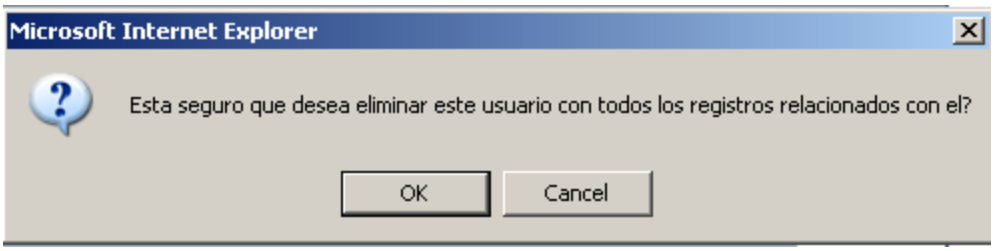


Figura 23: Mensaje de verificación. Eliminar Usuario.

Figura 24: Mensaje de validación. Insertar Usuario.

Figura 25: Mensaje de validación. Campo Teléfono del Usuario Jefe de Prueba.

2.8 Seguridad

La seguridad en el sitio está implementada a través del servidor de base de datos MySQL y el uso de variables de sesión para restringir el acceso de los usuarios a determinadas páginas.

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

Figura 26: Mensaje de verificación. Campo Contraseñas.

Figura 27: Interfaz de autenticación.

2.9 Concepción de la ayuda.

Para auxiliar a los usuarios que interactúen con el sistema se confeccionó un Manual de Usuario en formato Microsoft Word donde se incluirá una explicación detallada de cómo acceder y trabajar con el sistema, las restricciones que tienen como usuario y cada una de las funcionalidades que tiene la

aplicación. Se mantendrá también un continuo contacto con los realizadores del sistema para cuando surjan futuros problemas los mismos sean erradicados con la mayor rapidez.

2.10 Conclusiones

En este capítulo se efectúo todo el análisis y el diseño del sistema donde se dejó todo listo para el siguiente flujo de implementación. Se diseñó una base de datos donde se garantiza el cumplimiento de todos los requisitos funcionales, además de abordar en temas importantes como el de la seguridad y el tratamiento de los errores de la aplicación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el presente capítulo se construye el modelo necesario para desarrollar el proceso de implementación del sistema con los diagramas de componentes definidos. También se elabora el diagrama de despliegue donde se representan los nodos necesarios, en los que se distribuye la aplicación. Además se abordarán los temas relacionados con las pruebas aplicadas al software.

3.2 *Implementación*

3.2.1 Modelo de Implementación.

El modelo de implementación representa como los elementos del modelo de diseño y las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, entre otros.

Los diagramas de despliegue y componentes entre ellos conforman lo que se conoce como un modelo de implementación al describir los componentes, construir su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

Componente

Un componente es una parte física y reemplazable de un sistema que está compuesto por un conjunto de interfaces y proporciona la realización de dicho conjunto. Representa una unidad de código (fuente, binario o ejecutable) que permite mostrar las dependencias en tiempo de compilación y ejecución. Las instancias de componentes de software muestran unidades de software en tiempo de ejecución y generalmente contribuyen a identificar sus dependencias y su localización en nodos. Pueden mostrar también que interfaces implementan y qué objetos contienen.

Diagrama de Componente

Los diagramas de componentes contribuyen a un mejor entendimiento del modelo de implementación. Con ellos se representan los componentes lógicos de la aplicación así como las relaciones de dependencia que existen entre ellos.

Caso de Uso "Autentificar"

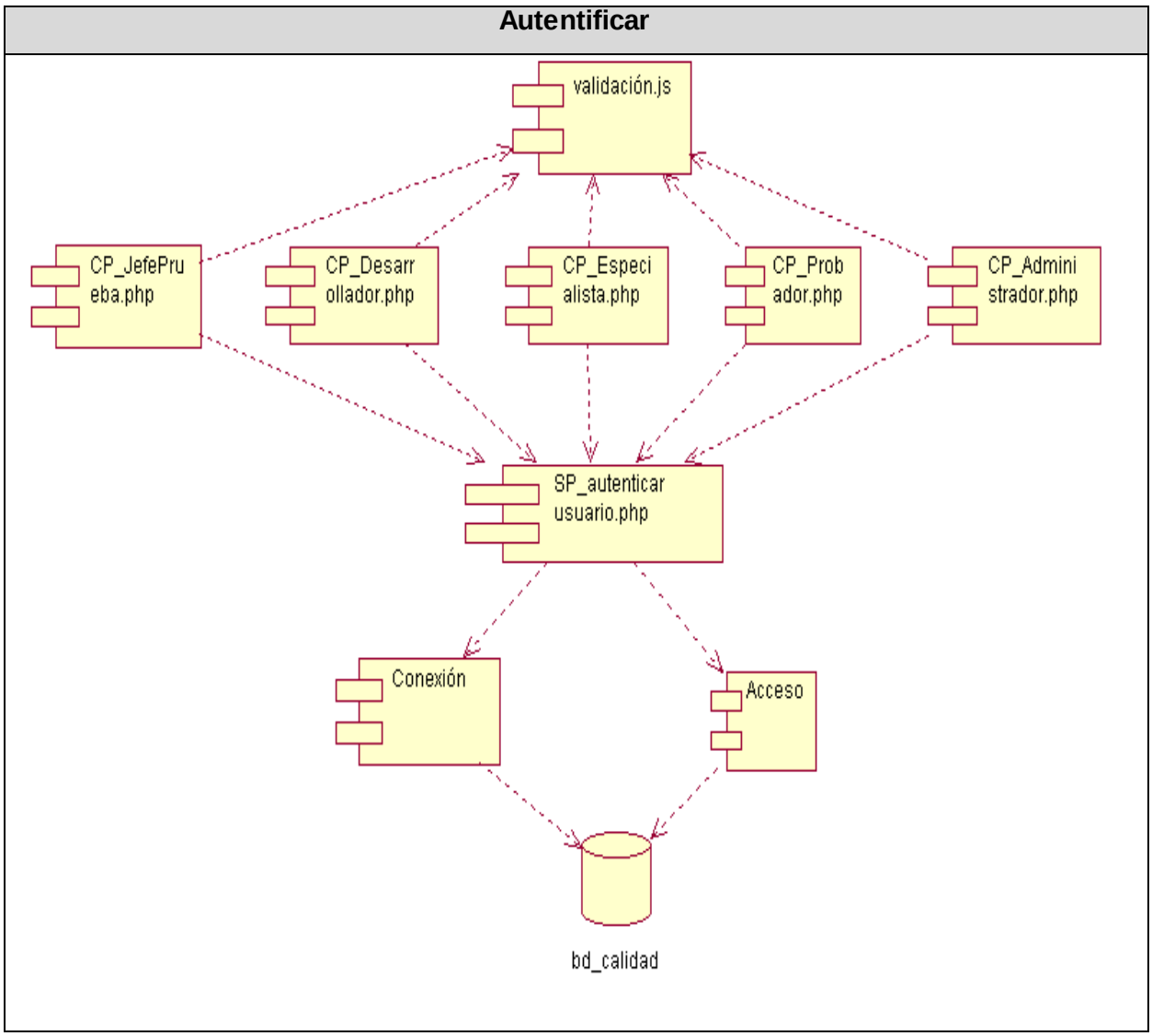


Figura 28: Diagrama de Componente. Caso de Uso "Autentificar"

