Universidad de las Ciencias Informáticas

Facultad # 4

Título: Sistema Asistente para Gestionar Pruebas de Caja Negra

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Lianet Aguilera Reyes

Annie Cuesta Rodríguez

Tutor: Ing. Michael González Jorrín

Julio del 2007

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 _____.

Lianet Aguilera Reyes	Annie Cuesta Rodríguez	Michael González Jorrín
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Sistema Asistente para Gestionar Pruebas de Caja Negra

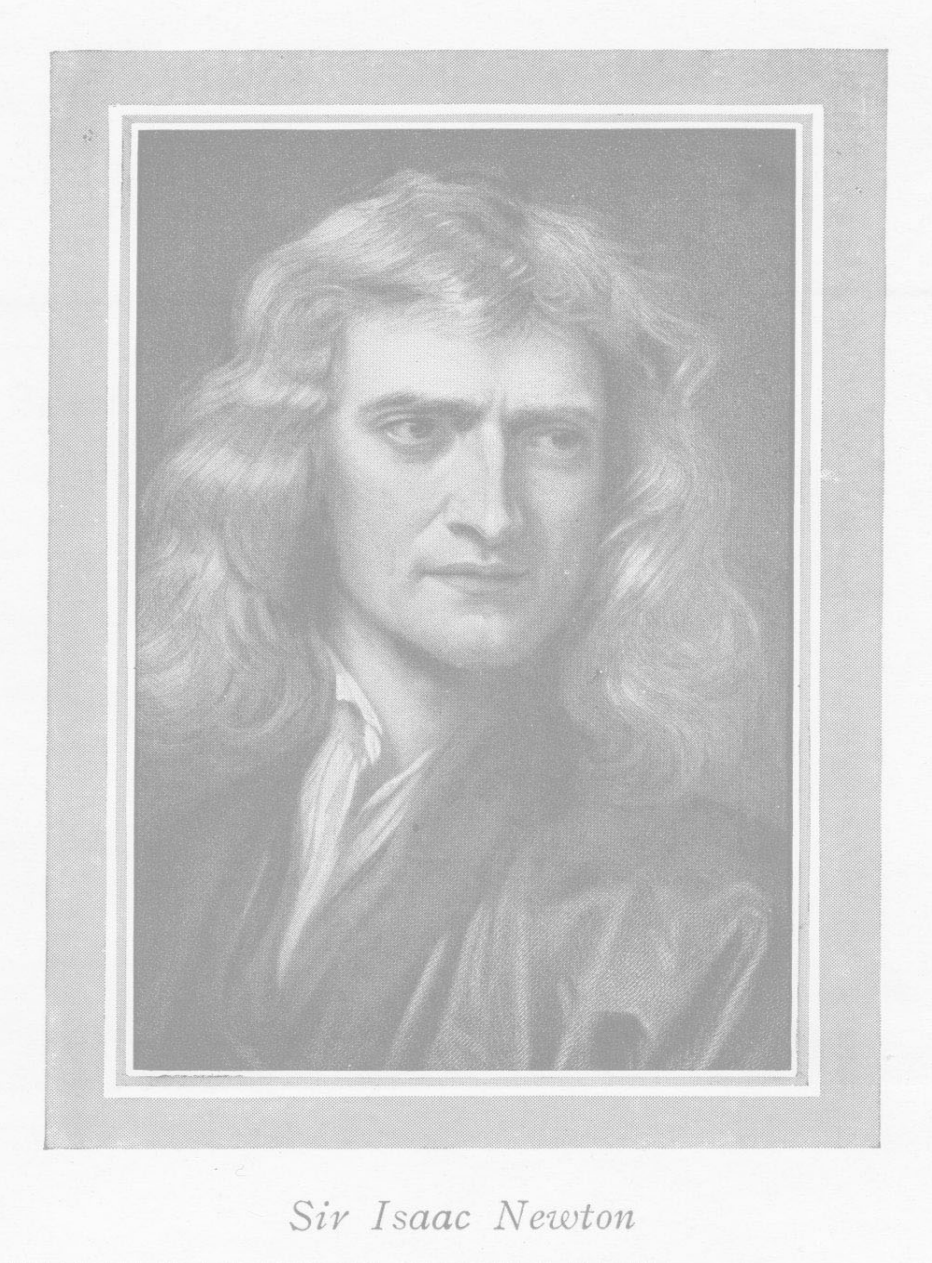
Autores: Lianet Aguilera Reyes y Annie Cuesta Rodríguez

El tutor del presente Trabajo de Diploma considera que durante su ejecución las estudiantes mostraron las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de ____ .

_____ ____ de julio del 2007

Firma



"Lo que sabemos es una gota de agua; lo que ignoramos es el océano."

Isaac Newton

Agradecimientos

"A las autoras les complace dejar constancia de su reconocimiento a un grupo de compañeros que con su eficaz concurso permitieron arribar a la culminación de este trabajo y al mismo tiempo sienten el temor a la omisión cuando se señale a Michael González Jorrín, Michel López Camino, tutor y colaborador, respectivamente; a Alexander García García, Yosvani Turruelles Tejeda, a los propios padres, a muchos familiares, algunos ya ausentes como el querido "Apocho", a todos los profesores, especialmente de la UCI, por eso y otras obvias razones han preferido resumir en un agradecimiento general a la obra de la Revolución, a Fidel por ser la máxima representación de ella, a la sociedad, de la que se consideran eternas deudoras".

Resumen

El presente trabajo tiene como tema Sistema Asistente para Gestionar Pruebas de Caja Negra. En el país, y específicamente en la Universidad de las Ciencias Informáticas no existe un sistema que apoye y ayude a los especialistas de calidad a realizar las pruebas de caja negra de forma automatizada lo que trae consigo que este proceso no se realice con la eficiencia requerida. Debido a la problemática existente se hace necesaria la creación de un prototipo, que ayude a reunir la mayor cantidad de requisitos posibles, para una vez que esté listo proceder a la implementación de un futuro Sistema Asistente para gestionar pruebas de caja negra.

Después de realizar una intensa investigación en el tema de las pruebas de caja negra y sus adyacentes se ha llegado a la elaboración de un prototipo que cumple con los requisitos propuestos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La sociedad cada vez necesita más de los servicios informáticos para su desempeño y es esta, una de las razones por las cuales esta rama de las ciencias ha tomado tanto auge y crece tan vertiginosamente en los últimos tiempos, aparejado a ello la demanda de software con más calidad en menos periodo de tiempo y más barato.

Al calor de la batalla de ideas surge la idea por nuestro Comandante de la creación de la Universidad de las Ciencias Informáticas (UCI), que para Cuba sería la mayor fuente de creación de software para la informatización de nuestra sociedad y para insertarse también en el mercado mundial de software y así elevar la economía nacional.

Los clientes muchas veces se quejan de la calidad del producto, debido a que no quedan satisfechos por muchas razones, una de ellas es que el software no cumple con todos los requisitos requeridos.

La UCI para cumplir con estos objetivos y brindar a los clientes productos que cumplan sus exigencias se ha trazado garantizar la calidad de sus producciones. Para lograr esto se hace imprescindible realizar pruebas de software para alcanzar el nivel de calidad requerido.

Las pruebas de software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y la codificación.

El software debe probarse desde dos perspectivas diferentes: la lógica interna del programa se comprueba utilizando técnicas de diseño de casos de prueba de caja blanca y los requisitos del software se comprueban utilizando técnicas de diseño de casos de prueba de caja negra. En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo. [1]

Este trabajo se centra en el proceso de pruebas de caja negra que lleva a cabo la UCI para garantizar la calidad de los productos elaborados en la propia universidad. Con dicho objetivo la Universidad de las Ciencias Informáticas y en especial la Dirección de Calidad y Normas de la misma ha decidido la creación de un Sistema Asistente para Gestionar Pruebas de Caja Negra, este sistema servirá de apoyo al

personal involucrado en el proceso de pruebas, permitirá registrar los resultados que las mismas arrojan, estas pruebas se realizarán en menos tiempo y ayudará en la toma de decisiones.

Las pruebas de caja negra son las pruebas que se le realizan desde la interfaz del producto, para verificar que sus funcionalidades sean operativas, estas pruebas se realizan sin interesar la parte interna del programa. Este tipo de prueba intenta descubrir errores y casos en el que el módulo no cumple con sus especificaciones.

En el momento de realizar las pruebas de caja negra existe una pérdida de información (datos y fechas del probador y el evaluador), dificultad al construir un expediente, es difícil darle seguimiento a un proceso, el control de las no conformidades se dificulta (registro, clasificación y seguimiento), hay problemas de uniformidad en el manejo de plantillas, dificultades en el entrenamiento del probador y al llevar estadísticas de las personas, proyectos, productos y defectos, se hace imposible la recolección de datos que permitan la aplicación de métricas adecuadas de calidad, se dificulta la estimación del tiempo de pruebas, así como la confección de cronogramas. De toda esta problemática se deriva la molestia de los trabajadores puesto que éstos están tan interesados como el cliente de que el producto quede con la máxima calidad y cumpla con los requisitos exigidos. Los problemas mencionados anteriormente son debido a que las pruebas de caja negra se realizan de forma manual.

Hasta antes de la revolución industrial la calidad se determinaba por la inspección personal por parte del cliente, el cual confiaba en la habilidad del fabricante, ya que éste era siempre un artesano conocido. Con la expansión de los mercados en las ciudades, los artesanos se agruparon en gremios y la calidad se certificaba por medio de especificaciones impuestas entre ellos mismos y por la inspección del comportamiento y desempeño de sus integrantes. Con la revolución industrial llego la estandarización y las especificaciones escritas de los materiales, procesos, etcétera; pero conservando mucho de los principios de la labor artesanal.

Al finalizar el siglo XIX la escuela de administración científica de Taylor rompió con los esquemas del pasado y puso énfasis en la productividad, con lo cual la calidad pasó a segundo plano. Antes, la inspección de la calidad era realizada por la misma gente que participaba en la producción, sin embargo la

división de funciones del taylorismo encomendó el control de calidad a un departamento central de inspección.

De aquí en adelante se fueron perfilando las técnicas para el control de la calidad y en la cantera de la informática más aún, pues ésta siempre está en constante desarrollo y evolución. Una de estas técnicas es la de Prueba de caja negra que no hace más que realizar pruebas sobre la interfaz del software para probar que sus funcionalidades sean operativas.

En la actualidad en las pruebas de software intervienen probadores, coordinadores de calisoft, evaluadores y especialistas, mientras se ejecutan estas pruebas se va dejando constancia de los errores encontrados en un documento de no conformidades y finalmente dicho documento es entregado al cliente que solicitó la realización de las pruebas.

El objetivo general desarrollar un prototipo de un sistema asistente para gestionar pruebas de caja negra, que sea capaz de simular los procesos que tendrá el futuro sistema.

De acuerdo con esto resultan los siguientes objetivos específicos, de los cuales los dos primeros responden a los objetivos de investigación y los restantes a los objetivos relacionados con el objeto de automatización:

1. Conceptualizar sobre modelos y estándares de calidad, calidad de software, pruebas, prototipos, tendencias, tecnologías y herramientas.
2. Identificar los elementos que componen el proceso de pruebas general llevadas a cabo en la UCI.
3. Obtener los requisitos primarios para la confección del prototipo.
4. Confeccionar un Prototipo de un Sistema asistente para realizar pruebas de caja negra.
5. Obtener el análisis y diseño de un Sistema Asistente para Gestionar Pruebas de Caja Negra.

Para alcanzar los objetivos anteriormente planteados y dar solución a la situación problémica descrita en el Capítulo 2 es imprescindible la realización de las siguientes tareas.

Tareas para cumplir el objetivo 1

- Estudio de los modelos y estándares de calidad, calidad de software, pruebas, prototipos, tendencias, tecnologías y herramientas.
- Estudio de las aplicaciones existentes en el mundo que se dedican a apoyar el proceso de pruebas de caja negra.
- Seleccionar las herramientas a utilizar para la confección de la solución propuesta.

Tarea para cumplir el objetivo 2

- Investigación del proceso de pruebas llevadas a cabo en la UCI.

Tarea para cumplir el objetivo 3

- Recopilación de los requisitos de la solución.

Tarea para cumplir el objetivo 4

- Confección de la solución propuesta, o sea, el prototipo.

Tareas para cumplir el objetivo 5

- Análisis y diseño del sistema.
- Propuesta del lenguaje y herramientas para la construcción del sistema.

Los métodos utilizados para dar cumplimiento a las tareas trazadas en este trabajo fueron los siguientes:

Métodos teóricos:

Análisis y síntesis: Mediante este método se realizó el procesamiento de la información, se tomó el problema y se dividió en sub-problemas para un mejor entendimiento, se analizaron los conceptos por separado profundizando en cada uno de ellos y contrastando cada uno de los autores, conceptos como prototipo, calidad de software, plan de prueba, casos de prueba, pruebas de caja negra. Una vez entendido el objeto de estudio, de la situación problémica y la teoría se pasó a la unión de todos estos conocimientos para dar solución al problema.

Histórico lógico: Brindó la posibilidad de conocer el proceso actual de pruebas llevadas a cabo en la UCI y la situación en el mundo de sistemas de apoyo en la realización de pruebas de caja negra, este método reproduce el objeto de en su forma superior y permite unir al estudio de la estructura del objeto de investigación con su concepción histórica y con el tratamiento de la información.

Métodos empíricos:

Entrevistas: Constituyó un medio para el conocimiento de los problemas existentes al realizar las pruebas de forma manual, y conocer la información referente a como se espera que funcione el sistema. Adquiriendo experiencias sobre el proceso de pruebas para solucionar el problema de forma eficiente.

Análisis de documentos: Con este método se analizaron los documentos que se manejan en el proceso de pruebas de caja negra para identificar los elementos que se desean automatizar.

De estos métodos, resultan los más utilizados el análisis de documentos y el análisis y síntesis ya que fueron los que en gran parte ayudaron a la solución del problema.

Al concluir la investigación se espera obtener un prototipo y el análisis y diseño lo más completo posible del futuro Sistema Asistente para Gestionar Pruebas de Caja Negra. Con ello se logrará agilizar el proceso de pruebas a un software, involucrar menos personal a la hora de hacer las pruebas, permitir el diseño de casos de pruebas de forma eficiente, una mayor calidad del producto, ayudar en la toma de decisiones, la obtención de métricas y mejorar la gestión de proyectos y de información. Y lo más importante satisfacer las necesidades del cliente.

El presente documento que se presenta como culminación de estudios para optar por el título de Ingeniero en Ciencias Informáticas está compuesto por tres capítulos, que incluyen todo lo relacionado con el trabajo investigativo realizado, con el cual se llegaron a varias conclusiones, así como el análisis y diseño del sistema y el prototipo que lo simula. Además cuenta con resumen, índice, introducción, conclusiones, recomendaciones, bibliografía, referencias bibliográficas, anexos y un glosario de términos.

En el Capítulo 1: Fundamentación teórica, se realiza un estudio realizado sobre los modelos y estándares de calidad, calidad de software, casos de pruebas, plan de pruebas, pruebas de caja negra y tipos de pruebas de caja negra, prototipos y tipos de prototipos, importancia de éstos, Internet, lenguajes de programación web, arquitectura cliente-servidor y herramientas. Por otra parte se efectuó un estudio a nivel internacional de las aplicaciones existentes similares al proceso que simulará la solución propuesta.
En el Capítulo 2: Características del sistema, se define cual es el objeto de estudio, problema y situación problémica, objeto de automatización, una propuesta del sistema, los requisitos funcionales y no funcionales así como el modelado del negocio. Además en este capítulo se describen los procesos que se realizan para realizar las pruebas de caja negra.

En el Capítulo 3: Análisis y Diseño del sistema, se presentan diferentes diagramas como: el diagrama de clases del análisis, de interacción, de clases del diseño, una descripción de éstas últimas, el diseño de la base de datos, así como los diagramas de despliegue y de componentes. Todos estos elementos constituyen la base de la construcción del futuro Sistema Asistente para Gestionar Pruebas de Caja Negra.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Este capítulo ofrece un enfoque general del estudio realizado sobre calidad de software y requisitos de calidad, los modelos y estándares de calidad, pruebas: casos de pruebas, plan de pruebas, pruebas de caja negra y tipos de pruebas de caja negra, prototipos, Internet, lenguajes de programación web, arquitectura cliente-servidor y herramientas como: Macromedia Dreamweaver 8, Adobe Photoshop CS2 y Rational Rose. Por otra parte se efectuó un estudio a nivel internacional de las aplicaciones existentes similares al proceso que simulará la solución propuesta.

Seguidamente es necesario conocer sobre la calidad de software, ya que la propuesta de solución debe cumplir con ello, para que así el producto final quede como el cliente lo exige.

1.2 Calidad de Software

La calidad de software es un tema muy estudiando, ya que ésta es un problema que afecta tanto a clientes como a creadores. Con el desarrollo creciente de la informática en los últimos tiempos y la necesidad cada vez más amplia de la informatización de la sociedad aparejado a ello, los clientes y creadores le dedican menos importancia a la calidad de sus productos.

La calidad de software es una actividad de protección que se aplica a lo largo de todo el proceso de Ingeniería del Software. Ésta engloba los siguientes aspectos:

- ✓ Un enfoque de gestión de calidad.
- ✓ Tecnología de Ingeniería del Software efectiva (métodos y herramientas).
- ✓ Revisiones técnicas formales que se aplican durante el proceso del software.
- ✓ Una estrategia de prueba multiescala.
- ✓ El control de la documentación del software y de los cambios realizados.
- ✓ Un procedimiento que asegure un ajuste a los estándares de desarrollo del software.
- ✓ Mecanismos de medición y de generación de informes.

La calidad debe ser especificada, planificada, administrada, medida y certificada. Esto implica una visión integral que arroja la comprobación del software, con el fin de lograr un mayor grado de satisfacción y confianza del cliente hacia la organización productora de software. Constituye entonces las pruebas del software, tarea de alta prioridad para las empresas productoras. [4]

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. [5]

1.2.1 Requisitos de Calidad

Para asegurar la calidad de un producto software el mismo deberá cumplir con los requisitos de calidad Planteados inicialmente por la Dirección de Calidad y Normas de la UCI, los cuales serían: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

Finalizado este epígrafe se arriba a que la mejor y más completa definición de calidad de software es la anteriormente planteada, porque una vez que el producto cumpla con todas esas cualidades se puede decir que es un producto con calidad y apto para su uso.

Seguidamente se realiza un estudio acerca de los modelos y estándares de calidad, puesto que estos modelos y estándares normalizan como debe ser medida la misma.