Caso de Uso "Gestionar pruebas"

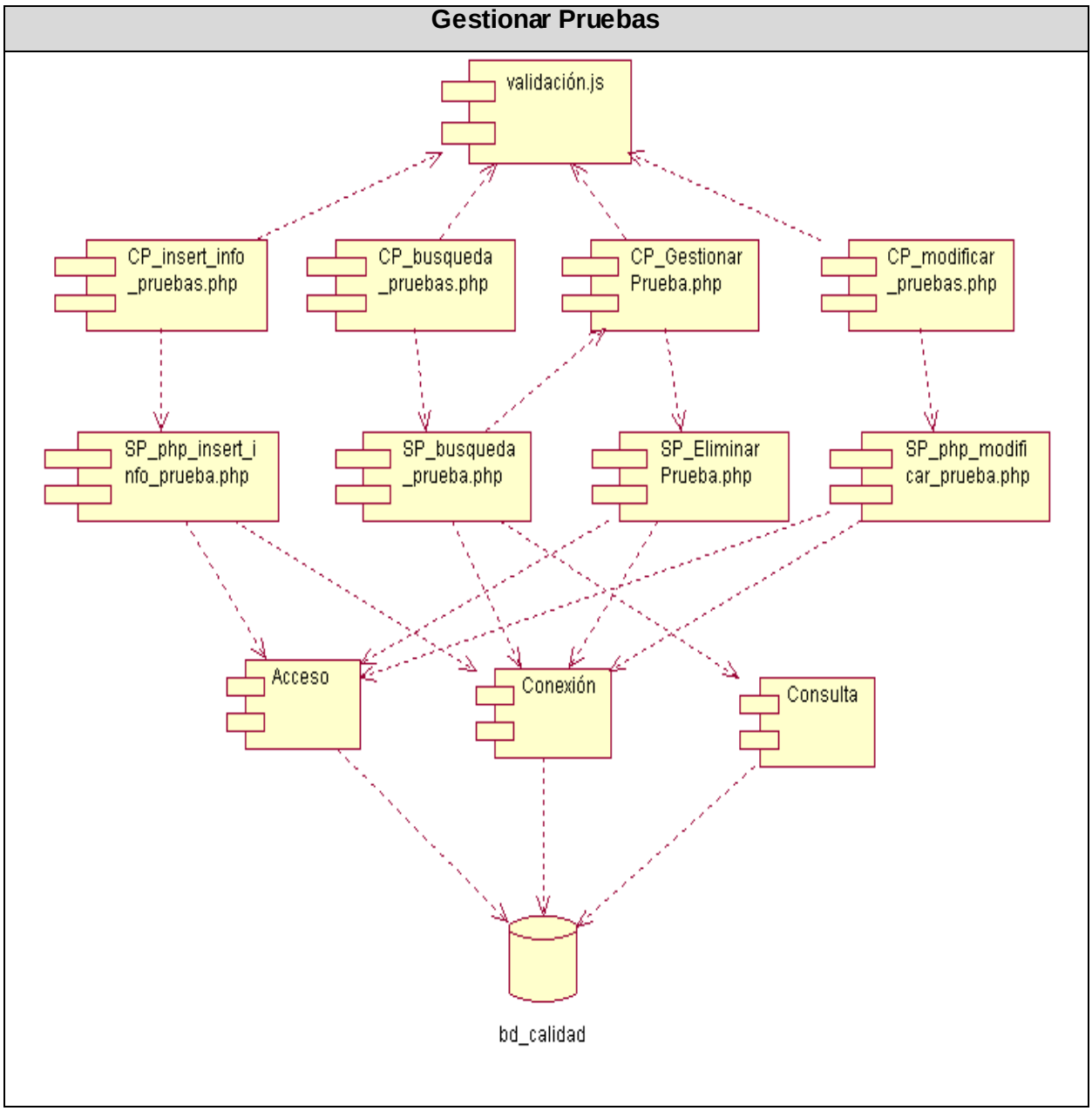


Figura 29: Diagrama de Componente. Caso de Uso "Gestionar pruebas"

Caso de uso "Planificar pruebas"

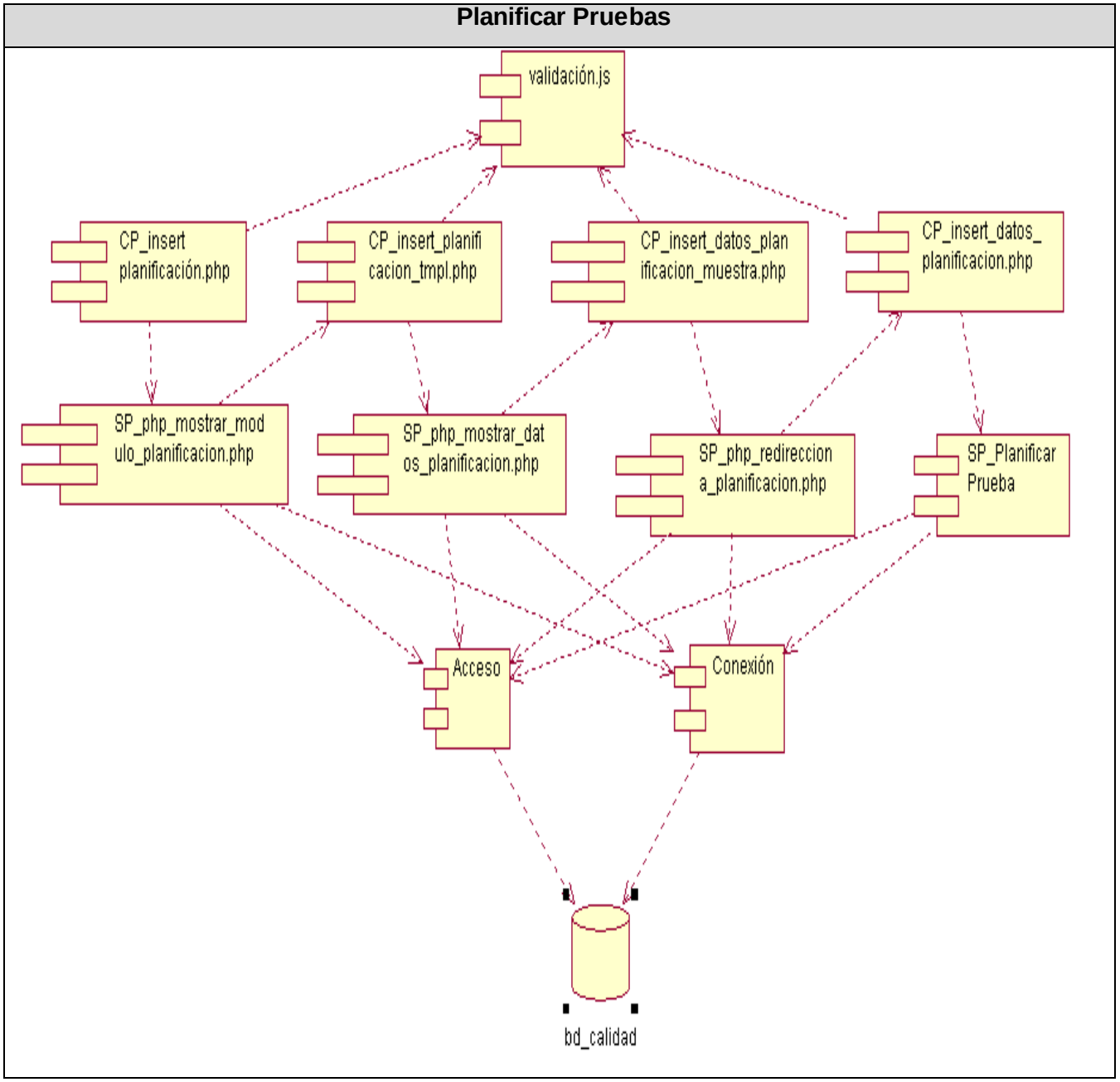


Figura 30: Diagrama de Componente. Caso de Uso "Planificar pruebas"

Caso de uso "Insertar No Conformidades"

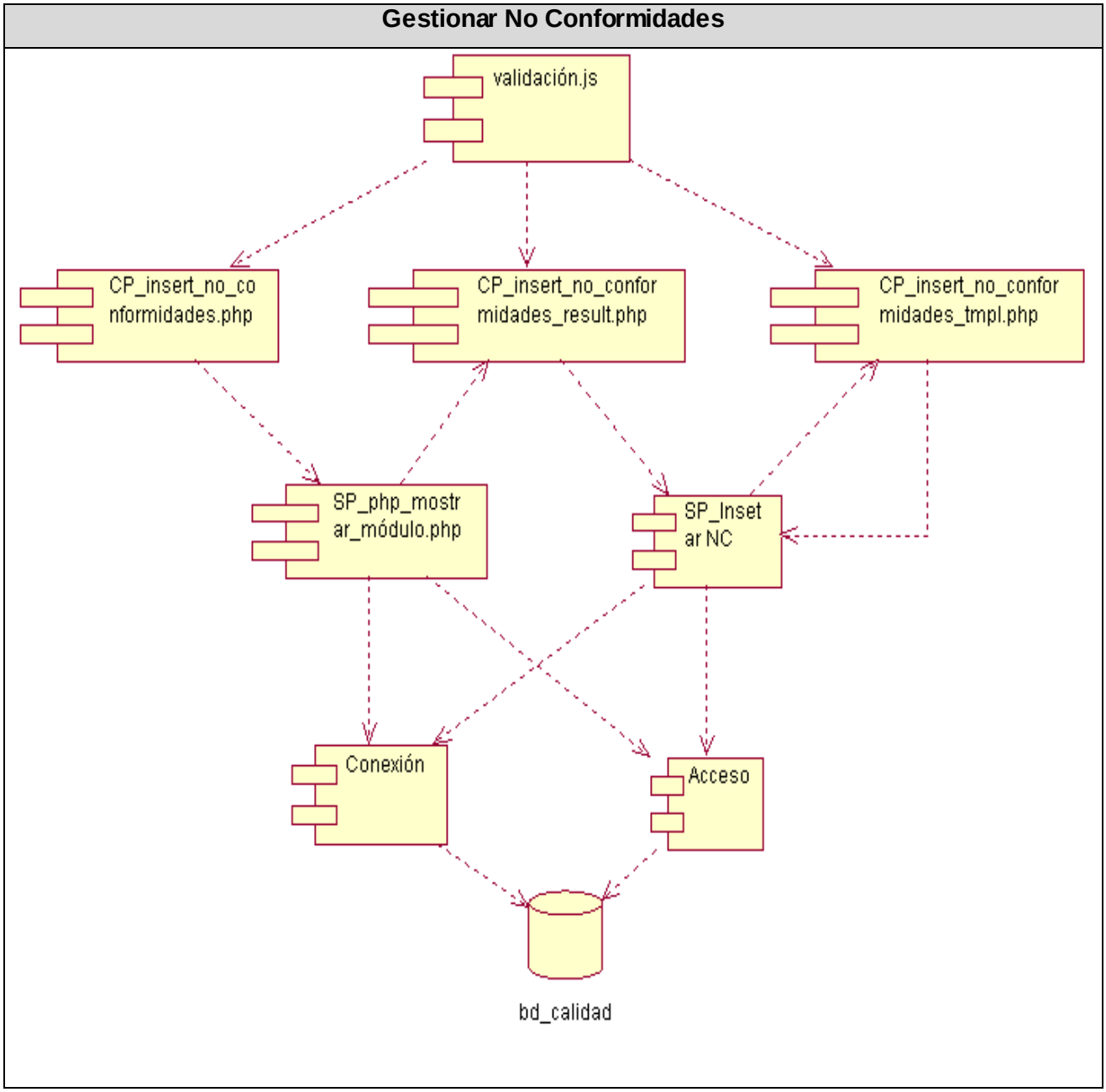


Figura 31: Diagrama de Componente. Caso de Uso "Insertar No Conformidades"

Caso de uso "Seguimiento PC"