1.3 Modelos y estándares de Calidad

Para garantizar la calidad de software se debe seguir un plan de calidad. El plan de calidad se basa en las normas y modelos.

Una norma de calidad puede considerarse una regla o directriz para un conjunto de actividades. Es un documento, establecido por consenso y probado por un organismo o entidad reconocida (nacional o internacional) que proporciona, para un uso común y repetido, reglas, directrices o características para las

actividades de calidad o sus resultados, con el fin de conseguir un grado óptimo y eficiente de orden en el contexto de la calidad.

Un Modelo de Calidad es un conjunto de buenas prácticas para el ciclo de vida del software, enfocado en los procesos de gestión y desarrollo de proyectos. [2]

El uso de normas y modelos de calidad en la confección de productos software proporciona diversas ventajas al proyecto y al producto en sí, pues ayuda a garantizar la calidad del software que se esté desarrollando. Se considera una de las mejores maneras de garantizar que se cumpla con los requisitos solicitados y esperados en todo momento por el cliente; no cabe dudas de que su puesta en práctica reduce considerablemente la probabilidad o riesgo de ocurrencia de errores logrando una mayor fiabilidad.

Organizaciones como la ISO (International Organization for Standardization), BOOTSTRAP, IEC (International Electrotechnical Commission), entre otras se han dedicado a crear modelos y normas

para que las empresa se rijan por principios de organización y para que den estabilidad en el mercado y en la sociedad para es orientar, coordinar, simplificar y unificar los usos para garantizar la calidad de los procesos, entre ellas están:

- ISO 9000-3
- CMMI
- ISO/SPICE
- MoProSoft
- TICK IT

1.3.1 ISO 900-3 Norma para la gestión y aseguramiento de la calidad

La ISO 9000-3 proviene de la orientación de la organización ISO en la aplicación de la ISO 9001:2000 que fue escrita para ser utilizada por toda clase de industrias, es regularmente difícil interpretarla para el desarrollo de software, por lo cual se publicó la ISO 9000-3 que sirve de guía para la aplicación de ISO 9001 para el desarrollo, implementación, funcionamiento y mantenimiento de software.

1.3.2 ISO/SPICE 15504

Es un marco para métodos de evaluación, no un método o modelo en sí; y se define que abarca:

- Evaluación de procesos
- Mejora de procesos
- Determinación de capacidad

Este estándar internacional ISO/IEC 15504 define, a un alto nivel, las actividades fundamentales que son esenciales para una buena ingeniería del software. Describe qué actividades se requieren, no cómo se van a implementar. Las prácticas base pueden ampliarse mediante la generación de guías prácticas de un sector específico para tener en cuenta una industria, sector u otros requisitos específicos**.**
El modelo describe los procesos que una organización puede ejecutar, adquirir, suplir, desarrollar, operar, evolucionar, brindar soporte de software y todas las prácticas genéricas que caracterizan las potencialidades de estos procesos.

El modelo agrupa a los procesos en cinco categorías:

1. Procesos Cliente
2. Procesos de Ingeniería.
3. Procesos de Proyecto.
4. Procesos de Soporte.
5. Procesos de la Organización.

1.3.3 La integración del modelo de la madurez de la capacidad (CMMI)

El CMM es desarrollado a partir de 1987 hasta 1997 y sirvió de base a su sucesor CMMI. En 2002 la versión 1.1 de CMMI fue lanzada; y seguida a ésta la v1.2 en agosto de 2006. La versión última CMMI v1.2 consta de 22 áreas de proceso con los niveles de la capacidad o de la madurez.

Es conveniente el uso de CMMI pues se debe adaptar a cada compañía individual, por lo que puede decirse que este nuevo modelo no certifica a las compañías sino que las valora mediante un método determinado en un nivel específico.

Las ayudas CMMI integran tradicionalmente funciones de organización separadas, proporcionan la dirección para los procesos de la calidad, fijan metas de la mejora de proceso y prioridades, y proporcionan un punto de la referencia sólido para valorar procesos actuales.

El CMMI en su estructura viene con dos representaciones, estas son: escalonada (por niveles de madurez) y continua. La primera, agrupa las áreas de proceso de los grupos en 5 niveles de la madurez e integración (Inicial o Nivel 1, Repetible o Nivel 2, Definido o Nivel 3, Cuantitativamente Gestionado o Nivel 4 y Optimizado o Nivel 5). La representación continua, que fue utilizada en la ingeniería de sistemas del antepasado CMM, define niveles de la capacidad dentro de cada perfil. El modelo para ingeniería de sistemas (SE-CMM) establece 6 niveles posibles de capacidad para una de las 18 áreas de proceso implicadas en la ingeniería de sistemas. No agrupa los procesos en 5 tramos para definir el nivel de madurez de la organización, sino que directamente analiza la capacidad de cada proceso por separado. Las diferencias en cualquiera de las dos representaciones citadas anteriormente son solamente de organización; porque el contenido es equivalente.

CMMI ventajas:

Las mejores prácticas CMMI permiten a las instituciones:

- Ligar más explícitamente las actividades de la gerencia y de la ingeniería a tus objetivos de negocio.
- Ampliar el alcance y la visibilidad en las actividades del ciclo de vida y de la ingeniería de producto para asegurarse de que las expectativas del cliente de las reuniones del producto o del servicio.
- Incorporar las lecciones aprendidas de áreas adicionales de la mejor práctica (e.j. medida, gerencia de riesgo, y la gerencia del surtidor).
- Poner prácticas más robustas con un alto grado de madurez en ejecución.
- Poner prácticas más robustas de la alto-madurez en ejecución.
- Tratar las funciones de organización adicionales críticas a tus productos y servicios.

1.3.4 MoProSoft y Tick IT

MoProSoft y Tick IT están basados en otros modelos internacionales como ISO, y CMMI; constituyen una alternativa para pequeñas y medianas empresas sentando las bases para lograr la certificación posterior con otras normas, están desarrollados según las características específicas de sus países de origen.
Para escoger un modelo para la realización de un proyecto y aplicación a determinado producto, se debe hacer un estudio de los modelos que sean adaptables al proyecto y al producto en sí, para escoger de ellos el que se ajuste de mejor forma al proceso que se esté llevando a cabo o se desee desarrollar, así como otros parámetros que se estimen importantes tener en cuenta para hacer una acertada selección.
Habiéndose realizado un estudio de algunos modelos y normas se presenta una descripción de ellos que tratan de alguna manera las pruebas o verificación como llaman algunos autores, de los productos software, evaluando en todo momento la posibilidad de adaptar estos modelos a los proyectos que desarrolla la Universidad de las Ciencias Informáticas.

Todos estos modelos y estándares tienen un solo objetivo que es lograr la calidad del producto, realizando una comparación entre sus características se realiza la elección de cual estándar usar para la normalización de la producción.
Los objetivos de dicha normalización son: [3]

Aumentar la confianza del usuario final en las aplicaciones desarrolladas, ofreciéndole un entorno familiar, sin sorpresas ni excepciones.

- Promover la colaboración entre los diferentes grupos de desarrollo permitiendo que soluciones elaboradas para una aplicación se utilicen en el desarrollo de otras.
- Facilitar el mantenimiento de aplicaciones haciendo que un módulo no esté ligado al programador del mismo.

La implantación de una norma o estándar debe comenzar con un análisis de lo que existe lo que se necesita y además se debe conocer bien el ambiente y las circunstancias internas de la organización de tal forma que sea útil, de bajo costo y acorde a la organización.

Después de haber estudiado los modelos y estándares de calidad se llega a la conclusión de que debe usarse CMMI, porque ha demostrado que mejora las prácticas más óptimas de modelos anteriores. CMMI está en la vanguardia de la mejora de proceso, su uso proporciona las mejores prácticas, más recientes y novedosas para el producto, desarrollo y mantenimiento de servicio. Además el producto que se desea hacer debe asistir como soporte a las pruebas de caja negra relacionadas con algunas prácticas y áreas de proceso de CMMI como son: Desarrollo de Requerimientos, Solución Técnica, Comprobación y la Validación.

1.4 Pruebas

Para comprobar la calidad del producto es necesario realizar pruebas, por tanto este tema también debe ser abordado para una mejor comprensión del Capítulo 2.

1.4.1 Casos de Prueba

Los Casos de pruebas definen un conjunto específico de entradas de pruebas, ejecución de condiciones y resultados esperados. [6]

Caso de pruebas es una condición a ser probada que incluye su propia identificación y la respuesta esperada. [7]. Un diseño de los casos de prueba es fundamental para el éxito de las pruebas.

Otro concepto de caso de prueba es: un caso de prueba específica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse [8]

Para elaborar los casos de pruebas y aplicarlos hay que tener en cuenta dos aspectos importantes:

a) Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces.

b) Una prueba tiene éxito si se descubren errores.

En la Ingeniería del software, los casos de prueba especifican una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. [9]

Un caso de prueba es como un reemplazo de un actor del sistema. El caso de prueba simula las interacciones del actor con el sistema para verificar que el sistema hace lo que se espera de él. Así, el principal artefacto para obtener pruebas del sistema son los requisitos funcionales.

Los Casos de Prueba, no son más que hipótesis (enunciado de la relación entre dos o más variables sujetas a comprobación empírica), de ahí que las “variables” sean propiedades que pueden variar y cuya variación es susceptible de medirse.

Como bien se dice en la definición que aparece en el párrafo 8 de este sub-epígrafe: “Un caso de prueba es como un reemplazo de un actor del sistema” es totalmente verídica, porque un caso de prueba es un conjunto de datos que se le entran a la interfaz para verificar si los resultados son los esperados, lo cual es lo mismo que hace un actor del sistema, al trabajar con éste el actor puede comprobar si ejecutada una acción recibe lo deseado.

Para poder llevar a cabo un caso de prueba hace falta organizarse y esto lo permite un plan de pruebas.

1.4.2 Plan de Prueba

El propósito del plan de pruebas es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario, los responsables y el manejo de riesgos de un proceso de pruebas. [10]
Cada prueba debe:

- Dejar claro qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad, amigabilidad,...).
- Dejar claro cómo se mide el resultado.
- Especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta).
- Definir cual es el resultado que se espera (identificación, tolerancia,...). ¿Cómo se decide que el resultado es acorde con lo esperado?
- Las pruebas carecen de utilidad, tanto, sí no se sabe exactamente lo que se quiere probar, sí no se está claro cómo se prueba, o si el análisis del resultado se hace a simple vista.

Estas mismas ideas se suelen agrupar diciendo que un caso de prueba consta de 3 bloques de información:

1. El propósito de la prueba.
2. Los pasos de ejecución de la prueba.
3. El resultado que se espera.

Todos y cada uno de esos puntos deben quedar perfectamente documentados.
El plan de pruebas señala el enfoque, los recursos y el esquema de actividades de prueba, así como los elementos a probar, las características, las actividades de prueba, el personal responsable y los riesgos.

Plan de prueba: Es un documento de alto nivel que define el proyecto de prueba del software, para poder medirlo y controlarlo correctamente. Define la estrategia de la prueba y los elementos organizados del ciclo vital de la prueba, incluyendo requisitos de recurso, proyecciones de horario y prueban requisitos. [11]

La definición anterior está muy clara e integral, un plan de pruebas es imprescindible para la organización, para realizar pruebas al software estas pruebas deben estar en un documento para poder medir y controlar mejor los resultados.

El proceso de pruebas que se desea simular en la propuesta de solución, se incluye dentro de las pruebas de caja negra, por tanto a continuación se explica en que consisten dichas pruebas y los tipos más utilizados de estas pruebas que existen.

1.4.3 Pruebas de Caja Negra

Existen muchas definiciones acerca de lo que es una prueba de caja negra, a continuación se muestran algunas:

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores. [12]

En las pruebas de caja negra se comprueban las funcionalidades sin tener en cuenta la estructura interna. [13]

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. [14]

También conocido como la comprobación funcional. Una técnica de software donde el funcionamiento interno del elemento a probarse no es conocido por el probador. Por ejemplo, en una prueba de la caja negra en un plan del software el verificador sabe sólo las entradas y lo que los resultados esperados deben ser y no cómo el programa llega a esos rendimientos. El verificador nunca examina el código de la programación y no necesita cualquier conocimiento extenso del programa que sus especificaciones. [15]

La definición más completa de Prueba de Caja Negra es propuesta por Beizer, en el párrafo anterior, ya que en ella incluye todo lo referente a este tipo de prueba de software, que no hace más que verificar las funcionalidades de los requisitos obteniendo condiciones de entradas para las cuales las respuestas deben ser válidas sin interesarle a la persona que prueba como está implementada esa funcionalidad.

1.4.3.1 Tipos de Prueba de Caja Negra

Mediante las técnicas de prueba de caja negra se obtiene un conjunto de casos de prueba que satisfacen los siguientes criterios: (1) casos de prueba que reducen, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable y (2) casos de prueba que nos dicen algo sobre la presencia o ausencia de clases de errores en lugar de errores asociados solamente con la prueba que estamos realizando. [16].

- **Partición equivalente**

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o

inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. [17]
Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- ✓ Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- ✓ Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- ✓ Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- ✓ Si una entrada es booleana, hay 2 clases: si o no.
- ✓ Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar
- ✓ resultados en todas y cada una de las clases.

Aplicando estas directrices se ejecutan casos de pruebas para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez.

- **Valores Límites**

La técnica de Análisis de Valores Límites selecciona casos de prueba que ejerciten los valores límite dada la tendencia de la aglomeración de errores en los extremos. Complementa la de la partición equivalente. En lugar de realizar la prueba con cualquier elemento de la partición equivalente, se escogen los valores en los bordes de la clase. Se derivan tanto casos de prueba a partir de las condiciones de entrada como con las de salida. [18]

- **Técnica de Grafos de Causa-Efecto**

Es un enfoque sistemático para seleccionar conjuntos de casos de prueba de alto rendimiento que exploran combinaciones en las condiciones de entrada. Se registran restricciones describiendo

combinaciones de causas y efectos posibles, también se crean grafos de causa-efecto y se convierten en tablas de decisiones.
Proporcionan una concisa representación de las condiciones lógicas y sus correspondientes acciones. Sigue cuatro pasos:
1. Se listan para un módulo las causas (condiciones de entrada) y los efectos (acciones), asignando un identificador a cada uno de ellos.
2. Se desarrolla un grafo de causa-efecto.
3. Se convierte el grafo en una tabla de decisión.
4. Las reglas de la tabla de decisión se convierten a casos de prueba.

De las técnicas anteriormente mencionadas resulta como la más efectiva una combinación entre Partición equivalente y Valores Límites, la primera permite simplificar el número de casos de prueba, obteniendo valores válidos e inválidos de las entradas. Teniendo como ventaja que cada vez que un usuario usa el programa está a la vez realizando una prueba al mismo, mientras que la segunda complementa a la de Partición equivalente porque en lugar de realizar la prueba con cualquier elemento de la partición equivalente, se escogen los valores en los bordes de la clase. Se derivan tanto casos de prueba a partir de las condiciones de entrada como con las de salida.

La propuesta de solución es un prototipo, en el siguiente epígrafe se hace referencia al tema para una mejor comprensión del lector acerca de lo que se quiere hacer.

1.5 Prototipos

1.5.1 Prototipo

Un prototipo es una representación limitada del diseño de un producto que permite a las partes responsables de su creación experimentar su uso, probarlo en situaciones reales y explorar su uso.
Un prototipo puede ser cualquier cosa, desde un trozo de papel con sencillos dibujos a un complejo software.[19]
Un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y entender mejor el “problema” y su solución.[20]

Los prototipos pueden ser el comienzo de futuros software o sólo una forma de mostrarle al cliente los requisitos que debe cumplir el sistema y luego desecharse puesto que no se parecerá al producto final.

1.5.1.1 Características de un prototipo

Baja Fidelidad vs. Alta Fidelidad

- Baja Fidelidad: conjunto de dibujos (por ejemplo, una presentación de escenarios) que constituye una maqueta estática, no computerizada y no operativa de una interfaz de usuario para un sistema en planificación.
- Alta Fidelidad: conjunto de pantallas que proporcionan un modelo dinámico, computerizado y operativo de un sistema en planificación.

Exploratorio vs. Experimental vs. Operacional

- Exploratorio: prototipo no reutilizable utilizado para clarificar las metas del proyecto, identificar requerimientos, examinar alternativas de diseño o investigar un sistema extenso y complejo.
- Experimental: prototipo utilizado para la validación de especificaciones de sistema.
- Operacional: prototipo iterativo que es progresivamente refinado hasta que se convierte en el sistema final.

Horizontal vs. Vertical vs. Diagonal

- Horizontal: prototipo que modela muchas características de un sistema pero con poco detalle. Dicho detalle alcanzará una profundidad determinada, va a resultar especialmente útil en las etapas tempranas de diseño y tiene como objetivo el test del modo de interacción global, al contemplar funciones comunes que el usuario va a utilizar frecuentemente.
- Vertical: prototipo que modela pocas características de un sistema pero con mucho detalle. Va a resultar especialmente útil en etapas más avanzadas del diseño y tiene como objetivo el test de detalles del diseño.
- Diagonal: prototipo horizontal hasta un cierto nivel, a partir del cual se puede considerar vertical.

Global vs. Local

- Global: prototipo del sistema completo. Prototipo horizontal expandido que modela una gran cantidad de características y cubre un amplio rango de funcionalidades. Va a resultar muy útil a lo largo de todo el proceso de diseño.
- Local: prototipo de un único componente o característica del sistema de usabilidad crítica. Va a resultar de utilidad en algunas etapas específicas del proceso de diseño.[21]

1.5.2 Importancia de un prototipo

Los prototipos son cruciales para diseñar un buen sitio web, facilitan la planificación del proceso de creación, reducen el coste de las evaluaciones, aumentan su efectividad y evitan graves errores en el diseño. [22]
Además hay circunstancias que requieren la construcción de un prototipo al comienzo del análisis, puesto que el modelo es el único mediante el que los requerimientos pueden ser derivados efectivamente.

Después de haber concluido esta tarea se decide que la propuesta de solución sea un prototipo de alta fidelidad, experimental, operacional y global debido a la argumentación que presentan dichas características. Para mostrarle al cliente la funcionalidad que va a tener el sistema lo más cercanamente posible a la realidad.

Es indispensable antes de proceder a la confección del prototipo realizar un estudio acerca de Internet, la información a través de esta tecnología, una comparación entre aplicaciones web y sitios web, también desarrollar un estudio sobre los lenguajes de programación web, los sistemas gestores de bases de datos, la arquitectura cliente-servidor y algunas herramientas útiles.