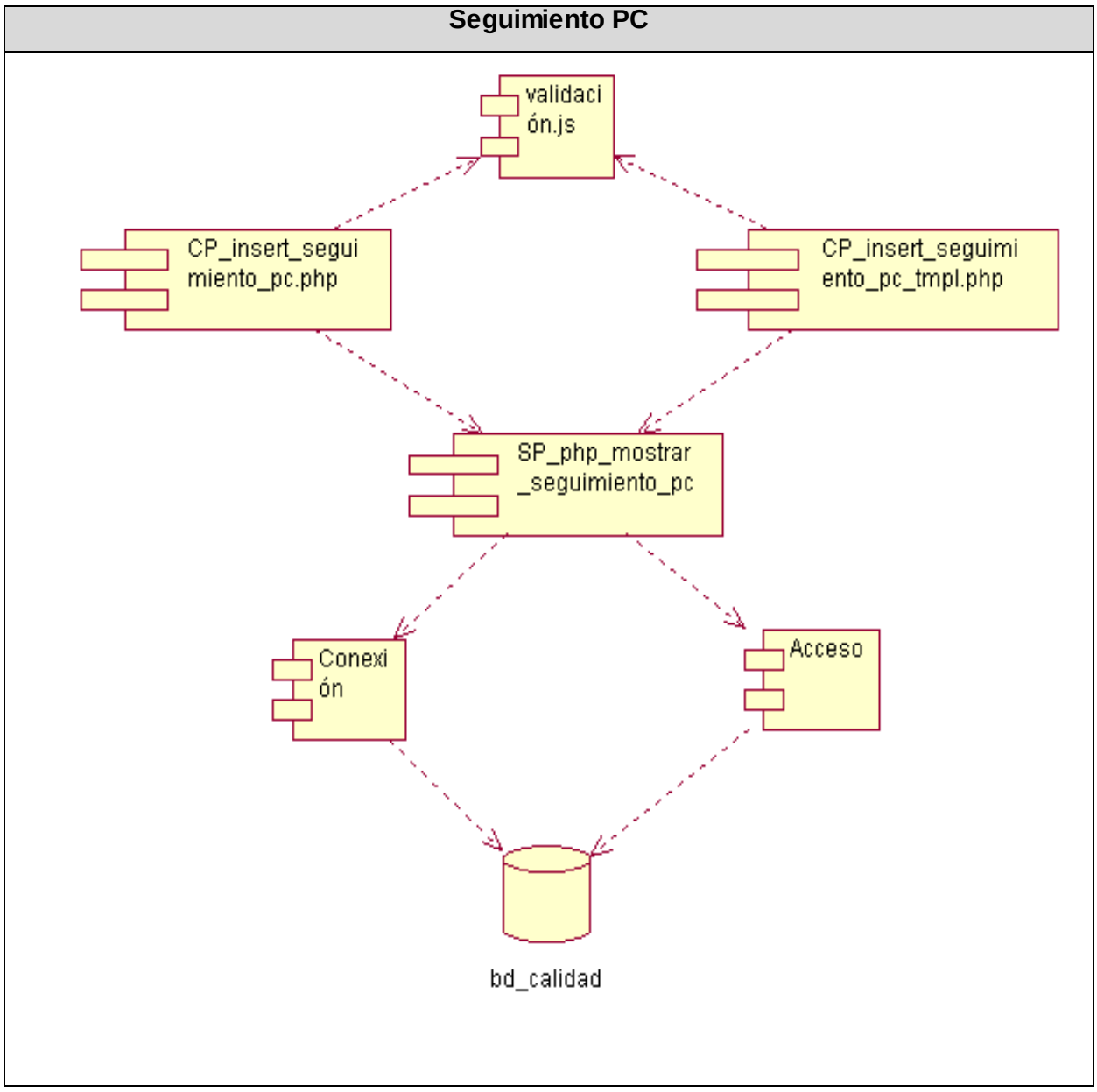


Figura 32: Diagrama de Componente. Caso de Uso "Seguimiento PC"

3.2.2 Modelo de despliegue.

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos.

El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. El modelo consiste en uno o mas nodos (elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos), dispositivos (nodos estereotipados con una capacidad de procesamiento en el nivel modelado de abstracción), y conectores, entre nodos, y entre nodos y dispositivos. El modelo de despliegue también mapea procesos dentro de estos elementos de procesamiento, permitiendo la distribución del comportamiento a través de los nodos que son representados.

El Modelo de Despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos [10].

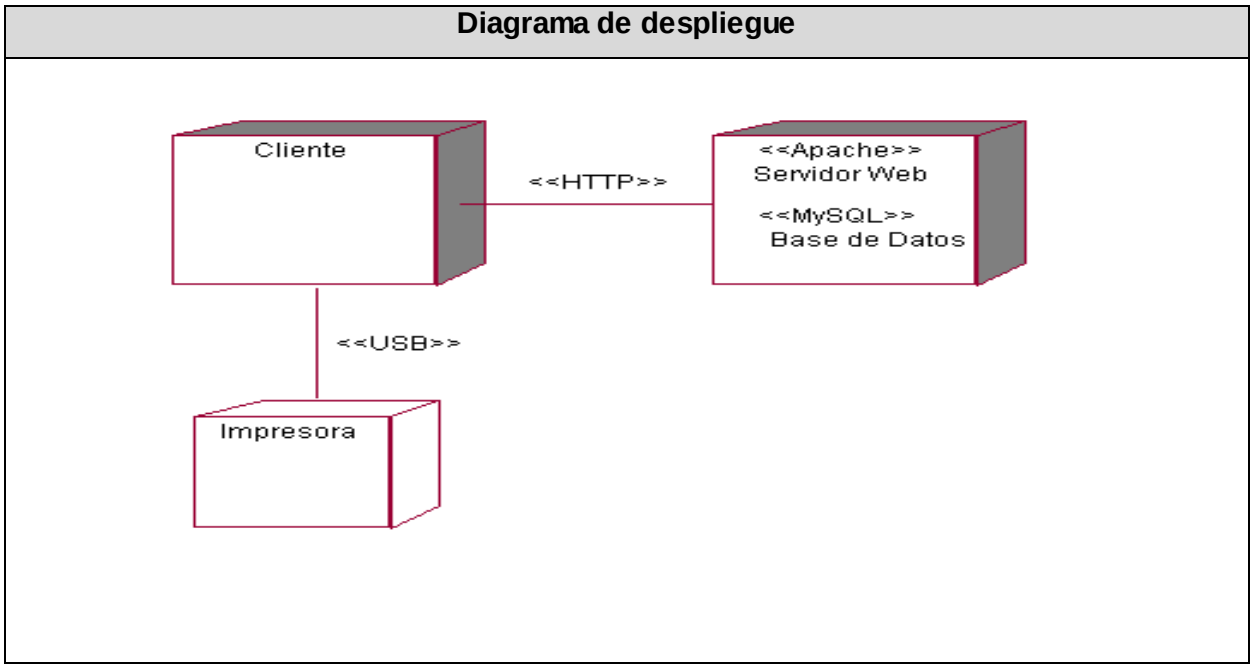


Figura 33: Diagrama de despliegue.