1.6 Tecnologías actuales, lenguajes y herramientas

1.6.1 Internet

Internet es el uso de la tecnología de Conmutación de Paquetes propicia que todas las computadoras estén al mismo nivel en la red, lo que descentraliza su control, cumpliéndose así el objetivo de evitar la existencia de una computadora central.

Para que todas estas computadoras puedan comunicarse entre sí, debe existir un medio físico que las una (pares de cobre, enlaces satelitales, infrarrojos, microondas, fibra óptica, etc.). Además deben “ponerse de acuerdo” en cuanto al lenguaje que utilizarán para comunicarse, lo que técnicamente se conoce como Protocolo de Comunicación.

Los protocolos de comunicación son conjuntos de reglas que rigen la forma de comunicación entre máquinas. Al igual que los humanos usamos ciertas reglas para comunicarnos, el mundo de los unos y los ceros de las computadoras también las tienen. Cada unas de esas formas es llamada protocolo de

comunicación. Un conjunto de esos protocolos está encapsulado en el TCP/IP (Transmission Control Protocol/ Internet Protocol), los que fueron adoptados por decisión de DARPA en 1982 como el conjunto de protocolos estándar para todas las computadoras conectadas a ARPANet. Decisión gracias a la cual se convirtieron en el lenguaje de comunicación de Internet.

1.6.2 La información a través de Internet. El World Wide Web

World Wide Web (WWW), o simplemente Web, es el universo de información accesible a través de Internet, una recopilación universal del conocimiento humano. Es un sistema de información global, interactivo, dinámico, distribuido, gráfico, basado en Hipertexto, con plataforma de enlaces cruzados, que se ejecuta en Internet.

El componente más usado en Internet es definitivamente la Web. Su característica sobresaliente es el texto remarcado, un método para referencias cruzadas instantáneas. Usando la Web, se tiene acceso a millones de páginas de información en todo el mundo. La exploración se realiza por medio de un software especial denominado “Browser” o Navegador. La apariencia de un Sitio Web puede variar ligeramente dependiendo del navegador que se utilice. Las versiones más recientes de estos navegadores incorporan funcionalidades adicionales para permitir la reproducción de animaciones, realidad virtual, sonido y video en formato digital. El protocolo que se utiliza para la comunicación en la Web es el http (Hypertext Transfer Protocol) y el formato que se utiliza para la transferencia es el HTML (Hypertext Markup Language).

1.6.3 Aplicaciones web vs Sitios web

Las aplicaciones Web se desarrollan como una extensión de los Sistemas Web para agregar funcionalidad de negocio al proceso. En términos más simples, una Aplicación Web es un Sistema Web que permite a los usuarios ejecutar lógica de negocio a través de un Navegador (Browser), o lo que es lo mismo, modificar el estado del negocio.

Las Aplicaciones Web utilizan las tecnologías existentes para generar contenidos dinámicos y permitir a los usuarios del sistema modificar la lógica del negocio en el servidor. Si no existe lógica de negocios en el servidor, el sistema no puede ser considerado una aplicación Web, en ese caso se considera como un sitio Web. En esencia una aplicación Web usa un sitio Web como entrada (front-end) a una aplicación típica.

La arquitectura de un Sitio Web es simple. Contiene como componentes principales: el Servidor Web, una Red y uno o más Navegadores o clientes. El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red para lo cual se usa el protocolo HTTP.

El cliente o un navegador (*browser)* es el responsable de mostrar la información al usuario y de hacer validaciones sencillas en la entrada de datos para que la información sea mostrada al usuario.

1.6.4 Lenguajes de programación web.

Los Lenguajes de Programación orientados al Web se clasifican en lenguajes del lado del cliente y lenguajes del lado del servidor.

Entre los lenguajes que trabajan del lado del servidor existen algunos, que se destacan por ser los más sobresalientes como son PERL, ASP, PHP, Java, JSP, los módulos CGIs e ISAPIs, etc. Estos lenguajes desarrollan la lógica de negocio dentro del servidor, además se encargan de los accesos a los distintos Sistemas de Gestión de Bases de Datos. Dentro de los lenguajes que trabajan del lado del cliente se encuentran el JavaScript, XSLT y el Visual Basic Script, estos dos últimos al combinarse con el HTML forman lo que se conoce como DHTML, es decir, salida estándar dinámica o HTML dinámico.

Esta distinción entre los lenguajes ha sido necesaria debido a que el protocolo http es un protocolo sin estado (state less), no guarda información sobre conexiones anteriores y al finalizar la transacción los datos se pierden, cada petición/respuesta es una operación distinta, por lo que la Web trabaja en modo desconectado; o sea, un usuario a través de un navegador hace una petición de una página Web a un Servidor Web (Request), el Servidor obtiene la petición, la procesa y le envía la respuesta al Cliente (Response), éste la recepciona y se desconecta.

- **PERL**

Es un lenguaje de programación muy utilizado para construir aplicaciones CGI para el web. Perl es un acrónimo de Practical Extracting and Reporting Languaje, que viene a indicar que se trata de un lenguaje de programación muy práctico para extraer información de archivos de texto y generar informes a partir del contendido de los ficheros.

Es un lenguaje libre de uso, eso quiere decir que es gratuito. Antes estaba muy asociado a la plataforma Uníx, pero en la actualidad está disponible en otros sistemas operativos como Windows.

- **ASP**

ASP (Active Server Pages) es la tecnología desarrollada por Microsoft para la creación de páginas dinámicas del servidor. ASP se escribe en la misma página web, utilizando el lenguaje Visual Basic Script o Jscript (Javascript de Microsoft).

Con las ASP se puede realizar muchos tipos de aplicaciones distintas. Permite acceso a bases de datos, al sistema de archivos del servidor y en general a todos los recursos que tenga el propio servidor. La mayor desventaja de este lenguaje es que solo se puede implementar sobre los Servidores Web de su desarrollador: Microsoft. Actualmente se ha presentado ya la segunda versión: el ASP.NET, que comprende algunas mejoras en cuanto a las posibilidades del lenguaje y rapidez con la que funciona. ASP.NET tiene algunas diferencias en cuanto a la sintaxis con el ASP, de modo que tienen formas distintas de utilizarse. Para implementarlo es necesario montar en el Servidor la Plataforma.NET

- **PHP**

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es un lenguaje para programar scripts del lado del servidor, que se incrustan dentro del código HTML. Este lenguaje es gratuito y multiplataforma. PHP se escribe dentro del código HTML, lo que lo hace realmente fácil de utilizar. Algunas de sus ventajas son su gratuidad, independencia de plataforma, rapidez y seguridad.

Algunas de las más importantes capacidades de PHP son: compatibilidad con las bases de datos más comunes, como MySQL, MSSQL, mSQL, Oracle, Informix, y ODBC, por ejemplo. Incluye funciones para el envío de correo electrónico, upload de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red, por poner dos ejemplos.

Algunas de las más importantes capacidades de PHP son: compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, y ODBC, por ejemplo. Incluye funciones para el envío de correo electrónico, upload de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

Resumiendo, el PHP corre en 7 plataformas, funciona en 11 tipos de servidores, ofrece soporte para unos 20 Gestores de Bases de Datos y contiene unas 40 extensiones estables sin contar las que se están experimentando, además de que:

- Simplicidad. Su sintaxis está inspirada en C, ligeramente modificada para adaptarla al entorno en el que trabaja, de modo que si se está familiarizado con esta sintaxis, le resultará muy fácil aprender PHP.

- Si bien es cierto que hay ciertas características avanzadas que presentan las plataformas J2EE o .NET y que PHP no las tiene, no todas las aplicaciones Internet ameritan tal grado de complejidad. PHP fácilmente puede cubrir más del 75% de las necesidades del mercado.

- Es multiplataforma, es decir, puede ser utilizado en cualquiera de los principales sistemas operativos del mercado actual y es soportado por la mayoría de los servidores Web.

- Es software libre, lo que implica menos costos y servidores más baratos, por lo que se puede usar en proyectos comerciales si se quiere, sin tener que pagar por su licencia. El tiempo, es uno de los costos más altos que hay que tener en cuenta antes de empezar un proyecto. Para empezar, el tiempo de aprendizaje de PHP es muy corto gracias a su simplicidad. Luego, el tiempo de desarrollo es también corto. Podríamos hacer más de un proyecto Web con PHP en el mismo tiempo que tomaría hacer un solo proyecto con Java o .NET. Otro aspecto que hay que tener en cuenta es el del hardware. Para desarrollar en PHP no se requiere tener grandes capacidades de hardware, como sí lo requieren los pesados IDEs para programar en Java o .Net. Luego, en el caso de los servidores, una aplicación en PHP no requiere tanta memoria de máquina como podría requerir una aplicación en Java con sus servidores de aplicaciones que podrían requerir hasta varios procesadores y varios Gigas de memoria RAM.

- Es muy rápido. Su integración con la base de datos MySQL y el servidor Apache, le permite constituirse como una de las alternativas más atractivas del mercado.

- Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si se está familiarizado con esta sintaxis, le resultara muy fácil aprender PHP.

- Su librería estándar es realmente amplia, lo que permite reducir los llamados " costos ocultos", uno de los principales defectos de ASP.

PHP tiene una de las comunidades más grandes en Internet, por lo que es fácil ((con lo que no es complicado)) encontrar ayuda, documentación, artículos, noticias, y demás recursos.

Desventaja:

La legibilidad de código puede verse afectada al mezclar sentencias HTML y PHP.

- **JSP**

JSP es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es, pues, una tecnología orientada a crear páginas web con programación en Java.

Con JSP se pueden crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP se pueden escribir con el editor HTML/XML habitual.

- **Java Script**

Javascript es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web.

Con Javascript se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.

- **XSLT**

XSLT es el acrónimo de Extensible Stylesheet Languaje Transformation, es un lenguaje que se usa para convertir documentos XML en otros documentos XML; puede convertir un documento XML que obedezca a un DTD a otro que obedezca otro diferente, un documento XML bien formado a otro que siga un DTD, o, lo más habitual, convertirlo a "formatos finales", tales como WML (usado en los móviles WAP) o XHTML.

- **AJAX**

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la velocidad de interacción en la misma.

AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente y que se muestra a continuación: [23]

XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.

Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.

El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto IFrame en lugar del XMLHttpRequest para realizar dichos intercambios.

XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Ventajas del AJAX:

Interactividad

Las aplicaciones AJAX se ejecutan en la máquina del usuario, manipulando la página actual dentro de sus navegadores usando métodos de Document Object Model. Puede ser usado para multitud de tareas como actualizar o eliminar registros, expandir formularios web, devolver peticiones simples de búsqueda, o editar árboles de categorías; todo sin tener la necesidad de recargar toda la página de HTML cada vez que se realiza un cambio. Generalmente solo requiere enviar pequeñas peticiones al servidor, y se devuelven respuestas relativamente cortas. Esto permite el desarrollo de aplicaciones interactivas con más interfaces de usuario más responsivas gracias al uso de las técnicas DHTML.

Portabilidad

Las aplicaciones construidas con AJAX utilizan características bien documentadas presentes en todos los navegadores importantes en la mayoría de las plataformas existentes. Aunque esta situación podría cambiar en el futuro, en este momento, los usos de AJAX son efectivos entre plataformas. Mientras que la plataforma de AJAX está más restringida que la plataforma de Java, las aplicaciones actuales de AJAX llenan con eficacia la parte de los Java Applets: ampliar el navegador con mini-aplicaciones ligeras.

Desventajas del AJAX:

Críticas de usabilidad

Una de las mayores críticas contra el uso de AJAX en aplicaciones web es que puede fácilmente acabar con el comportamiento normal del botón atrás del navegador. Las diversas expectativas entre volver a una página que se ha modificado dinámicamente y la vuelta a una página estática pueden ser sutiles. Otro problema relacionado es que las actualizaciones dinámicas hacen difícil al usuario agregar a los marcadores/favoritos un momento particular de la aplicación.

JavaScript

Aunque AJAX no necesita ningún tipo de plug-in para el navegador, requiere que los usuarios tengan el JavaScript activado. Esto se aplica a todos los navegadores que soportan esta tecnología excepto para Microsoft Internet Explorer 6 y anteriores los cuales necesitan también tener el ActiveX activado, ya que el objeto XMLHttpRequest está implementado junto con el ActiveX en este navegador.

Como ocurre con las aplicaciones DHTML, las de AJAX deben de ser probadas rigurosamente para adaptarse a los diferentes navegadores y plataformas.

- **Java**

El lenguaje para la programación en Java, es un lenguaje orientado a objeto, de una plataforma independiente. El lenguaje para la programación en Java, fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas WEB.

Con la programación en Java, se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una pagina HTML en un servidor WEB, Por lo general los applets son programas pequeños y de propósitos específicos.

1.6.5 Sistemas de Gestión de Bases de Datos (SGBD)

Los SGBD son un conjunto de programas diseñados para la creación, mantenimiento, actualización e integridad de las bases de datos. Estos sistemas también permiten generar nuevas tablas de información a partir de los datos existentes, según las necesidades del usuario. Un SGBD tiene como objetivo:

Un SGBD tiene los siguientes objetivos específicos:

- Independencia de los datos y los programas de aplicación
- Minimización de la redundancia
- Integración y sincronización de las bases de datos
- Integridad de los datos
- Seguridad y protección de los datos
- Facilidad de manipulación de la información

- **SQL Server**

Microsoft SQL Server es un programa informático de gestión y administración de bases de datos relacionales basada en el lenguaje SQL, que incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. Con la aparición de SQL

Server 2005 el mundo de las Bases de datos está cambiando. Los desarrolladores ahora pueden ubicar su código apropiadamente en relación a su funcionalidad, acceder a datos nativos como XML, y construir sistemas complejos que sean manejados por el servidor de Bases de Datos. Estos puntos hacen que el desarrollo de Bases de Datos esté encaminado hacia una integración.SQL Server 2005 es más que un sistema gestor de Bases de Datos ya que incluye múltiples componentes y servicios que la convierten en una plataforma de aplicaciones corporativas.

- **MySQL**

MySQL Database Server es la base de datos de código fuente abierto más usada del mundo. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar. La extensiva reutilización del código dentro del software y una aproximación minimalística para producir características funcionalmente ricas, ha dado lugar a un sistema de administración de la base de datos incomparable en velocidad, compactación, estabilidad y facilidad de despliegue. La exclusiva separación del core server del manejador de tablas, permite funcionar a MyQSL bajo control estricto de transacciones o con acceso a disco no transaccional ultrarrápido.

MySQL Database Server es muy rápido, confiable y fácil de usar. Si eso es lo que usted está buscando, debe tenerlo y usarlo. MySQL Server también tiene un práctico set de características desarrollado en cercana cooperación con nuestros usuarios. MySQL Server fue desarrollado inicialmente para manejar grandes bases de datos mucho más rápidamente que las soluciones existentes y ha sido usado exitosamente por muchos años en ambientes de producción de alta demanda. A través de constante desarrollo, MySQL Server ofrece hoy una rica variedad de funciones. Su conectividad, velocidad y seguridad hacen a MySQL altamente satisfactorio para acceder bases de datos en Internet.

- **POSTGRE SQL**

PostgreSQL posee una amplia licencia BSD (esta licencia básicamente consiste en que el código puede ser redistribuido y modificado. La FSF lo considera, junto con la licencia GPL, Software libre) y ampliamente utilizado. Posee una estabilidad y confiabilidad legendaria nunca ha presentado caídas en varios años de operación de alta actividad. Fue diseñado para ambientes de alto volumen intentando estar a la altura de Oracle, Sybase o Interbase. Escala muy bien al aumentar el número de CPUs y la cantidad

de RAM. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para subselects, triggers, vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos. Sin embargo consume muchos recursos y no escala bien en la plataforma Windows.

Ventajas:

- Soporta transacciones y desde la versión 7.0, llaves foráneas (integridad referencial).
- Soporta un subconjunto de SQL92 MAYOR que el que soporta MySQL.

Inconvenientes:

- Consume bastantes recursos y carga más el sistema.

1.6.6 Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es una forma de dividir las responsabilidades de un Sistema de Información separando la interfaz de usuario (Nivel de presentación) de la gestión de la información (Nivel de gestión de datos).

Esta arquitectura consiste básicamente en que un programa, el Cliente informático realiza peticiones a otro programa, el servidor, que les da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema multiusuario distribuido a través de una red de computadoras.

1.6.7 Macromedia Dreamweaver 8

Macromedia Dreamweaver 8 es un software fácil de usar que permite crear páginas Web profesionales. Las funciones de edición visual de Dreamweaver 8 permiten agregar rápidamente diseño y funcionalidad a las páginas, sin la necesidad de programar manualmente el código HTML. Se puede crear tablas, editar marcos, trabajar con capas, insertar comportamientos JavaScript, etc., de una forma muy sencilla y visual.

Además incluye un software de cliente FTP completo, permitiendo entre otras cosas trabajar con mapas visuales de los sitios Web, actualizando el sitio Web en el servidor sin salir del programa.

1.6.8 Adobe Photoshop

Adobe Photoshop es un programa de edición digital , es una herramienta para retoque, composición fotográfica y animación. Photoshop ofrece cientos de herramientas de una impresionante calidad, con funciones y capacidades que van desde los retoques más básicos a fin de mejorar el color o la luminosidad de una imagen, hasta complicados montajes y transformaciones con lo que modificar completamente el aspecto de una foto. Todo ello sin olvidar el impresionante abanico de filtros y efectos especiales que también incluye.

1.6.9 Rational Rose

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacob-son), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. El navegador UML de Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el codigo ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vistade Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

Concluido este epígrafe se llega a la conclusión de que para la confección del prototipo se utilizará la Macromedia Dreamweaver 8, por su fácil manejo en la edición de páginas web y por la interfaz amena que le proporciona a éstas, además permite agregar funcionalidades, para así mostrar al usuario el verdadero propósito del sistema. El Rational Rose como modelador visual de la notación UML (Unified Modeling Language) para la confección de los diagramas que se ilustran en este documento, porque esta herramienta es muy completa y ofrece amplias potencialidades; y el Adobe Photoshop CS2 para la edición de imágenes incorporadas en la plantilla utilizada para el prototipo puesto que este programa incluye numerosas funcionalidades que permiten animar y mejorar las imágenes de forma extraordinaria.

Es importante conocer acerca de las aplicaciones que existen en el mundo similares a la que se desea construir, para tomar experiencia sobre los requisitos que a éstas les faltan, ver las ventajas que poseen, como funcionan, etc.

1.7 Estado del arte

En el mundo existen varios sistemas para realizar pruebas de caja negra, específicamente para testeo de aplicaciones web, como son: Apache JMeter, Mercury LoadRunner, OpenSTA, JUnit y Suite etest de Empirix, entre otras.