3.3 Prueba.

Un instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos.

3.3.1 Casos de Prueba.

Un Caso de Prueba es el conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular como, por ejemplo; ejercitar el camino concreto de un programa o verificar el cumplimiento de un requisito determinado. En la UCI son muy comunes en este entorno las pruebas de Caja Negra.

3.3.2 Prueba de Caja Negra.

Una Prueba de Caja Negra verifica el comportamiento observable externamente del sistema, o sea, se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Especifica como probar un caso de uso o un escenario específico de un caso de uso.

3.3.3 Modelo de Caso de Prueba.

Caso de Uso:	**Planificar Prueba**	
Caso de Prueba:	**Planificar Prueba**	
Entrada	Resultados	Condiciones
❖ Se escoge el Jefe de prueba. ❖ Se escoge el proyecto.	❖ Se muestra una interfaz con el nombre y apellido del jefe de prueba seleccionado y muestra el listado de los módulos del proyecto.	❖ El especialista de calidad o el administrador debió haber insertado con anterioridad el jefe de prueba. ❖ Se debió haber insertado con anterioridad el proyecto y el jefe de prueba que le realizará las pruebas a ese proyecto. ❖ El jefe de prueba seleccionado tiene que ser el mismo que tiene asignado el proyecto.
❖ Se selecciona la prueba a planificar.	❖ Aparece una interfaz donde se insertan los recursos a planificar.	❖ Se debió haber insertado con anterioridad la prueba a planificar.
❖ Se introducen los recursos que se tienen para la ejecución de la prueba.	❖ Aparece una interfaz donde se muestra la planificación y estimación de los recursos insertados.	

Tabla 30: Caso de Prueba para el Caso de Uso Planificar Prueba.

Caso de Uso:	**Gestionar Prueba**	
Caso de Prueba:	**Insertar Prueba**	
Entrada	Resultados	Condiciones
❖ Se selecciona insertar una prueba.	❖ El sistema inserta nuevas pruebas.	❖ Se debió haber insertado con anterioridad: ❖ Nombre de la prueba ❖ Clasificación ❖ Métricas asociadas a la prueba.

Tabla 31: Caso de Prueba para el Caso de Uso Gestionar Prueba (Sección Insertar Prueba).

Caso de Uso:	**Gestionar Proyecto**	
Caso de Prueba:	**Insertar Proyecto**	
Entrada	Resultados	Condiciones
❖ Se selecciona insertar un proyecto.	❖ El sistema inserta nuevos proyectos y muestra la interfaz para insertar sus módulos.	❖ Se debe introducir en el módulo de planificación: ❖ Nombre del proyecto ❖ Iteración del proyecto. ❖ Fecha ❖ Jefe de Prueba.

Tabla 32: Caso de Prueba para el Caso de Uso Gestionar Proyecto (Sección Insertar Proyecto).

Caso de Uso:	**Insertar No Conformidades**	
Caso de Prueba:	**Insertar No Conformidades**	
Entrada	Resultados	Condiciones
❖ Seleccionar proyecto.	❖ Se muestra la lista de módulos correspondientes al proyecto.	❖ Se debe de haber insertado el proyecto en el módulo de planificación.
❖ Se selecciona el módulo al cual se le van a hacer las pruebas.	❖ Se muestra la interfaz para insertar las no conformidades del módulo probado.	❖ Se debe de haber insertado los elementos de prueba en el módulo de No Conformidades.

Tabla 33: Caso de Prueba para el Caso de Uso Insertar No Conformidades.

Caso de Uso:	**Insertar elemento de prueba**	
Caso de Prueba:	**Insertar elemento de prueba**	
Entrada	Resultados	Condiciones
❖ Se selecciona insertar elementos de prueba.	❖ Se muestra una interfaz para escoger el módulo al cual se le van a insertar los elementos de prueba.	❖ Se debe de haber insertado el proyecto en el módulo de planificación.
❖ Se selecciona el módulo al cual corresponde el elemento de prueba.	❖ Se muestra una interfaz para que se introduzcan los datos del elemento de prueba.	

Tabla 34: Caso de Prueba para el Caso de Uso Insertar elemento de prueba.

Caso de Uso:	**Evaluar Probador**	
Caso de Prueba:	**Evaluar Probador**	
Entrada	Resultados	Condiciones
❖ Se selecciona evaluar probador.	❖ Muestra la interfaz para evaluar el probador.	❖Se debe de haber insertado el probador en el módulo de Seguimiento y revisar el trabajo que hizo en el módulo No Conformidades.

Tabla 35: Caso de Prueba para el Caso de Uso Evaluar Probador.

3.4 Conclusiones

En este capítulo se elaboró el diagrama de componentes mostrándose la relación existente entre las clases de diseño descritas como componentes, también se confeccionó el diagrama de despliegue donde se mostró como se encontraban distribuidos los nodos físicos del sistema unidos por conexiones de comunicación. También se muestran los casos de prueba confeccionados para las pruebas efectuadas a la aplicación.

CONCLUSIONES

Una vez concluida la investigación y la herramienta que informatiza los procesos en el laboratorio de pruebas de liberación, se ha dado cumplimiento a los objetivos planteados y se obtuvieron los resultados que a continuación se mencionan:

- Se realizó un análisis de las herramientas y tecnologías utilizadas para llevar a cabo el proceso de implementación.
- Se logró la realización del flujo de trabajo de análisis y diseño.
- Se implementó la herramienta informática propuesta.
- Se probó el sistema para valorar la eficacia del mismo.
- La aplicación posibilita la emisión de reportes para mantener actualizados a los directivos de la universidad.

RECOMENDACIONES

- Continuar con el perfeccionamiento constante de la aplicación, para mejorar su rendimiento y rapidez de procesamiento de la información que se maneja.
- Mejorar el sistema aplicando nuevas tecnologías y explotarlas al máximo para evitar que la tecnología con la cual el sistema fue desarrollado se vuelva obsoleta, logrando un mejor funcionamiento.
- Preparar al personal que va a interactuar con el software para evitar errores, explicándole en el funcionamiento del mismo antes de ponerlo en práctica.
- Buscar métodos que ayuden al personal a elevar los conocimientos tanto en lograr una mejor seguridad de la aplicación como en su desarrollo y funcionamiento.