El Apache JMeter es una aplicación 100% Java del proyecto Jakarta de Apache, permite definir comportamientos para casos de test y medir su rendimiento. Válido para contenido estático y dinámico (ficheros, Servlets, scripts de Perl, objetos Java, bases de datos y queries, FTP,...). Puede simular una gran carga en el servidor, HTTP y FTP testing y bases de datos mediante JDBC, multithreading y con grandes facilidades para extender su funcionalidad mediante plugins.

El Mercury LoadRunner ayuda a preveer costosos problemas de rendimiento, cuellos de botella antes de que una aplicación web sea actualizada o sacada a producción. Soporta un gran número de entornos como Web Services, .NET y J2EE. Se pueden obtener gráficas de rendimiento, saber si una actualización cumple con ciertos requerimientos prefijados e identifica y elimina cuellos de botellas encontrados en el ciclo de vida de la aplicación.

El Open STA no se trata de una herramienta específica, pero si una colección de herramientas que usando en una arquitectura distribuida basada en CORBA, realiza testeos a aplicaciones webs. Se requiere conocimiento de HTTP y de la aplicación en la que se está trabajando, por lo que no es sencillo su uso. No cumple ninguna metodología de testing, sino que es un sistema flexible para realizar testings y obtener datos.

JUnit es un conjunto de librerías creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. Es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor

esperado sea diferente al que regresó el método durante la ejecución devolverá un fallo en el método correspondiente.

JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

Por último está el Suite eTest de Empirix, esta herramienta permite de un modo sencillo y visual grabar secuencias de navegación, que pueden representar casos de uso de nuestro negocio, y que después se puede utilizar, gracias al resto de las herramientas de la Suite, para atacar al sistema:

Bajo demanda - Prueba de regresión

Periódicamente – Prueba de disponibilidad

De un modo masivo (pero simulando comportamientos reales) – Velocidad al aumentar el número de usuarios

De modo distribuido y simulando muchos usuarios concurrentes – Prueba de capacidad

Simulando muchos usuarios realmente concurrentes - Pruebas de sincronismo

Tras realizar la investigación de las aplicaciones existentes en el mundo que apoyan el proceso de la realización de las pruebas de caja negra, se llegó a la conclusión de que ninguna cumple las expectativas deseadas para el prototipo que se desea elaborar, puesto que todas tienen sus inconvenientes y a lo que se aspira con la propuesta de solución es que sea lo más completa posible, fácil de manejar y segura.

1.8 Conclusiones

Finalizado este capítulo se llega a la conclusión de que el estudio realizado sobre los temas mencionados en la introducción del mismo fue vital para poder comprender el proceso que va a simular la solución propuesta y para confeccionar la misma, además el estudio de las herramientas posibilitó la selección de éstas.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se dará paso a conocer el objeto de estudio, el problema y la situación problémica, objeto de automatización, información que se maneja, la propuesta de solución, el modelado del negocio, los requisitos funcionales y no funcionales, así como la definición de los casos de uso del sistema.

2.2 Objeto de estudio y Campo de Acción

La UCI es una universidad creada por nuestro Comandante en Jefe Fidel Castro Ruz, que tiene como objetivo la producción de software en Cuba, por lo cual es de vital importancia que esté informada acerca del tema de la calidad en el mismo, así como desarrollar sistemas para automatizar las pruebas que se le hacen a los software, para así brindarle a los compañeros que las realizan un fácil manejo, seguridad y al mismo tiempo al país un avance en la producción de software, puesto que no existe ningún sistema automatizado de este tipo. Por tanto el objeto de estudio es el proceso de Pruebas de Caja Negra. De ello se deriva que el campo de acción es el proceso de pruebas de caja negra en la Dirección de Calidad y Normas de la UCI**.**

2.3 Problema y Situación problémica

Este trabajo nace, como ya se ha dicho, de la carencia de un Sistema que le de solución a las problemáticas que se presentan hoy día en la UCI, por lo que el problema a solucionar consiste en que:

No existe un Sistema automatizado capaz de gestionar las Pruebas de Caja Negra.

Actualmente en la UCI estas pruebas son ejecutadas manualmente lo cual conlleva a una pérdida de información (datos y fechas del probador y el evaluador), dificultad al construir un expediente, es difícil darle seguimiento a un proceso, el control de las no conformidades se dificulta (registro, clasificación y seguimiento), hay problemas de uniformidad en el manejo de plantillas, dificultades en el entrenamiento

del probador y al llevar estadísticas de las personas, proyectos, productos y defectos, se hace imposible llevar métricas, dificulta la estimación del tiempo de pruebas, así como la confección de cronogramas.

Uno de los problemas que se afrontan actualmente en la esfera de la computación es la Calidad del Software. Las prácticas pobres de Ingeniería, la carencia de herramientas de recolección de datos que permitan la aplicación de métricas adecuadas de calidad y las decisiones de los directivos guiadas por una planificación irreal; traen como consecuencia tiempos de pruebas impredecibles, productos con muchos defectos, demoras en la aceptación de los usuarios y una extensa garantía de servicio y reparaciones. Una pobre calidad afecta la planificación y torna ineficiente el proceso de prueba. Las pruebas de calidad surgen ante la necesidad de contrarrestar estos problemas y producir software de alta calidad.

2.4 Objeto de automatización

Los procesos a automatizar son el proceso de medición y evaluación. El Proceso de Medición consta de tres subprocesos: Pre-Medición (Proceso de Convocatoria) donde se selecciona y convoca al Comité Evaluador compuesto en su totalidad por especialistas en los temas tratados y se convoca al Equipo Medidor, Logística (Proceso Logístico) en el cual se determina la parte logística necesaria y se formula un plan de trabajo; y Medición (Proceso de Medición) donde se comienzan a realizar pruebas al software por parte del Comité Técnico evaluador y equipo medidor y concluidas las pruebas se conforma un informe de medición con los resultados alcanzados.

El Proceso de Evaluación contiene dos subprocesos: Evaluación (Proceso de Selección) donde se reúne el Comité Técnico Evaluador y con los datos del informe de medición realiza una valoración completa y emite un resultado y al final del ciclo de convocatorias se otorgara a los mejores productos certificados un Premio a la Calidad y Legalización (Proceso Legal), si el resultado es negativo no se emite el sello, se cierra el expediente y se confecciona un documento con una serie de sugerencias, recomendaciones y no conformidades. Se devuelve la documentación del producto teniendo la posibilidad de presentarse nuevamente y si el resultado es positivo se otorga la certificación, respaldada por las resoluciones, se cierra el expediente y se archiva.

2.5 Información que se maneja

La información que se maneja en estos procesos se divide en documentos de entrada y salida, los documentos de entrada son:

Evidencias de que se llevó a cabo las pruebas de liberación dentro del proyecto; el producto debe ser presentado lo más completo posible, conteniendo en su expediente los aspectos necesarios con que debe contar éste, alineando su contenido con los Lineamientos Mínimos de Calidad (LMC); declaraciones de funcionalidad, sus requisitos en el formato que la metodología usada lo exija; las necesidades de personal para el montaje, instalación y prueba de la solución; las competencias del personal involucrado en los diferentes niveles de la aplicación, todos los tipos de usuarios, y del personal de mantenimiento, satisfaciendo el punto anterior; afirmaciones de calidad del producto respecto a los requisitos internacionales de calidad; forma de comprobar cada una de las declaraciones y afirmaciones; presentar documentación de soporte: manuales de asistencia técnica de los diferentes niveles, manuales de instalación y diagrama de despliegue; presentar documentación para el usuario; presentar la documentación para el cliente atendiendo a la etapa o fase del proyecto y lo pactado en el contrato; evidencias de validación de partes y entregables que componen la solución con el cliente de la misma y la entrega del software y el expediente debe quedar registrada en un acta de entrega que aclare los temas de confidencialidad.

Dentro de los documentos de salida se encuentran: siempre después del último cambio se impone una nueva revisión, responsabilidad de la dirección del proyecto y de la alta dirección de la entidad; es responsabilidad del Grupo de Calidad comunicar formalmente el estado del producto a presentar a las instancias superiores; esto conlleva obligatoriamente a la emisión de un aval del estado del producto por parte de la Facultad: el AVAL necesita de un valor mínimo de No Conformidades para su emisión positiva: la calidad y estabilidad de la solución software debe ser palpable y en su defecto se emite un informe detallado de No Conformidades del producto respecto de los requisitos declarados, Normas y Principios establecidos; para el caso particular de la liberación para la exportación o software médico: se emite un aval y se verifica la completitud del expediente del producto, pero extremando el cuidado y profundidad de las revisiones; para los casos particulares de software de terceros: deben existir evidencias de que se llevó a cabo las pruebas de liberación dentro del proyecto que originó el software y debe tener un aval de

la entidad que libera el producto y se hace responsable de su calidad; los recursos para la liberación son proporcionados por la Facultad.

2.6 Propuesta de solución

Tras la investigación realizada de los diferentes conceptos, procesos que se llevan a cabo y necesidades del cliente se ha decidido realizar la construcción de un prototipo que muestre al cliente la operabilidad del futuro sistema, para que éste apruebe si contiene todos los requisitos y dar paso a la implementación del mismo. El prototipo será realizado en el ambiente real de trabajo simulando el proceso que llevará a cabo el futuro sistema.

2.7 Modelo del Negocio

2.7.1 Descripción

El negocio consiste en que el cliente solicite la realización de pruebas, donde Calisoft gestiona todo el proceso y los probadores son los encargados de ejecutar las pruebas junto al comité de expertos. Además el cliente solicita los resultados de dichas pruebas donde nuevamente Calisoft gestiona todos los trámites y finalmente el comité de expertos emite los resultados.

2.7.2 Representación

2.7.2.1 Actores del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; cono los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

Actores del negocio	**Justificación**
Cliente	Es el principal beneficiado con todo el proceso pues es el que solicita realización y resultados de las pruebas.

Tabla 2.1 Descripción de los actores del negocio

2.7.2.2 Trabajadores

Un trabajador define el comportamiento y las responsabilidades de un individuo que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio.

Trabajadores del negocio	**Justificación**
Coordinador de Calisoft	Es quien coordina todas las actividades, solicita al probador y selecciona el experto o evaluador.
Evaluador	Participa en la evaluación de las pruebas y sus resultados y hace la entrega de los mismos, concilia las no conformidades, resuelve con instancias superiores las dudas.
Probador	Aplica casos de prueba y listas de chequeo ya diseñados, revisa documentación del expediente y emite informes de no conformidades.
Especialista	Liderea el proceso de prueba para un software determinado, diseña el plan de prueba con el caso de prueba incluido, forma a los probadores en el manejo del proceso de prueba.
Suministrador	Entrega el producto con el expediente de éste, monta el ambiente de prueba.

Tabla 2.2 Descripción de los trabajadores del negocio.

2.7.2.3 Diagrama de casos de uso del negocio

Diagrama de Casos de Uso del Negocio

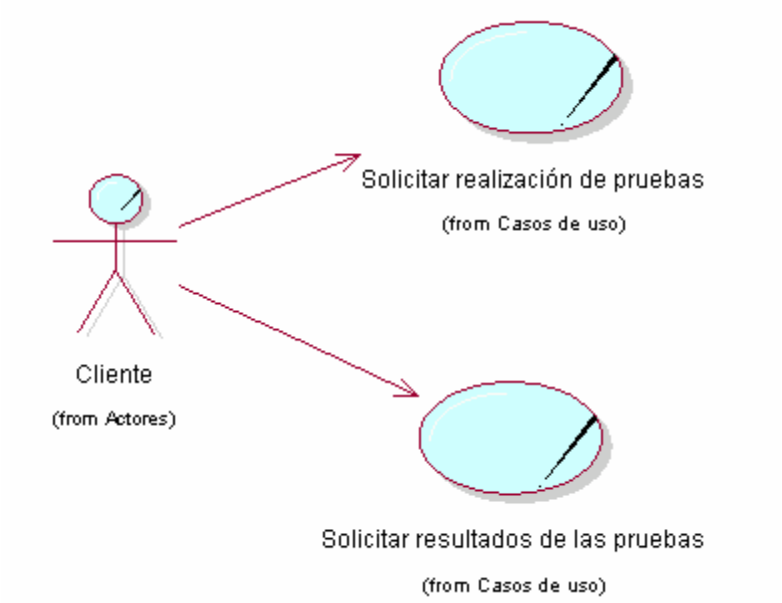


Figura 2.1 Diagrama de casos de uso del negocio.

2.7.2.4 Descripción del Caso de uso: Solicitar realización de pruebas

Caso de uso del negocio: Solicitar realización de pruebas	
Actores del negocio: Cliente	
Propósito: Permitir que el cliente solicite realización de las pruebas al software	
Resumen: El caso de uso se inicia cuando el cliente solicita la realización de pruebas. Luego Calisoft coordina con el probador y el Comité de expertos el plan de pruebas y finalmente se realizan.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1. El cliente solicita la realización de pruebas.	1.1. Calisoft recibe solicitud. 1.2. El coordinador de Calisoft llena la solicitud.

	1.3. Confecciona el registro del cliente. 1.4. Pregunta tipo de prueba.
2. El cliente informa tipo de prueba	2.1 El coordinador de Calisoft confecciona el registro de prueba. 2.2 Confecciona calendario. 2.3 Solicita probadores.
3. El cliente suministra probadores	3.1 El coordinador de Calisoft crea el listado de probadores. 3.2 Selecciona al evaluador. 3.3 Solicita producto y expediente del mismo. 3.4 El suministrador proporciona producto y expediente. 3.5 Calisoft recibe producto y expediente. 3.6 Calisoft informa que se puede montar el ambiente de prueba. 3.7 El suministrador colabora con el montaje del ambiente de pruebas. 3.8 El suministrador informa que todo está listo. 3.9 El coordinador de Calisoft informa que puede diseñarse el plan de prueba. 3.10 El especialista de experiencia recibe la información. 3.11 El especialista de experiencia consulta el registro de prueba. 3.12 Consulta el calendario. 3.13 Diseña el plan de pruebas y forma a

	los probadores. 3.14 Informa que todo está listo. • Informa al probador • Informa al evaluador
Sección Informa al probador	
Acción del actor:	**Respuesta del proceso de negocio:**
	3.14 El probador revisa el expediente del producto. 3.15 Aplica casos de prueba y lista de chequeos ya diseñados. 3.16 Crea expediente de aceptación. 3.17 Emite informes de no conformidades.
Sección Informa al evaluador	
Acción del actor:	**Respuesta del proceso de negocio:**
	3.14 El evaluador realiza la evaluación de las pruebas. 3.15 Revisa no conformidades. 3.16 Chequea probadores que participaron. 3.17 Se registra.
Prioridad Alta	
Otras secciones:	

Tabla 2.3 Especificación textual del caso de uso del negocio “Solicitar realización de las pruebas”

2.7.2.5 Descripción del Caso de uso: Solicitar resultados de las pruebas

Caso de uso del negocio: Solicitar resultados de las pruebas
Actores del negocio: Cliente
Propósito: Permitir que el cliente solicite los resultados de las pruebas
Resumen: El caso de uso se inicia cuando el cliente solicita los resultados de las pruebas. Luego

Calisoft informa al Comité de expertos y éstos proceden a la entrega de los resultados.	
Curso normal de los eventos:	
Acción del actor	**Respuesta del proceso de negocio**
1. El cliente solicita los resultados de la pruebas.	1.1 Calisoft recibe solicitud. 1.2 Consulta si el cliente pidió la realización de las pruebas previamente. 1.3 Si el cliente pidió la realización de las pruebas procede a registrar la solicitud de los resultados. 1.4 Informa al evaluador. 1.5 El evaluador recibe la información. 1.6 Procede a entregar los resultados.
Flujo alterno de los eventos:	
Acción del actor:	**Respuesta del proceso de negocio:**
2. El cliente de retira	1.4 Si el cliente no ha pedido que le realicen pruebas al software, Calisoft le informa.
Prioridad: Alta.	
Otras secciones:	

Tabla 2.4 Especificación textual caso de uso “Solicitar resultado de las pruebas”

2.7.3 Diagramas de Actividades (Ver Anexos 1)

Un diagrama de actividad demuestra la serie de actividades que deben ser realizadas en un proceso del negocio, así como las distintas rutas que se pueden ir desencadenando. Este es dividido en canales, donde cada canal representa el actor que está llevando a cabo la actividad y muestra cómo se utilizan las entidades del negocio.

2.7.4 Diagrama de clases del modelo de objetos.

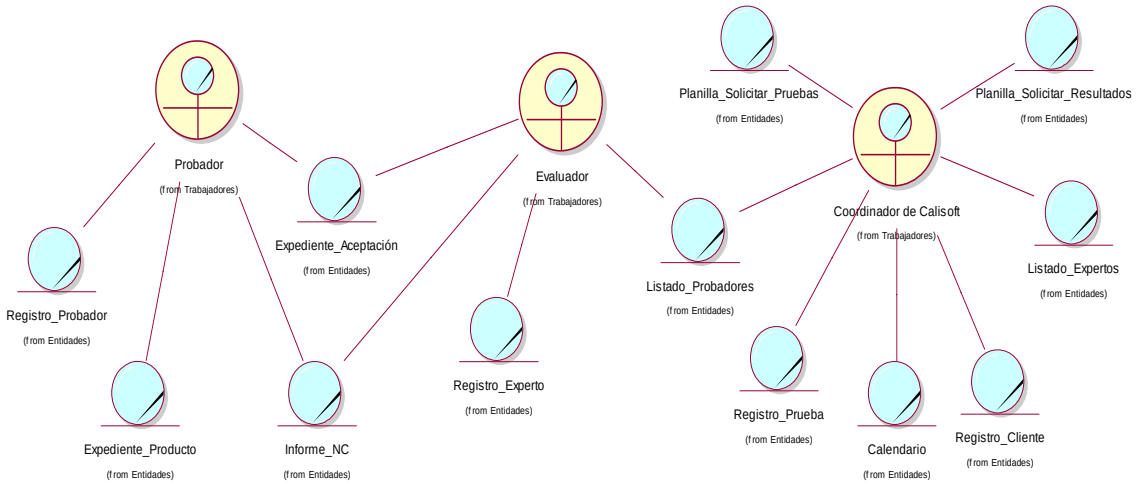


Figura 2.2 Diagrama de clases del modelo objetos.

2.8 Especificación de los requisitos de software

En la captura de requisitos se debe centrar el esfuerzo en reconocer el problema tal y como lo ve el usuario. El propósito principal del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una descripción de los requisitos del sistema suficientemente buena para que pueda llegarse a un acuerdo entre el cliente y a los desarrolladores sobre qué debe y qué no debe hacer el sistema. Los requisitos se clasifican en funcionales y no funcionales. Todo lo antes mencionado se evidencia a continuación.