REFERENCIAS BIBLIOGRAFICAS

1. Echeverría, R.C.a.D., *Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad: Módulo Planificación.*, Universidad de las Ciencias Informáticas: Ciudad Habana.
2. Góngora, M.N.a.A., *Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad de Software: Módulo Gestión de las No Conformidades.* . 2007, Universidad de las Ciencias Informáticas: Ciudad Habana.
3. Martínez, Y.L.a.K., *Sistema automatizado para los procesos en el laboratorio de Calidad. Módulo:Seguimiento de los recursos.* . 2007, Universidad de las Ciencias Informáticas: Ciudad de la Habana.
4. *SIGNO - Sistema para Gestión de No Conformidades*. [cited 2008; Available from: http://www.profitek.com.co/productos/signo.htm#arriba.
5. *Introducción a la Ingeniería de Software.* 2007 - 2008.
6. *Netcraft*. 2007 [cited 2008 Available from: http://news.netcraft.com/archives/web_server_survey.html.
7. *Una Introducción a APACHE*. 2006 [cited 2008; Available from: http://linux.ciberaula.com/articulo/linux_apache_intro.
8. Alvarez, M.A. *WampServer*. [cited 2008; Available from: http://www.desarrolloweb.com/articulos/1598.php.
9. *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis.* 2007-2008.
10. *Algunos aspectos sobre Modelo de diseño.* , in *Conferencia Diseño*. 2007-2008.
11. *Material de apoyo Conferencia de Diseño*. 2008.

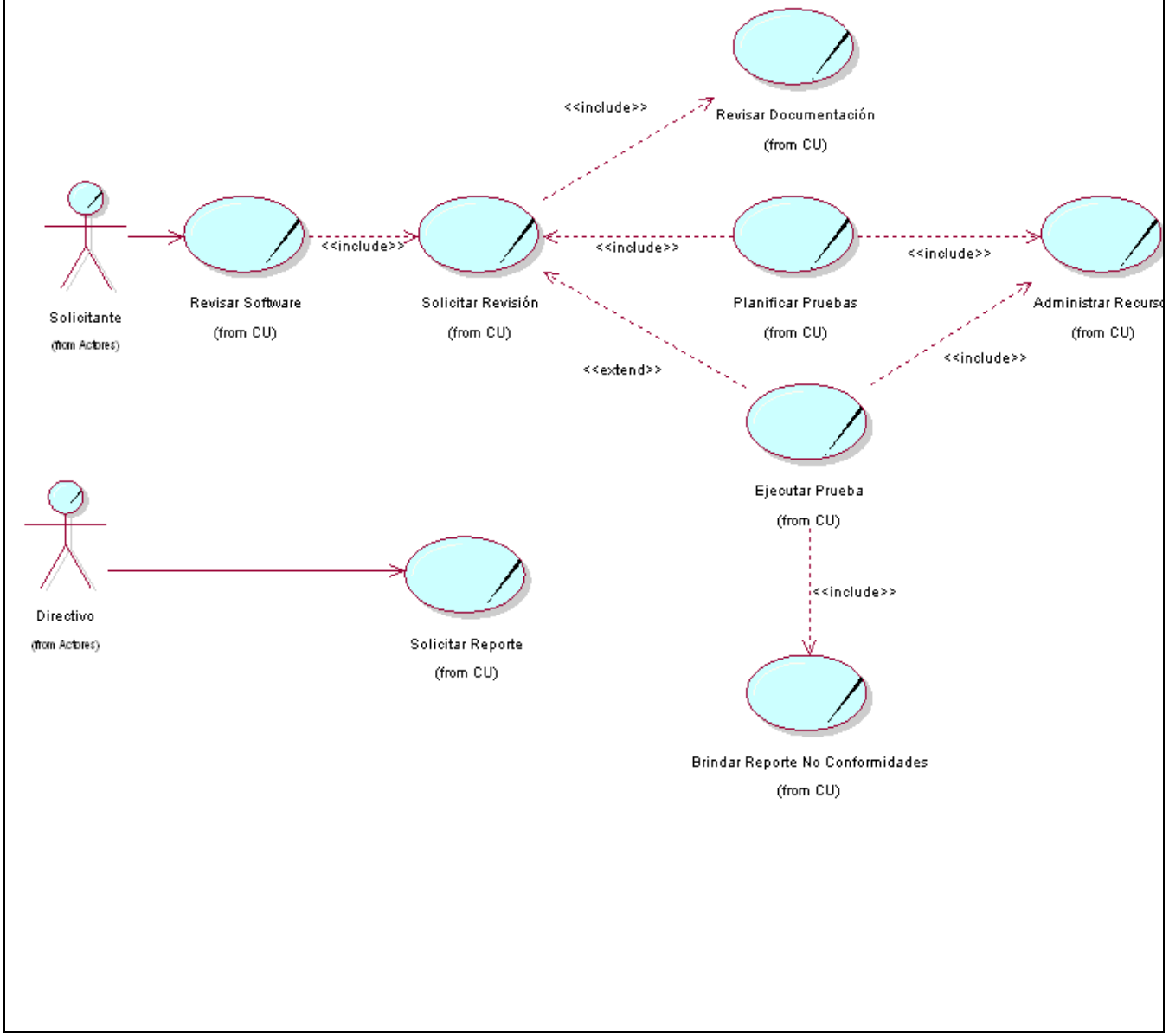
BIBLIOGRAFÍA

1. Echeverría, R.C.a.D., *Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad: Módulo Planificación.*, Universidad de las Ciencias Informáticas: Ciudad Habana.
2. Góngora, M.N.a.A., *Herramienta Informática para automatizar los procesos en el Laboratorio de Calidad de Software: Módulo Gestión de las No Conformidades.* . 2007, Universidad de las Ciencias Informáticas: Ciudad Habana.
3. Martínez, Y.L.a.K., *Sistema automatizado para los procesos en el laboratorio de Calidad. Módulo:Seguimiento de los recursos.* . 2007, Universidad de las Ciencias Informáticas: Ciudad de la Habana.
4. *SIGNO - Sistema para Gestión de No Conformidades*. [cited 2008; Available from: http://www.profitek.com.co/productos/signo.htm#arriba.
5. *Introducción a la Ingeniería de Software*. 2007 - 2008.
6. *Netcraft*. 2007 [cited 2008 Available from: http://news.netcraft.com/archives/web_server_survey.html.
7. *Una Introducción a APACHE*. 2006 [cited 2008; Available from: http://linux.ciberaula.com/articulo/linux_apache_intro.
8. Alvarez, M.A. *WampServer*. [cited 2008; Available from: http://www.desarrolloweb.com/articulos/1598.php.
9. *Fase de Inicio. Flujo de Análisis y Diseño. Modelo de Análisis.* 2007-2008.
10. *Algunos aspectos sobre Modelo de diseño.* , in *Conferencia Diseño*. 2007-2008.
11. *Material de apoyo Conferencia de Diseño*. 2008.