2.8.1 Requerimientos Funcionales

R1 Administrar Sistema.

1.1 Adicionar usuario.

1.2 Actualizar usuario.

1.3 Eliminar usuario.

1.4 Buscar usuario.

R2 Autenticar Usuario.

2.1 Visualizar formulario.

R3 Cambiar Contraseña.

3.1 Visualizar formulario.

R4 Crear Usuario.

4.1 Insertar datos del usuario.

4.2 Seleccionar si es interno.

4.3 Seleccionar tipo de vivienda.

4.4 Seleccionar tipo de Dirección.

R5 Realizar solicitud de las pruebas.

Insertar datos de la solicitud.

Escoger cantidad de personas.

Escoger estado de la solicitud.

R6 Confeccionar Cronograma.

6.1 Insertar datos.

6.2 Escoger equipo de probadores.

6.3 Escoger nombre de proyecto.

R7 Asignar casos de prueba.

7.1 Seleccionar caso de prueba.

7.2 Seleccionar probador.

7.3 Visualizar asignación.

7.4 Guardar asignación.

R8 Elaborar lista de chequeos.

8.1 Insertar lista de chequeo.

8.2 Guardar lista de chequeo.

R9 Ajustar listas de chequeos.

9.1 Seleccionar nombre del proyecto.

9.2 Seleccionar nivel.

9.3 Seleccionar grado de calidad.

9.4 Seleccionar criterio de evaluación.

R10 Aplicar casos de pruebas.

10.1 Seleccionar caso de prueba.

10.2 Introduce resultado de la prueba.

10.3 Selecciona evaluación.

R11 Gestionar no conformidades.

11.1 Realizar reporte de no conformidades.

11.2 Imprimir informe de no conformidades.

R12 Aplicar listas de chequeos.

12.1 Seleccionar evaluación.

12.2 Seleccionar si procede o no.

12.3 Introducir comentario.

R13 Diseñar casos de prueba.

13.1 Seleccionar nombre del proyecto.

13.2 Introduce datos de la prueba.

13.3 Agrega iteraciones.

R14 Realizar reportes.

14.1 Seleccionar conclusión.

14.2 Imprimir resumen general.

2.8.2 Requerimientos no funcionales

Usabilidad

El sistema desarrollado podrá ser usado por los especialistas de la Dirección de Calidad y Normas de la UCI.

Rendimiento

Se garantiza que la respuesta a solicitudes de los usuarios sea en un periodo de tiempo breve. El sistema deberá ser lo mas estable y confiable posible.

Soporte

Sistema operativo Windows 98 o superior, como mínimo 256 MB de memoria RAM, microprocesador Intel Pentium II o superior.

Seguridad

El sistema se encarga de controlar los diferentes niveles de acceso y funcionalidad de usuarios al sistema, de identificar al usuario antes de que pueda realizar cualquier acción sobre el sistema. Garantiza que la información sea vista únicamente por quien tiene derecho a verla.

Existen tres niveles de accesibilidad, en el primer nivel están las funciones asociadas al coordinador de Calisoft o sea todos los archivos con los que debe interactuar y la preparación de las pruebas; el segundo nivel contiene todo lo necesario para la realización y el proceso de las pruebas; y el tercer nivel está diseñado para confeccionar los reportes.

Legales

El sistema se basa en un estándar que se rige por normas internacionales y cumple con las normas y leyes establecidas en nuestro país.

Confiabilidad

El sistema debe permitir salvar datos.

Interfaz

Es una interfaz amigable y de fácil comprensión.

2.9 Definición de los casos de uso

Los Casos de Uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un Caso de Uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia.

2.9.1 Descripción de los actores del sistema

Los actores del sistema pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado, son parte del sistema, y pueden intercambiar información con él o ser recipientes pasivos de información.

Actores	Justificación
Coordinador de Calisoft	Elabora las listas de chequeo, confecciona el cronograma, asigna productos a los probadores, asigna casos de prueba a los probadores y administra el sistema.
Probador	Encargado de realizar las pruebas, elaborar informe de no conformidades, realizar reportes, ajustar las listas de chequeo y puede ejercer también la aplicación de éstas.
Cliente	Es quien llena la planilla de solicitud.
Usuario	Actor que representa una generalización de todos los actores para autenticarse antes de acceder al sistema y para cambiar contraseña.
Evaluador	Aplica las listas de chequeo.
Especialista	Realiza los diseños de los casos de prueba.
Rol evaluador	Actor que representa una generalización de los actores evaluador y probador, puesto que los dos pueden ejercer el mismo rol, o sea, el rol de evaluador.

Tabla 2.5 Descripción de los actores del sistema

2.9.2 Diagrama de casos de uso

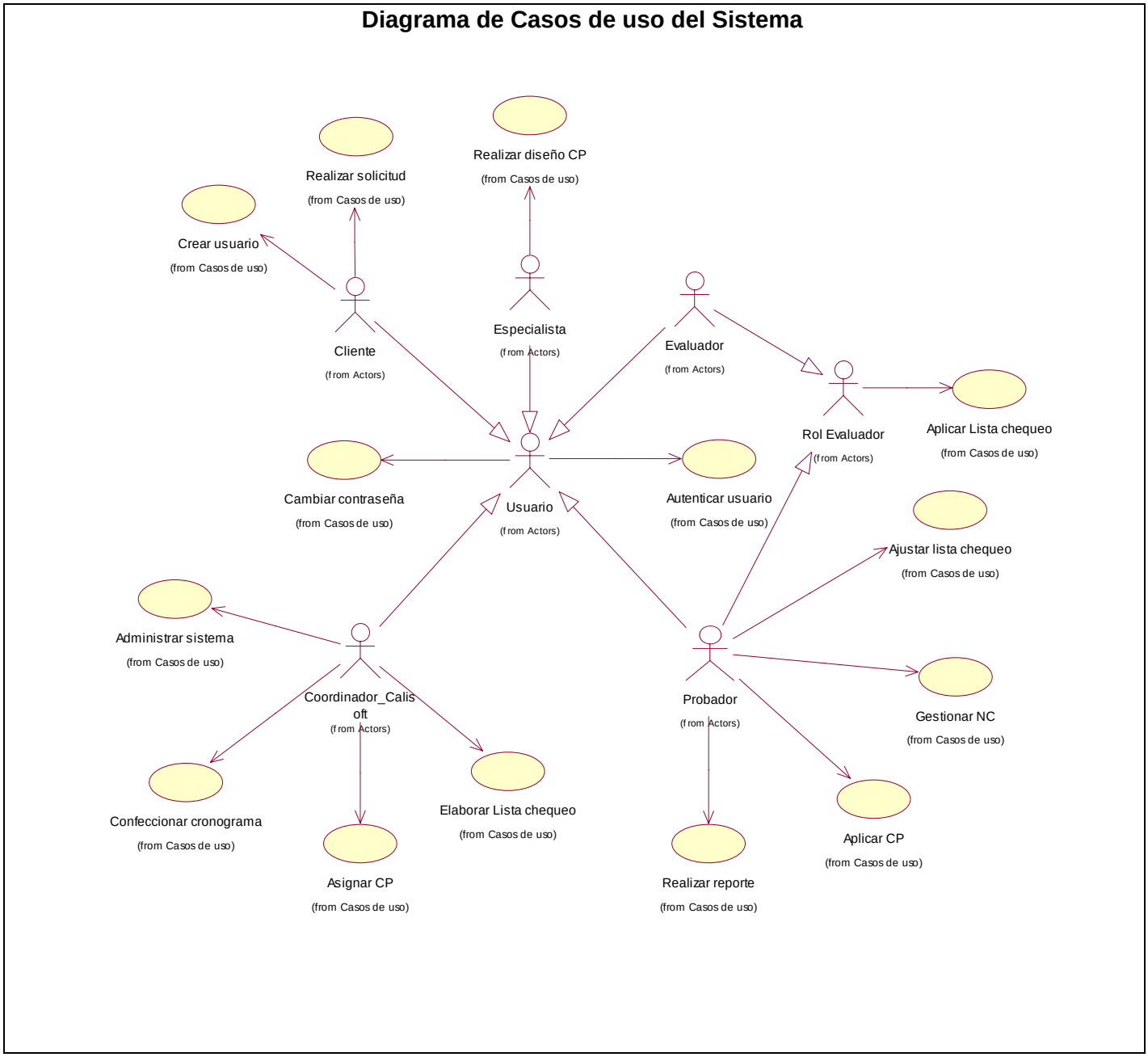


Figura 2.3 Diagrama de casos de uso del sistema

2.9.3 Descripción de los casos de uso

Caso de uso	
CU-1	Autenticar usuario
Propósito	Permitir que los usuarios se autentiquen antes de acceder al sistema
Actores Usuario	
Resumen: El caso de uso se inicia cuando un usuario accede al sistema, en la página de inicio se encuentra un hipervínculo “Autenticar”, seguidamente cuando el usuario lo pincha sale la página correspondiente, introduce los datos y si son correctos accede a su nivel de acceso.	
Referencias	Requisito funcional 2.1.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El usuario va a la página de “Inicio”	1.1 El sistema muestra la página correspondiente.
2. El usuario va a “Autenticar”	2.1 El sistema muestra una página para logearse.
3. El usuario introduce los datos.	3.1 Si los datos son correctos va a su nivel de acceso.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El usuario introduce datos incorrectos.	1.2 Si los datos entrados no son válidos, se emite un mensaje de error para que se introduzcan correctamente el usuario y la contraseña o ambos.

Tabla 2.6 Descripción del caso de uso “Autenticar usuario”

Caso de uso	
CU-2	Administrar sistema
Propósito	Permitir adicionar, eliminar, actualizar y buscar un usuario
Actores Coordinador de Calisoft	
Resumen: El caso de uso se inicia cuando el coordinador de Calisoft accede al sistema y pincha el hipervínculo “Administrar”, le sale la pantalla correspondiente y a continuación puede agregar un nuevo usuario, eliminar un usuario ya existente, actualizar un usuario y buscar por tipo un usuario.	
Referencias	Requisitos funcionales 1.1, 1.2, 1.3, 1.4.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El coordinador va a “Administrar”	1.1 El sistema muestra la interfaz correspondiente.

2. El coordinador de Calisoft escoge una de las operaciones a realizar.	2.1 El sistema ejecuta una de las siguientes acciones. • Si pinchó “Agregar” • Si pinchó sobre un usuario existente para cambiar algún dato. • Si pinchó “Eliminar”
Sección ¨Agregar usuario¨	
3. El coordinador de Calisoft introduce los datos del usuario.	3.1 El sistema verifica que los datos del usuario son válidos y que los obligatorios no estén vacíos. 3.2 El sistema verifica que el usuario no exista. 3.3 El usuario se agrega al sistema.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
	3.1 Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan los datos válidos. 3.2 Se emite un mensaje de error informando la existencia del usuario.
Sección ¨Actualizar usuario¨	
4. El coordinador de Calisoft elige el usuario que desea actualizar.	4.1 El sistema da la posibilidad de cambiar los datos.
5. El coordinador de Calisoft realiza las actualizaciones deseadas.	5.1 El sistema verifica que los datos del usuario son válidos y que los obligatorios no estén vacíos. 5.2 El sistema actualiza la información. 5.3 El sistema muestra el listado de los usuarios.
Flujo alternativo	
	5.1 Se emite un mensaje de error para que se llenen los campos obligatorios o se introduzcan correctamente los datos.

Sección ¨Eliminar usuario¨	
6. El coordinador de Calisoft selecciona dentro del listado de usuarios el usuario que desea eliminar.	6.1 El sistema muestra un mensaje de confirmación. 6.2 El sistema elimina el usuario.
Sección "Buscar usuario"	
7. El coordinador escoge el rol o escribe el nombre del usuario.	7.1 El sistema muestra todos los usuarios con ese rol o muestra el (los) usuario (s) con ese nombre.

Tabla 2.7 Descripción del caso de uso "Administrar sistema"

Caso de uso	
CU-3	Cambiar contraseña
Propósito	Permitir que los usuarios cambien su contraseña
Actores Usuario	
Resumen: El caso de uso se inicia cuando un usuario accede al sistema, en el menú le aparece la opción "Cambiar contraseña", el usuario escoge esta opción, el sistema le muestra la interfaz correspondiente y finalmente el usuario tiene la posibilidad de cambiar su contraseña.	
Referencias	Requisito funcional 3.1
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El usuario escoge "Cambiar contraseña"	1.1 El sistema le da acceso a la página correspondiente.
2. El usuario llena los campos.	2.1 El sistema cambia la contraseña.
3. El usuario introduce los datos.	3.1 Si los datos son correctos va a su nivel de acceso.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El usuario introduce datos incorrectos o deja campos vacíos.	1.2 Si los datos entrados no son válidos, se emite un mensaje de error para que se introduzcan correctamente el usuario y la contraseña o ambos. En caso de que haya dejado campos en blanco el sistema muestra un mensaje avisando.

Tabla 2.8 Descripción del caso de uso "Cambiar contraseña".

Caso de uso	
CU-4	Crear usuario
Propósito	Permitir que el cliente se cree un usuario si nunca a entrado al sistema.
Actores Cliente	
Resumen: El caso de uso se inicia cuando el cliente accede al sistema y escoge la opción “Crear usuario”, el sistema le da acceso al formulario y el cliente llena el mismo y da “Enviar”.	
Referencias	Requisitos funcionales 4.1, 4.2, 4.3, 4.4.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El cliente escoge “Crear usuario”	1.1 El sistema muestra el formulario.
2. El cliente llena el formulario y da “Enviar”	2.1 El sistema lo agrega.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El cliente introduce mal los datos o deja campos vacíos.	1.2 Si los datos entrados no son válidos, se emite un mensaje de error para que se introduzcan correctamente y si el usuario deja campos en blanco sale un mensaje avisando.

Tabla 2.9 Descripción del caso de uso “Crear usuario”

Caso de uso	
CU-5	Realizar solicitud
Propósito	Permitir que el cliente realice la solicitud de pruebas de aceptación
Actores Cliente	
Resumen: El caso de uso se inicia cuando el cliente se autentica, el sistema lo pasa a llenar la planilla de solicitud, el cliente la llena y por último escoge “Aceptar”.	
Referencias	Requisitos funcionales 5.1, 5.2, 5.3.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El cliente se autentica.	1.1 El sistema muestra la planilla de solicitud.
2. El usuario llena la planilla y da “Aceptar”.	2.1 El sistema registra la solicitud.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El cliente introduce datos incorrectos o deja campos sin llenar.	1.2 Si los datos entrados no son válidos, se emite un mensaje de error para que se introduzcan correctamente y si deja campos vacíos se emite un mensaje avisando.

Tabla 2.10 Descripción del caso de uso “Realizar solicitud”.

Caso de uso	
CU-6	Confeccionar cronograma
Propósito	Permitir que el coordinador de Calisoft confeccione el cronograma de un producto.
Actores Coordinador de Calisoft.	
Resumen: El caso de uso se inicia cuando el coordinador de Calisoft escoge “Cronograma” y el sistema le muestra la interfaz correspondiente, el coordinador hace el cronograma y da “Aceptar”.	
Referencias	Requisitos funcionales 6.1, 6.2, 6.3.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El coordinador escoge “Cronograma”.	1.1 El sistema muestra la página correspondiente.
2. El coordinador hace el cronograma y pincha “Aceptar”.	2.1 El sistema registra el cronograma.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El coordinador de Calisoft introduce datos incorrectos o deja campos vacíos.	1.2 Si los datos entrados no son válidos, se emite un mensaje de error para que se introduzcan correctamente, en caso de que deje campos en blanco el sistema muestra un mensaje de aviso.

Tabla 2.11 Descripción del caso de uso “Confeccionar cronograma”

Caso de uso	
CU-7	Asignar caso de prueba
Propósito	Permitir que el coordinador de Calisoft le asigne un caso de prueba al probador.
Actores Coordinador de Calisoft.	
Resumen: El caso de uso se inicia cuando el coordinador de Calisoft escoge “Asignar CP” y el sistema muestra la interfaz correspondiente, el coordinador asigna los casos de prueba a los probadores y pincha “Aceptar”	
Referencias	Requisitos funcionales 7.1, 7.2, 7.3, 7.4.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El coordinador escoge “Asignar CP”.	1.1 El sistema muestra la página correspondiente.
2. El coordinador asigna los casos de prueba a los probadores y da “Aceptar”.	2.1 El sistema registra las elecciones.
Flujo alternativo	
Acciones del actor	Respuesta del sistema

Tabla 2.12 Descripción del caso de uso “Asignar caso de prueba”

Caso de uso	
CU-8	Elaborar lista de chequeos
Propósito	Permitir que el coordinador de Calisoft confeccione la (s) lista (s) de chequeos
Actores Coordinador de Calisoft.	
Resumen: El caso de uso se inicia cuando el coordinador de Calisoft escoge “Elaborar lista chequeo”, el sistema muestra la página correspondiente y el coordinador elabora la lista de chequeos y escoge “Enviar”.	
Referencias	Requisito funcional 8.1
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El coordinador escoge “Elaborar lista chequeo”.	1.1 El sistema muestra la página correspondiente.
2. El coordinador hace la lista de chequeo y da “Enviar”	2.1 El sistema registra la lista de chequeos.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El coordinador de Calisoft introduce datos incorrectos o deja campos vacíos.	1.2 Si los datos entrados no son válidos, se emite un mensaje de error para que se introduzcan correctamente, en caso de que deje campos en blanco el sistema muestra un mensaje de aviso.

Tabla 2.13 Descripción del caso de uso “Elaborar lista de chequeos”.

Caso de uso	
CU-10	Realizar reportes
Propósito	Permitir que el probador realice los reportes.
Actor Probador.	
Resumen: El caso de uso se inicia cuando el probador escoge “Realizar reportes”, el sistema le da acceso a la interfaz correspondiente, el probador realiza sus tareas y finalmente pincha “Enviar”.	
Referencias	Requisitos funcionales 14.1, 14.2.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El probador escoge “Realizar reportes”	1.1 El sistema muestra la página correspondiente.
2. El probador hace sus tareas y da “Enviar”.	2.1 El sistema registra el reporte.
Flujo alternativo	
Acciones del actor	Respuesta del sistema

Tabla 2.14 Descripción del caso de uso “Realizar reportes”.

Caso de uso	
CU-11	Ajustar lista de chequeo
Propósito	Permitir que el probador ajuste la (s) lista (s) de chequeos.
Actor Probador.	
Resumen: El caso de uso se inicia cuando el probador escoge “Ajustar lista” y el sistema muestra la página correspondiente, el probador ajusta la lista de chequeos y pincha “Aceptar”.	
Referencias	Requisitos funcionales 9.1, 9.2, 9.3, 9.4.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El probador escoge “Ajustar lista”.	1.1 El sistema muestra la página correspondiente.
2. El probador ajusta la lista de chequeo y da “Aceptar”.	2.1 El sistema registra la lista de chequeo.
Flujo alternativo	
Acciones del actor	Respuesta del sistema

Tabla 2.15 Descripción del caso de uso “Ajustar lista de chequeo”.

Caso de uso	
CU-12	Aplicar caso de prueba
Propósito	Permitir que el probador aplique caso (s) de prueba al producto.
Actor Probador.	
Resumen: El caso de uso se inicia cuando el probador escoge “Aplicar CP” y el sistema muestra la interfaz correspondiente, el probador llena los campos con los resultados que ha obtenido de las pruebas que le realizó al producto y pincha “Aceptar”	
Referencias	Requisitos funcionales 10.1, 10.2, 10.3.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El probador escoge “Aplicar CP”.	1.1 El sistema muestra la página correspondiente.
2. El probador llena los campos y pincha “Aceptar”.	2.1 El sistema revisa que todos los campos estén llenos y de manera correcta.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El probador deja campos en blanco o los llena de manera incorrecta.	1.2 El sistema en ambos casos muestra un mensaje de aviso.

Tabla 2.16 Descripción del caso de uso “Aplicar caso de prueba”.

Caso de uso

CU-13	Gestionar no conformidades
Propósito	Permitir al probador registrar una no conformidad que haya detectado y elaborar el informe de no conformidades.
Actores Probador	
Resumen: El caso de uso se inicia cuando el probador escoge “No conformidades” y el sistema muestra la interfaz correspondiente, el probador registra la no conformidad detectada y pincha “Aceptar	
Referencias	Requisitos funcionales 11.1, 11.2.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El probador escoge “No conformidades”	1.1 El sistema muestra la interfaz correspondiente.
2. El probador llena los campos.	2.1 El sistema revisa que los campos estén llenados y de manera correcta.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El probador llena los campos de forma incorrecta o deja campos vacíos.	3.1 En ambos casos el sistema emite un mensaje de error.

Tabla 2.17 Descripción del caso de uso “Gestionar no conformidades”.

Caso de uso	
CU-14	Aplicar lista de chequeo
Propósito	Permitir que el evaluador aplique la (s) lista (s) de chequeos.
Actor Rol evaluador	
Resumen: El caso de uso se inicia cuando el evaluador escoge “Aplicar lista chequeo”, el sistema muestra la interfaz correspondiente, el evaluador procede con su tarea y finalmente escoge “Aceptar”	
Referencias	Requisitos funcionales 12.1, 12.2, 12.3.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El rol evaluador escoge “Aplicar lista chequeo”.	1.1 El sistema muestra la página correspondiente.
2. El rol evaluador procede a aplicar la lista de chequeo y escoge “Aceptar”.	2.1 El sistema revisa que todos los campos estén llenos y de manera correcta.
Flujo alternativo	
Acciones del actor	Respuesta del sistema

1. El rol evaluador deja campos en blanco o los llena de manera incorrecta.	1.2 El sistema en ambos casos muestra un mensaje de aviso.

Tabla 2.18 Descripción del caso de uso “Aplicar lista de chequeo”.

Caso de uso	
CU-15	Diseñar casos de prueba
Propósito	Permitir que el especialista diseñe los casos de prueba.
Actor Especialista	
Resumen: El caso de uso se inicia cuando el especialista escoge “Diseñar CP”, el sistema le da acceso a la interfaz correspondiente, el especialista realiza el diseño del caso (s) de prueba y finalmente escoge “Aceptar”.	
Referencias	Requisitos funcionales 13.1, 13.2, 13.3.
Curso Normal	
Acción del actor	**Respuesta del sistema**
1. El especialista escoge “Diseñar CP”.	1.1 El sistema muestra la página correspondiente.
2. El especialista realiza el diseño del caso de prueba y escoge “Aceptar”.	2.1 El sistema revisa que todos los campos estén llenos y de manera correcta.
Flujo alternativo	
Acciones del actor	Respuesta del sistema
1. El especialista deja campos en blanco o los llena de manera incorrecta.	1.2 El sistema en ambos casos muestra un mensaje de aviso.

Tabla 2.19 Descripción del caso de uso “Realizar diseño de caso de prueba”.

2.9 Conclusiones

Hasta aquí, se ha visto como el modelo del negocio ayuda a definir el contexto del sistema y como pueden derivarse los casos de uso a partir de un modelo del negocio. Se pudo apreciar además que los casos de uso se utilizan para capturar los requisitos formándose un modelo de caso de uso del sistema que ayuda a definir un conjunto de funcionalidades afines que el sistema debe cumplir para satisfacer todos los requerimientos funcionales que tiene a su cargo . Por todo lo mencionado anteriormente se pudo elaborar el prototipo de simulación del proceso de pruebas de caja negra. Todo esto marca un patrón para comenzar con el análisis y diseño del sistema.

CAPITULO 3. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN

3.1 Introducción

El objetivo de este capítulo es realizar el análisis y diseño de la solución según los requisitos definidos para una mejor comprensión y que ayude a estructurar el sistema, incluyendo su arquitectura. El mismo contará con varios artefactos que tienen como objetivos mostrar el diseño del sistema propuesto como solución.

3.2 Definición del modelo de análisis.

El análisis consiste en obtener una primera aproximación al modelo de diseño. A pesar de que en el análisis hay un refinamiento de los requisitos, no se toma en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reutilizables de otras aplicaciones, entre otras características que afectan al sistema, ya que el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución.

3.2.1 Diagramas de clases del análisis

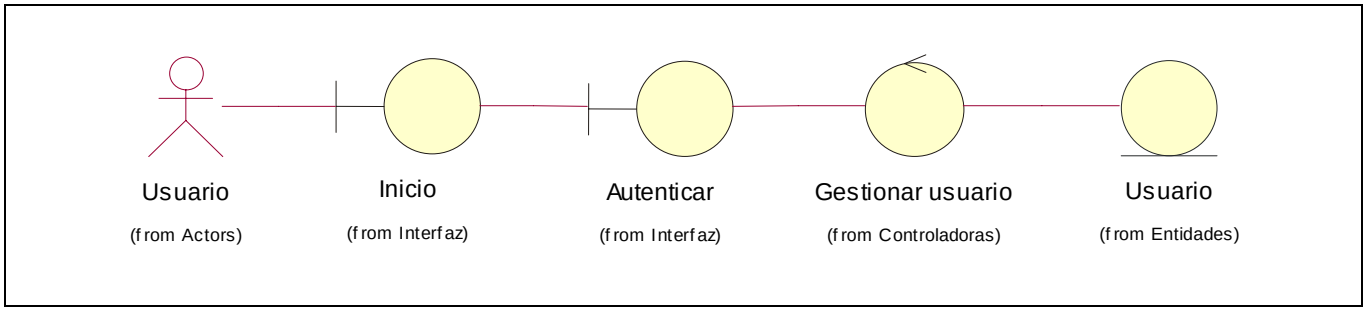


Figura 3.1 Diagrama de clases del análisis del caso de uso" Autenticar usuario"

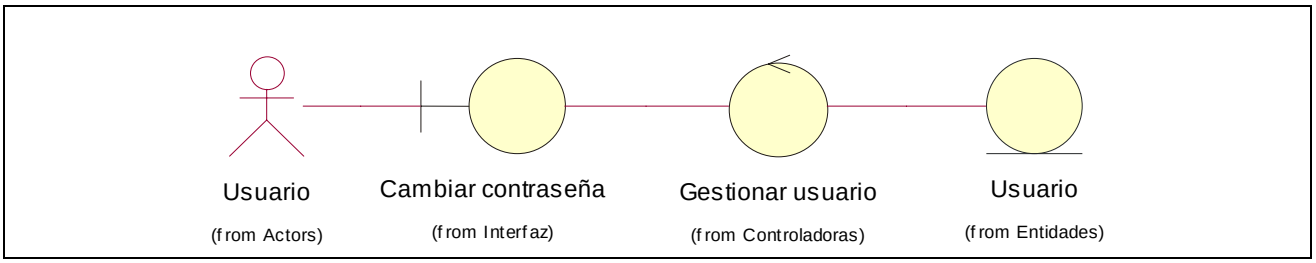


Figura 3.2 Diagrama de clases del análisis del caso de uso “Cambiar contraseña”

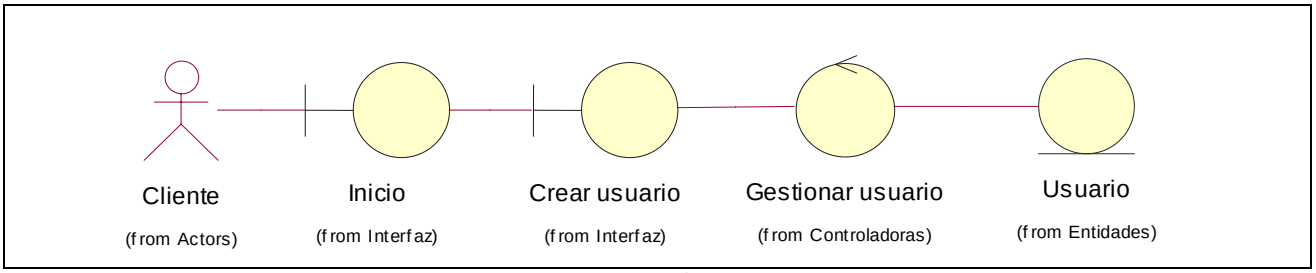


Figura 3.3 Diagrama de clases del análisis del caso de uso “Crear usuario”

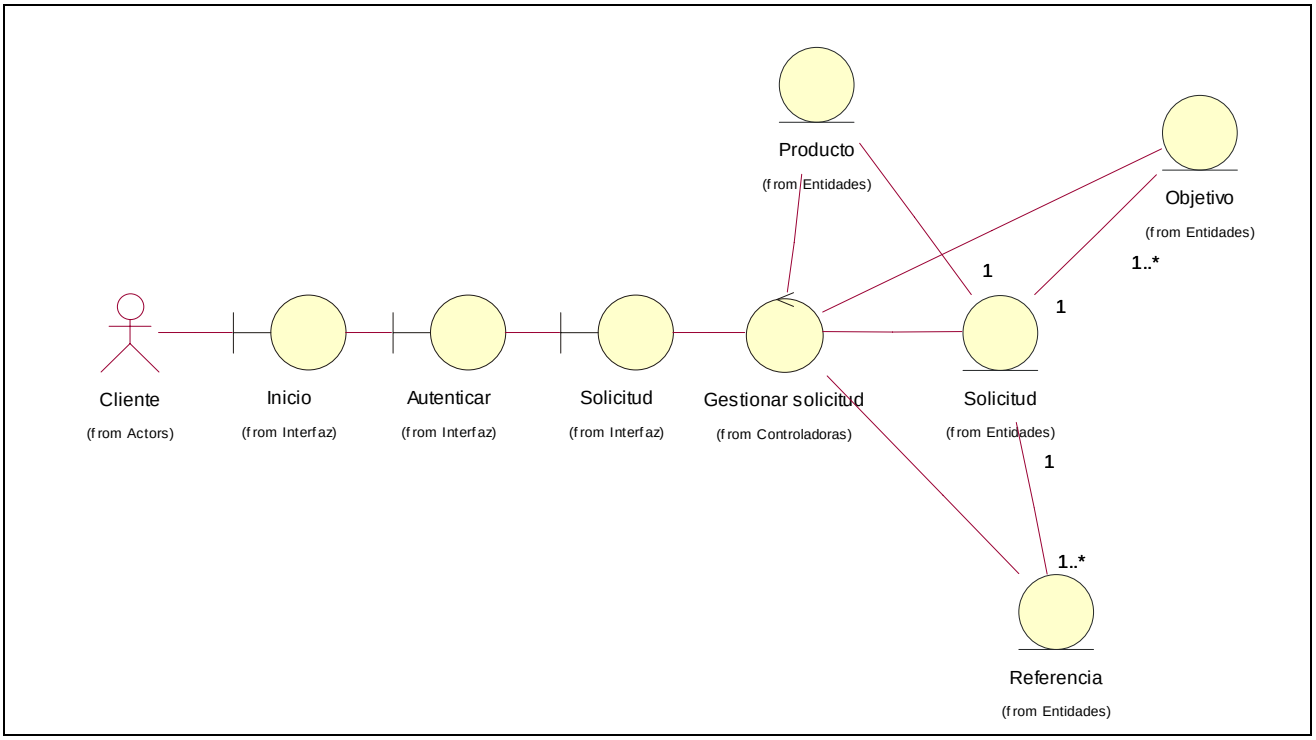


Figura 3.4 Diagrama de clases del análisis del caso de uso “Realizar solicitud”

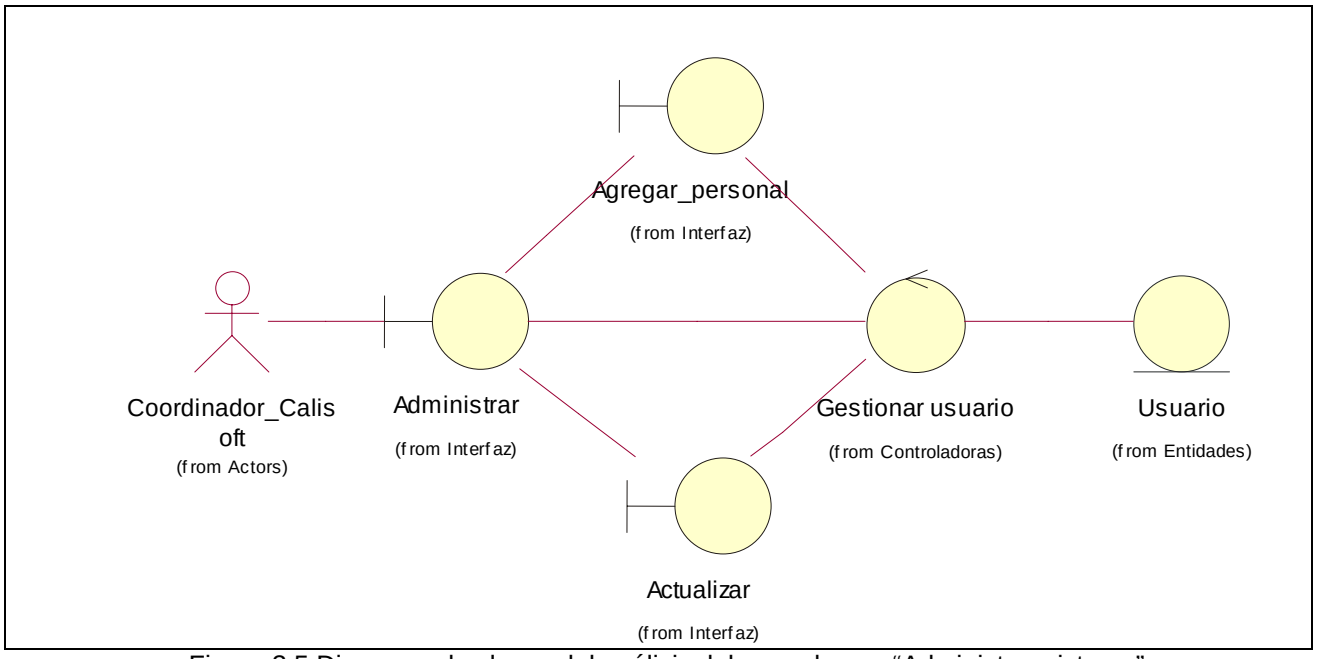


Figura 3.5 Diagrama de clases del análisis del caso de uso “Administrar sistema”

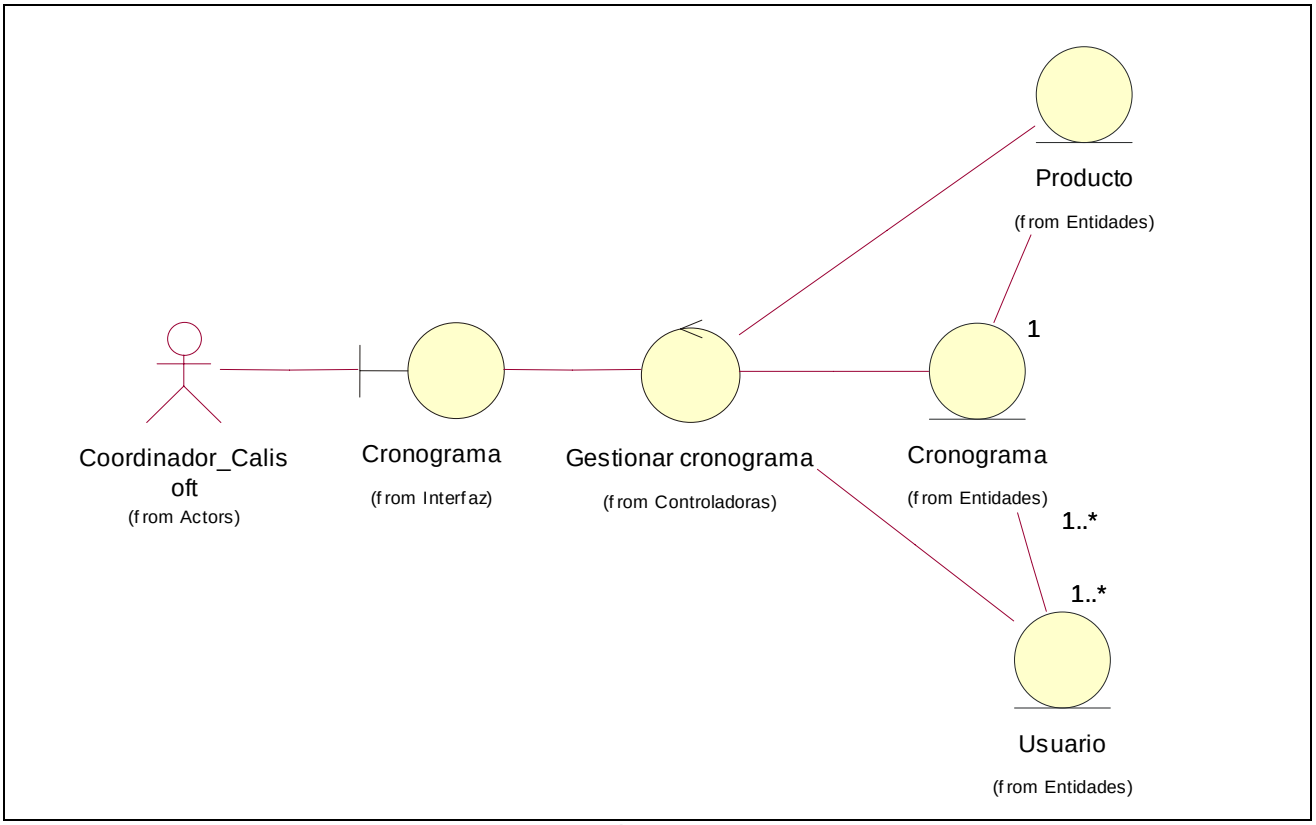


Figura 3.6 Diagrama de clases del análisis del caso de uso “Confeccionar cronograma”