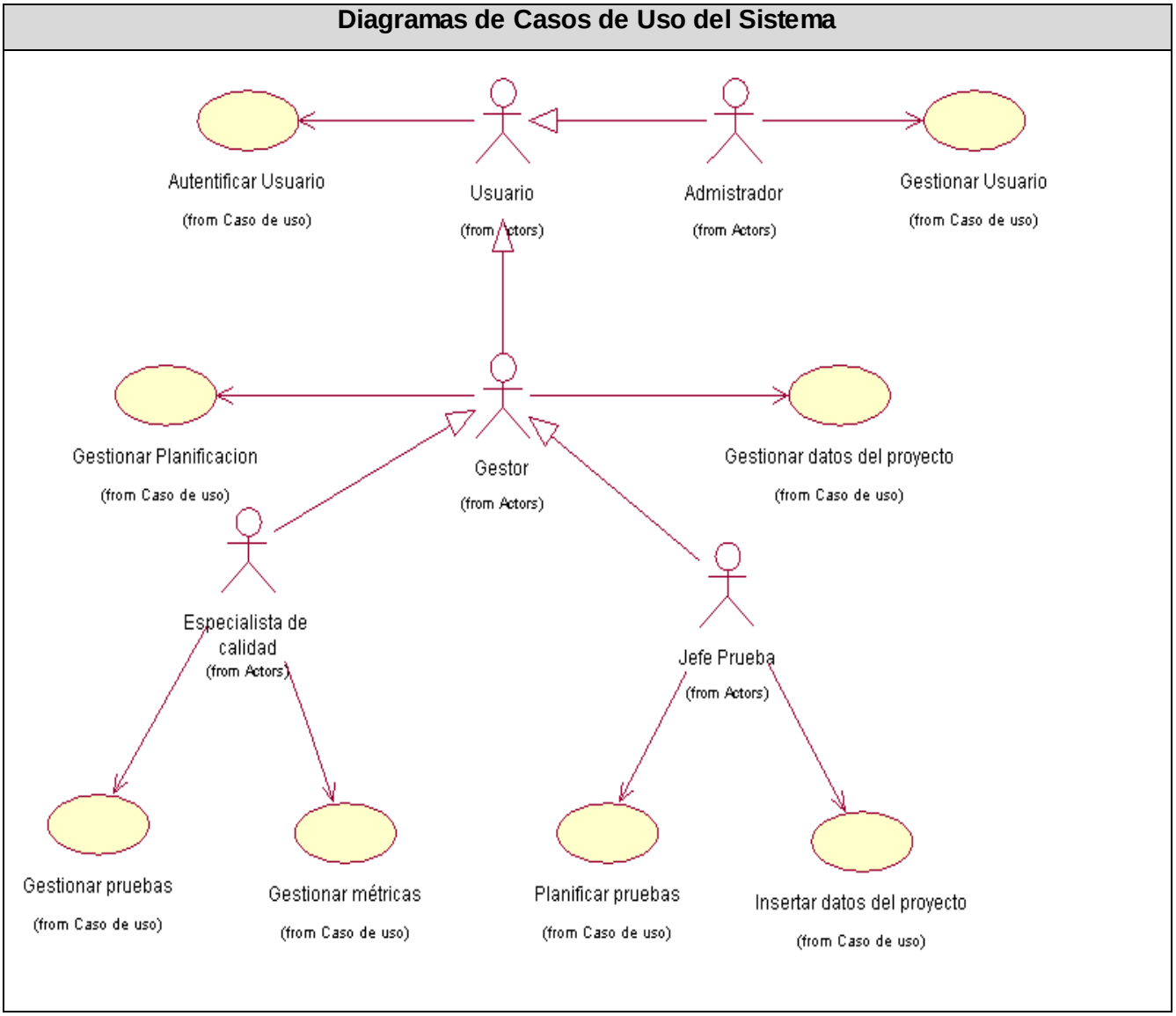
ANEXOS

Anexo # 1: Diagrama de casos de uso del negocio.

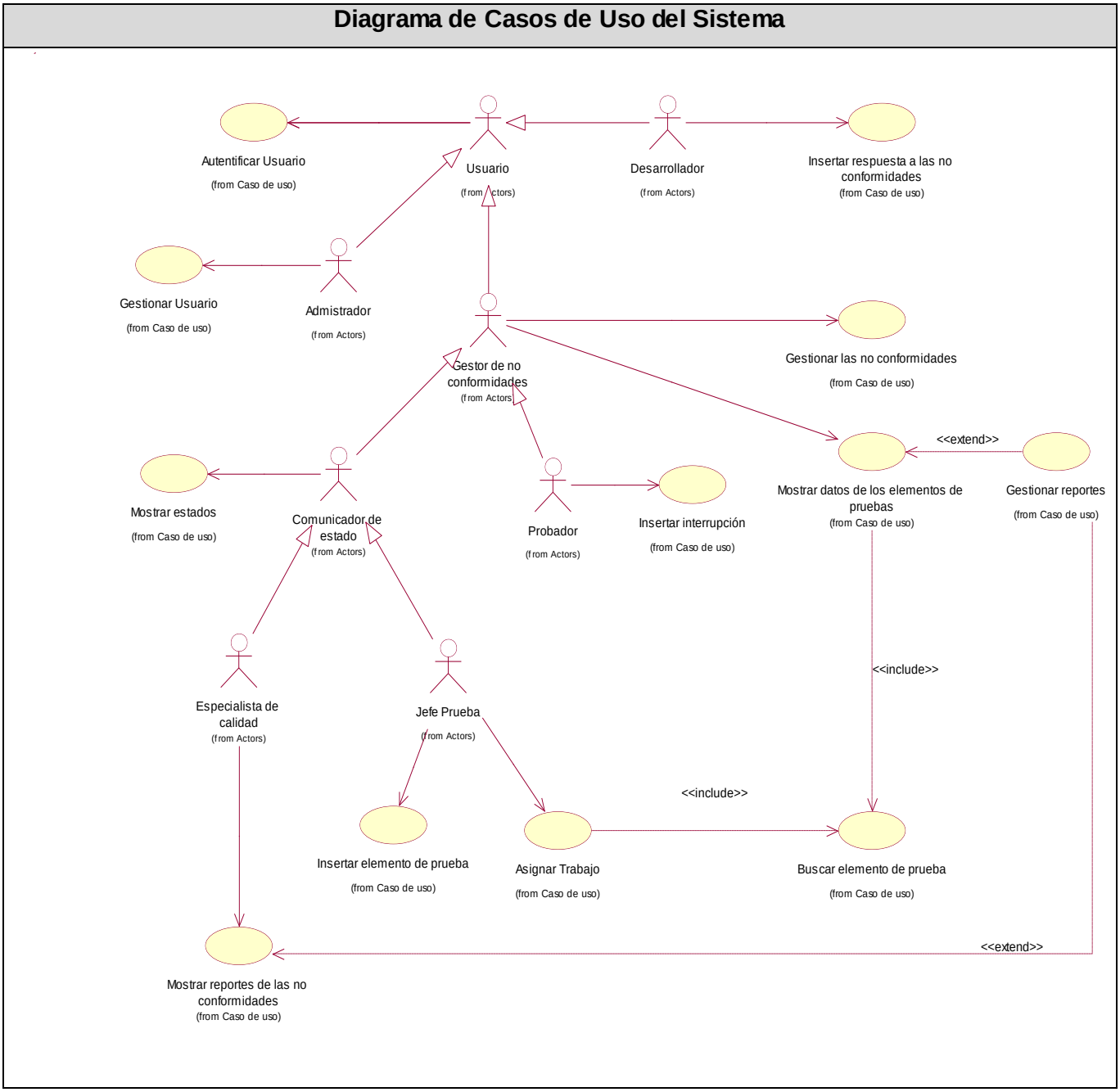
Diagrama de Casos de Uso del Negocio.