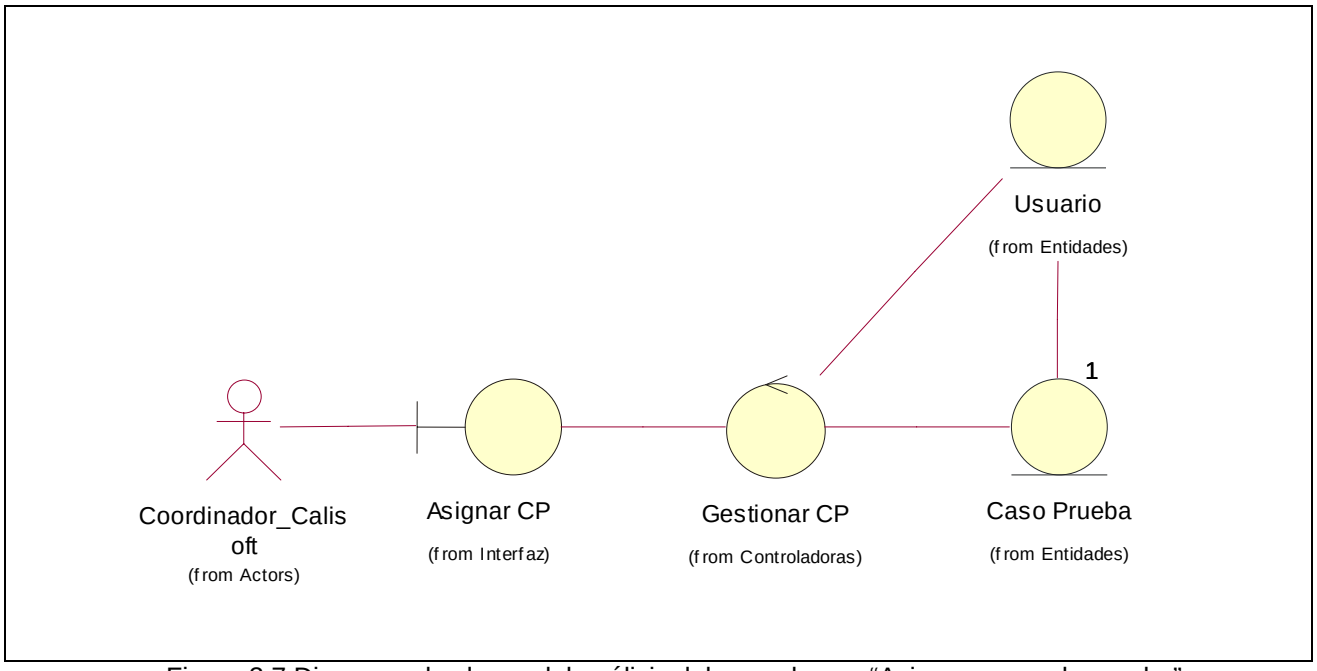


Figura 3.7 Diagrama de clases del análisis del caso de uso “Asignar casos de prueba”.

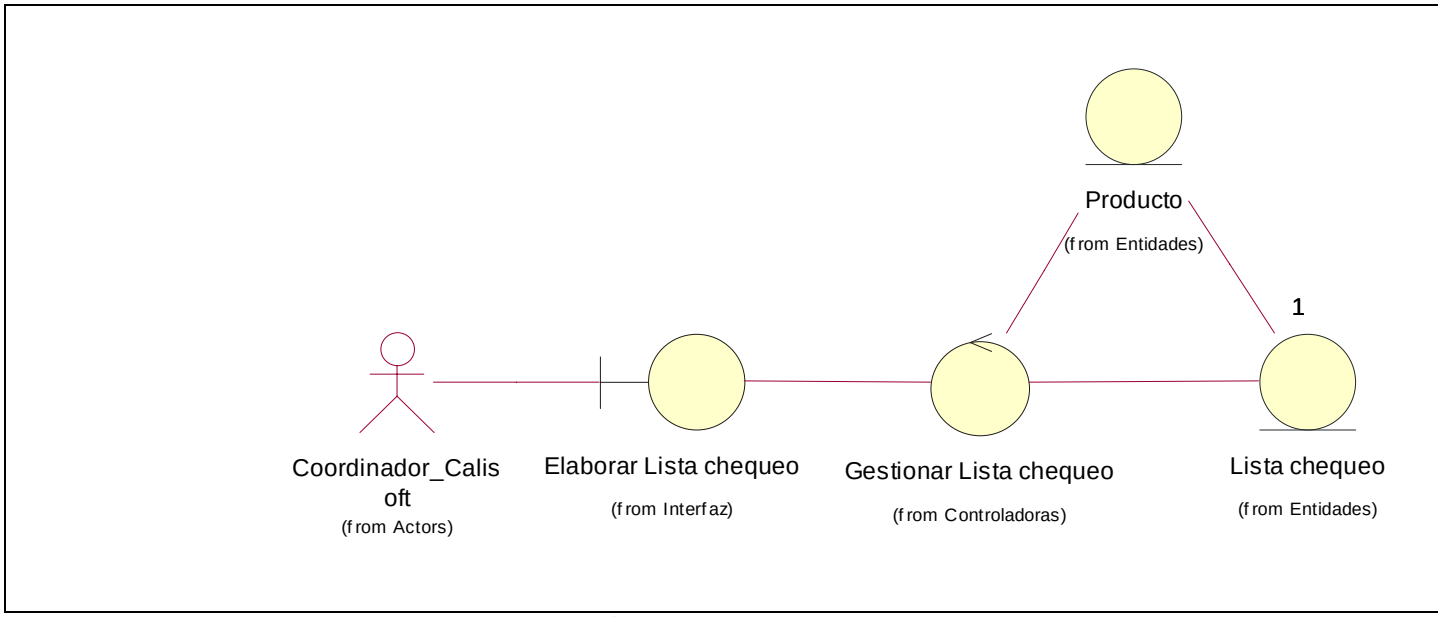


Figura 3.8 Diagrama de clases del análisis del caso de uso “Elaborar lista de chequeo”.

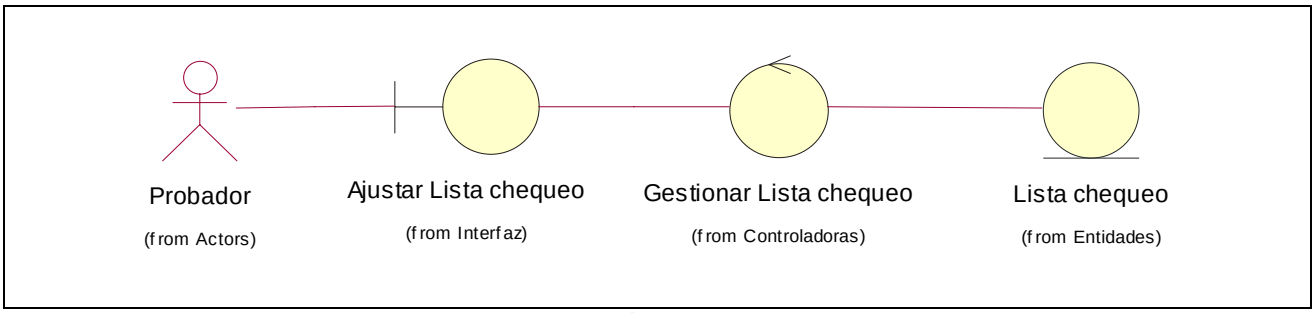


Figura 3.9 Diagrama de clases del análisis del caso de uso “Ajustar lista de chequeo”

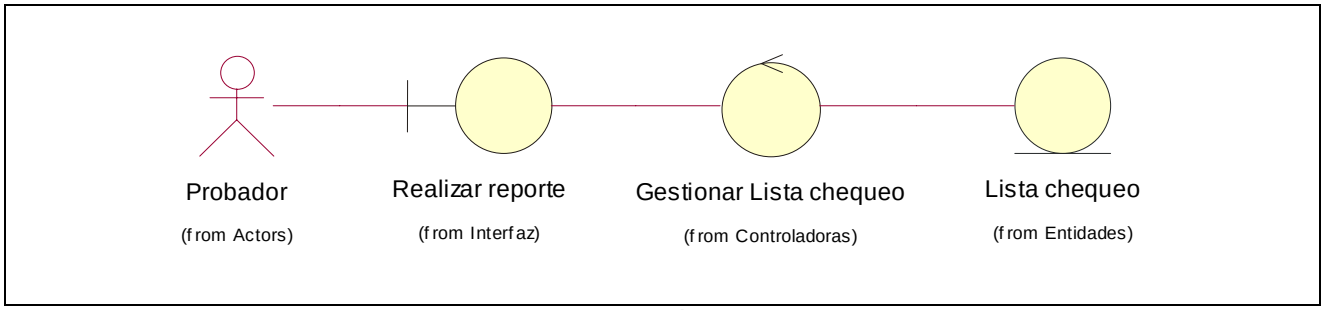


Figura 3.10 Diagrama de clases del análisis del caso de uso “Realizar reporte”

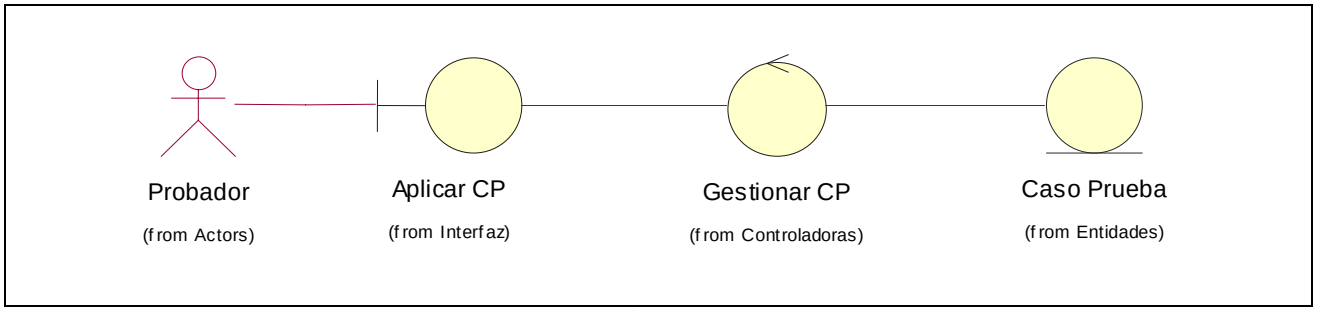


Figura 3.11 Diagrama de clases del análisis del caso de uso “Aplicar casos de prueba”

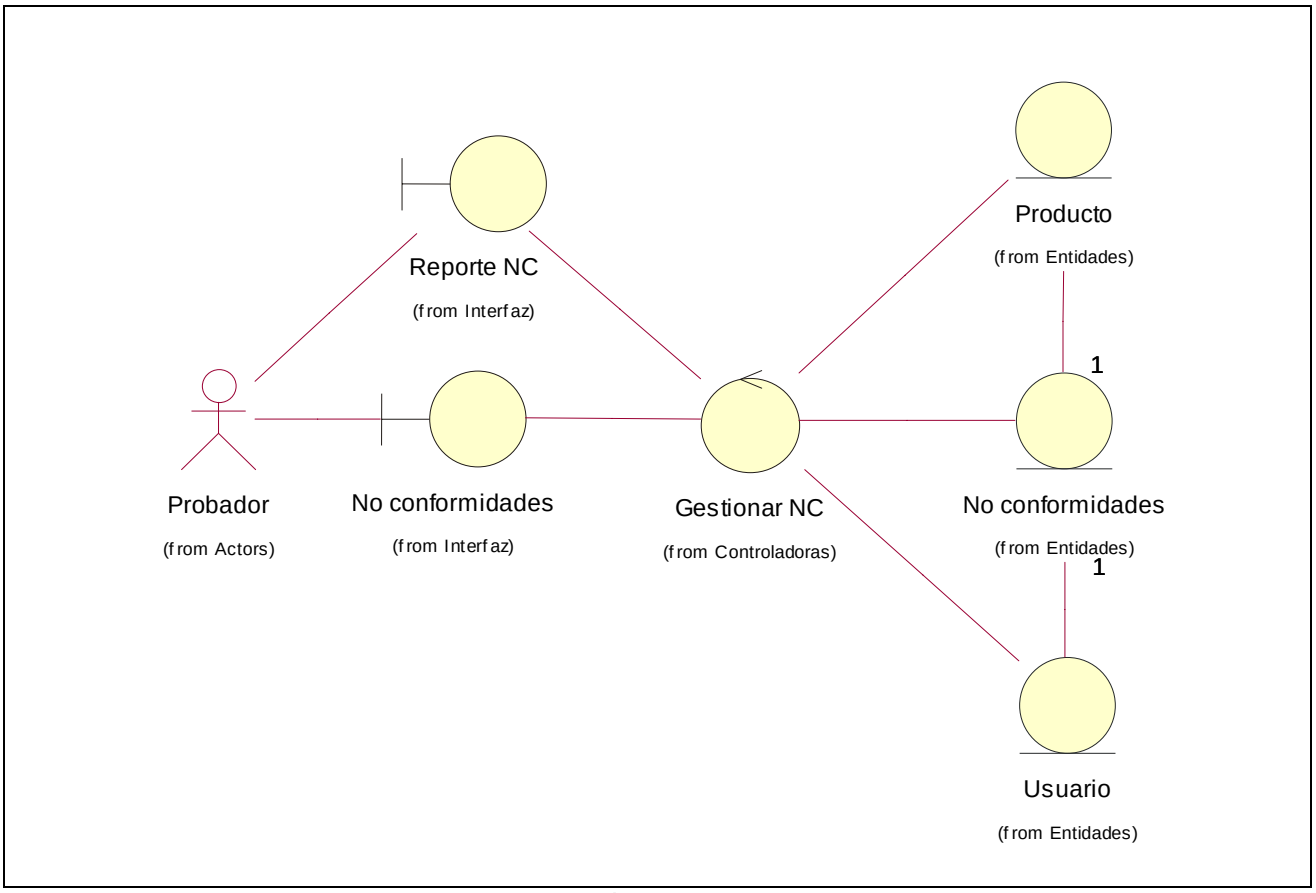


Figura 3.12 Diagrama de clases del análisis del caso de uso “Gestionar no conformidades”

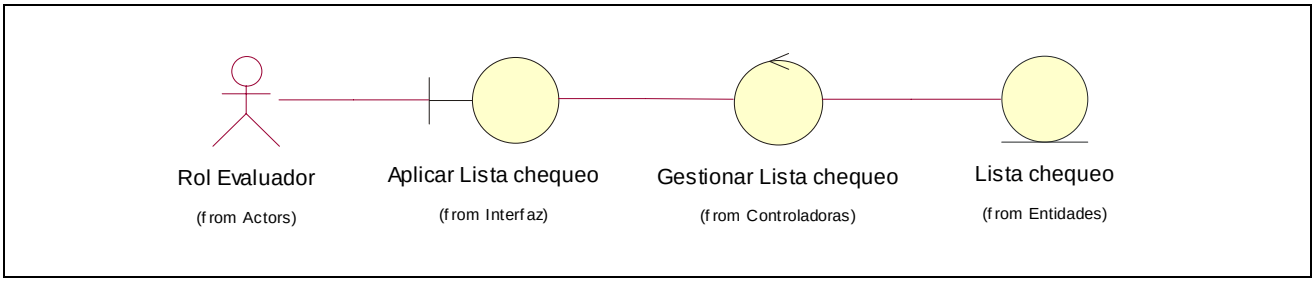


Figura 3.13 Diagrama de clases del análisis del caso de uso “Aplicar lista de chequeo”

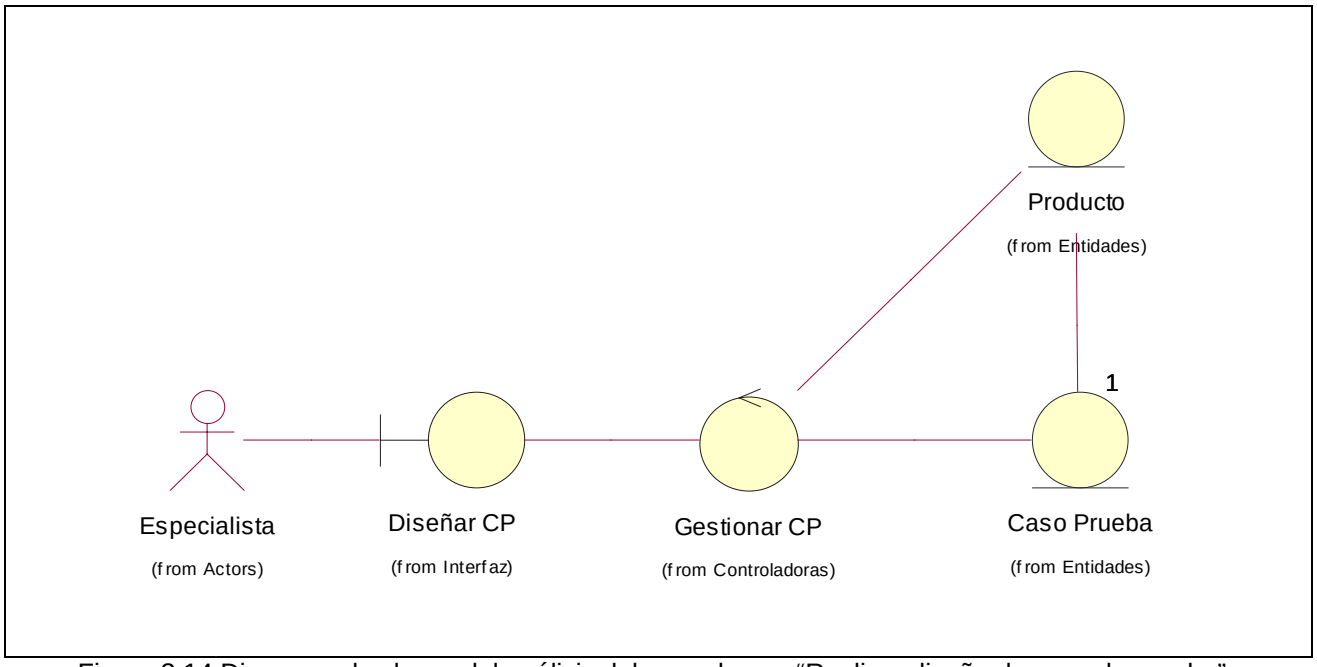


Figura 3.14 Diagrama de clases del análisis del caso de uso “Realizar diseño de caso de prueba”

3.3 Diseño

En el diseño se modela el sistema y encuentra su forma (incluida la arquitectura) 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 se debe conservar lo más fielmente posible cuando se de forma al sistema.