Anexo # 2: Diagrama de Casos de Usos del Sistema. "Planificación".

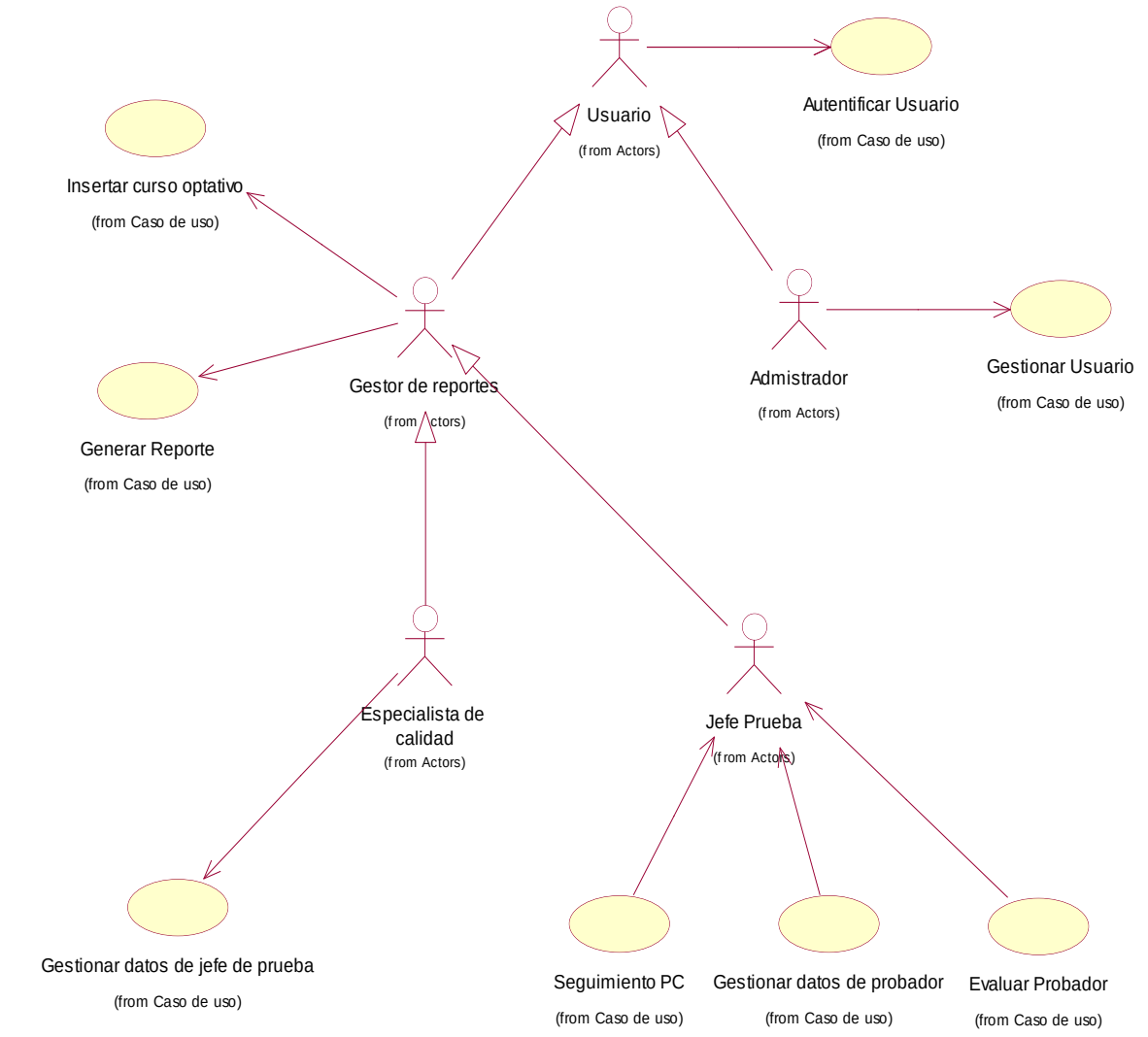


Anexo # 3: Diagrama de Casos de Uso del Sistema."Gestión de las No Conformidades"



Anexo # 4: Diagrama de Casos de Uso del Sistema."Seguimiento de los recursos"

Diagrama de Casos de Uso del Negocio



GLOSARIO

[1] **Software:** Programas, procedimientos y reglas para la ejecución de tareas específicas en un sistema de cómputo.

[2] **Calidad:** Conjunto de propiedades y características de un servicio, producto o proceso, que satisfacen las necesidades establecidas del cliente.

[3] **Pruebas de Liberación:** Pruebas que se le realizan a un software para validar su buen funcionamiento, esta prueba decide si el software está listo para ser entregado al cliente.

[4] **Certificar:** Validar la calidad del producto.

[5] **Planificación:** Acción global o conjunto de medidas pertenecientes a un plan establecido y concreto, realizado a la consecución de un fin.

[6] **Seguimiento:** Es el proceso mediante el cual se recopilan sistemáticamente y con cierta regularidad los datos referidos al desarrollo de un programa a lo largo del tiempo.

[7] **No Conformidades:** Un no cumplimiento a un requisito.

[8] **Métricas:** Duración de la prueba según la complejidad de lo que se vaya a probar (puede ser caso de uso o capítulos).