3.3.1 Arquitectura

La arquitectura establece los fundamentos para que analistas, diseñadores, y programadores trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema informático.

No se puede hablar de arquitectura sin mencionar algo muy esencial: los patrones de arquitectura.

3.3.1.1 ¿Qué es un patrón?

Un patrón describe un problema que ocurre una y otra vez en el entorno y describe también el núcleo de la solución al problema, de forma que puede reutilizarse continuamente.

¿A qué responden los patrones de arquitectura?

1. Son esquemas de organización de un sistema.
2. Especifican una serie de subsistemas y sus responsabilidades.
3. Incluyen reglas para organizar las relaciones entre ellos.

Para el desarrollo de la aplicación se propone utilizar la arquitectura de n Capas.

3.3.1.2 Arquitectura de n Capas

La arquitectura de n capas descompone un aplicación en un conjunto de capas independientes y ordenadas jerárquicamente. Cada nivel o capa usa inmediatamente la inferior y ofrece servicios a la capa superior.

Se utilizó la arquitectura de n Capas ya que este patrón es importante porque simplifica la comprensión y organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores.

3.3.1.3 Principios de diseño

El diseño, sea cual sea el objeto del mismo, tiene que basarse en el usuario ya que una navegación engorrosa, puede contribuir a perder posibles clientes-usuarios. A continuación se describen los principios de diseño a seguir para el diseño del sistema en cuestión.

Principios que garantizan la usabilidad de los diseños para aplicaciones Web.

1. Principio Uso equiparable: donde las características de privacidad, garantía y seguridad estén igualmente disponibles para todos los usuarios, y que el diseño sea atractivo para todos los usuarios.

2. Principio de la flexibilidad: donde se ofrezcan posibilidades de elección en los métodos de uso, que facilite al usuario la exactitud y precisión, y se adapte al paso o ritmo del usuario.

3. Principio de tolerancia al error: donde se dispongan los elementos para minimizar los riesgos y errores, por ejemplo utilizando elementos comunes; y los elementos peligrosos eliminados, aislados o tapados, que se proporcionen advertencias sobre peligros y errores. Hay que posibilitar el descubrimiento interactivo y el aprendizaje ensayo-error, y posibilitar la reversibilidad y la recuperabilidad de las acciones.

4. Principio de esfuerzo de acceso y uso: que minimicen las acciones repetitivas, y que proporcione una línea de visión clara hacia los elementos importantes tanto para un usuario sentado como de pie.

3.3.2 Diagramas de interacción (Ver anexos 2)

La vista de interacción describe secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema

3.3.3 Diagrama de clases del diseño (Ver anexos 3)

Diagramas de clases Web

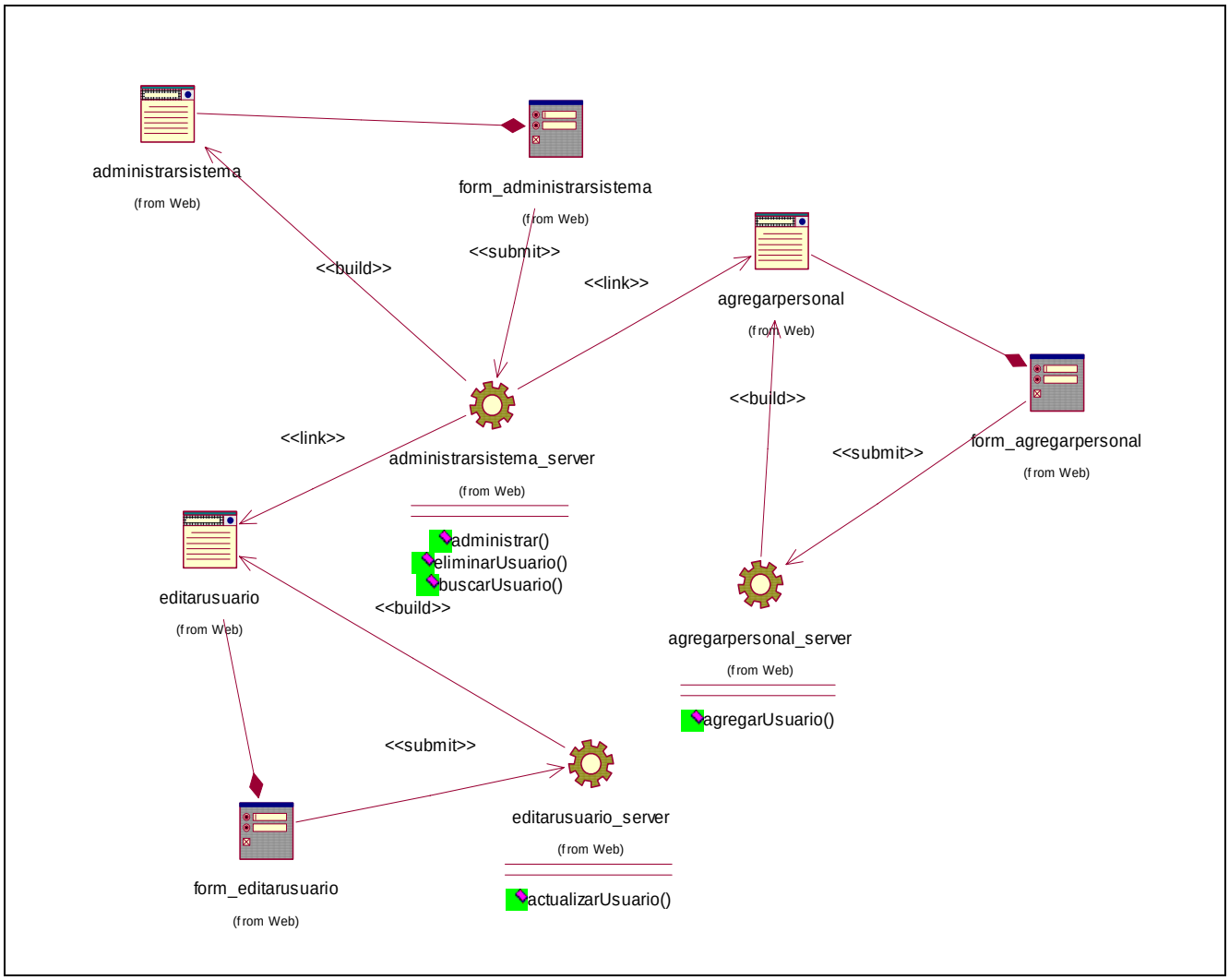


Figura 3.15 Diagrama de clases web para el caso de uso “Administrar sistema”

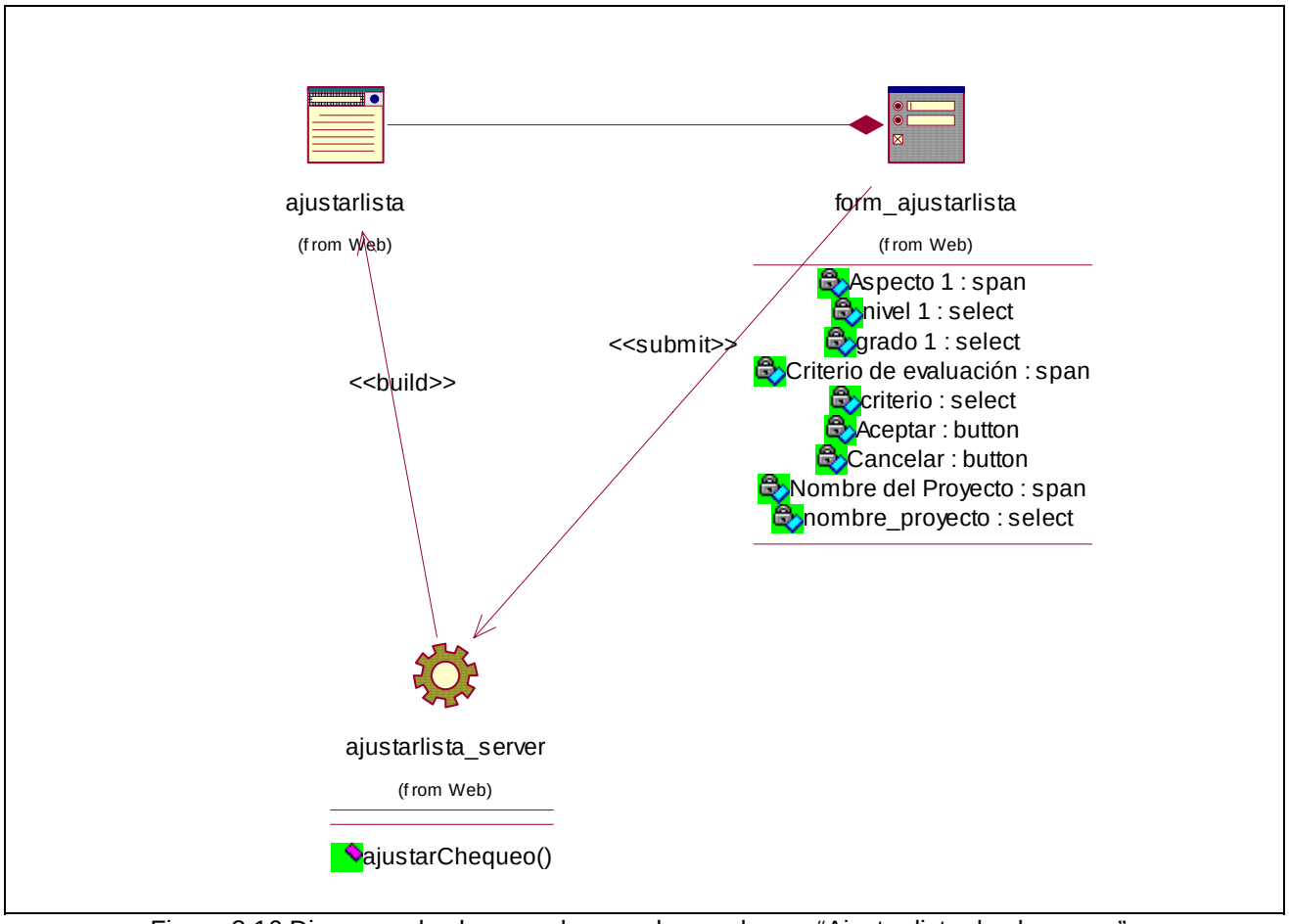


Figura 3.16 Diagrama de clases web para el caso de uso “Ajustar lista de chequeos”

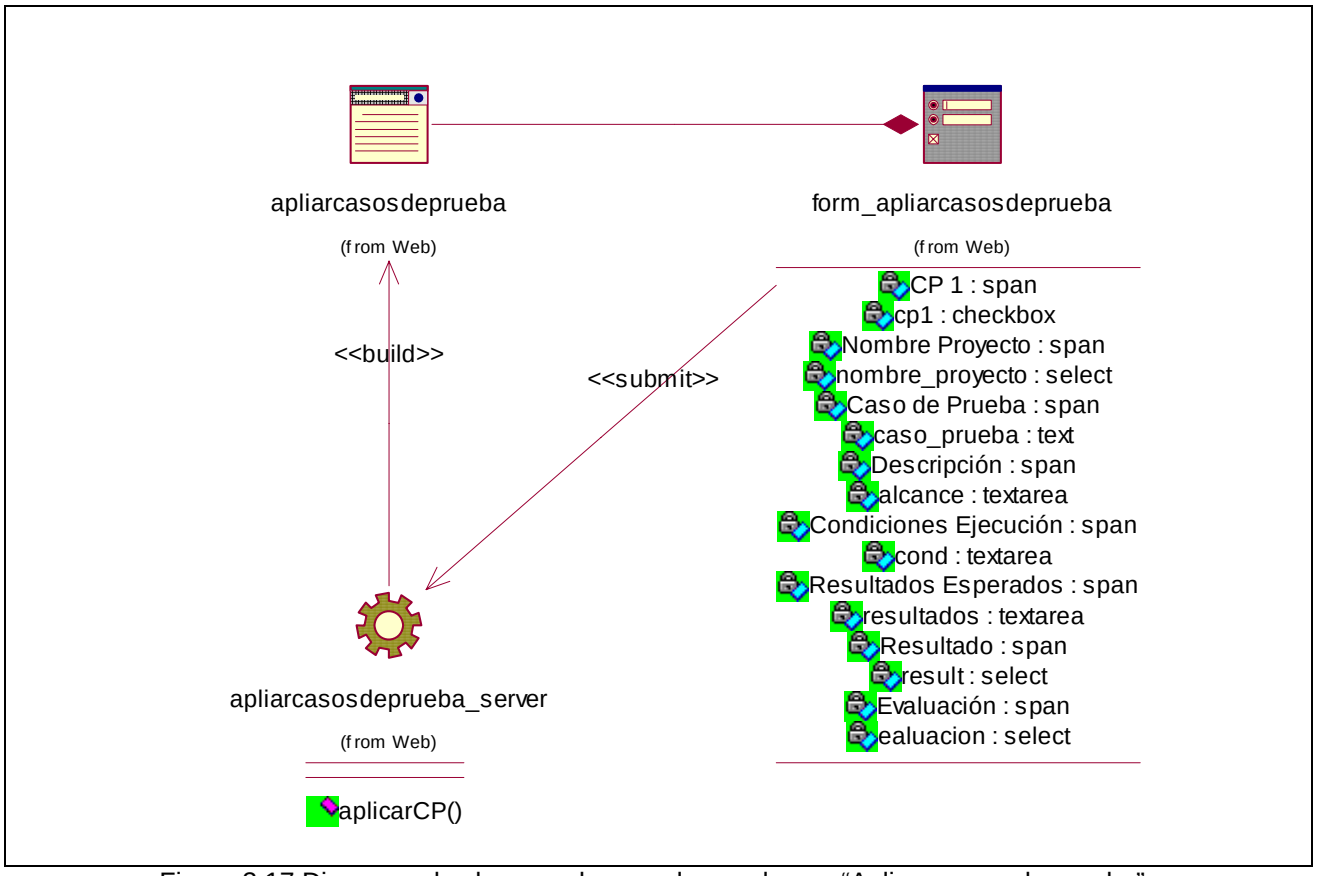


Figura 3.17 Diagrama de clases web para el caso de uso “Aplicar casos de prueba”

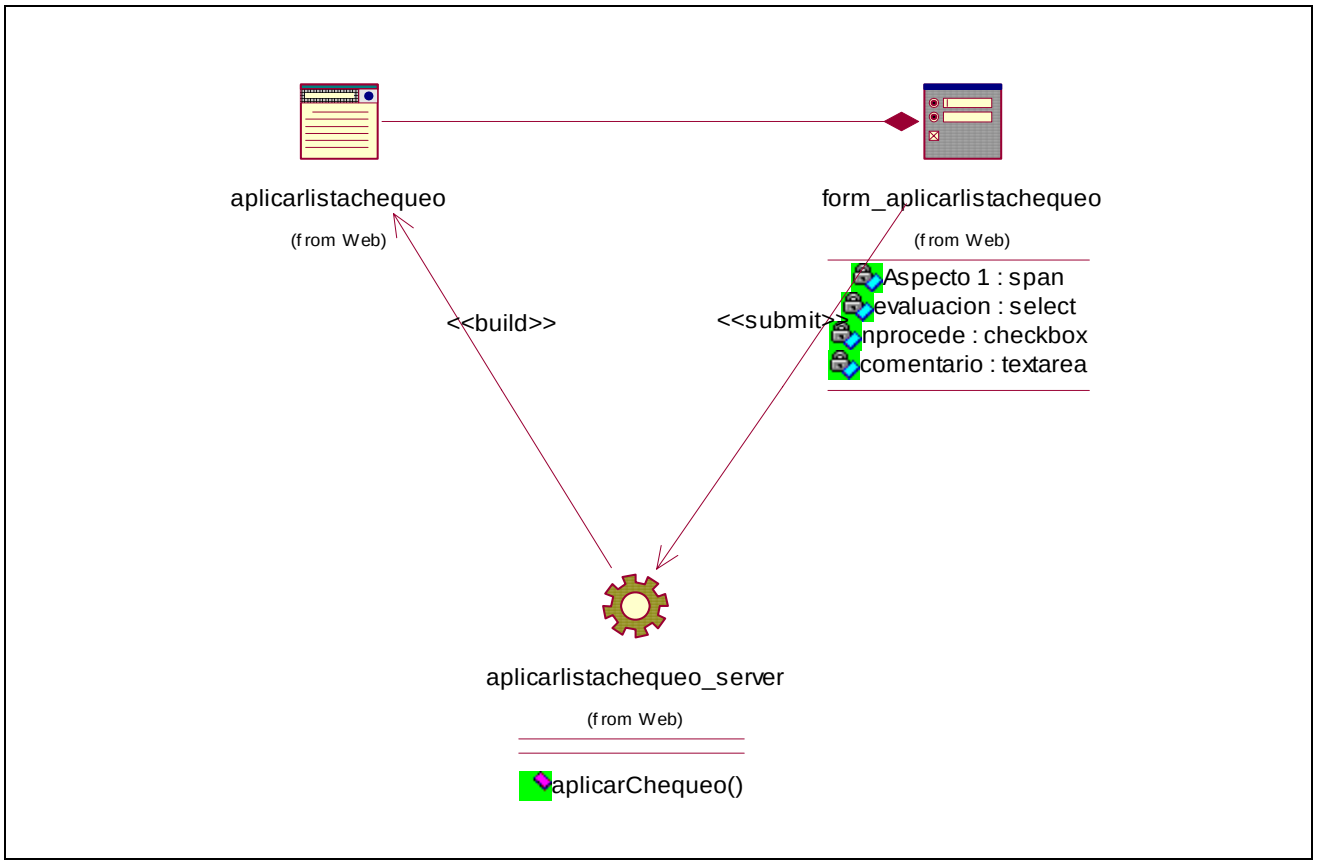


Figura 3.18 Diagrama de clases web para el caso de uso “Aplicar lista de chequeos”

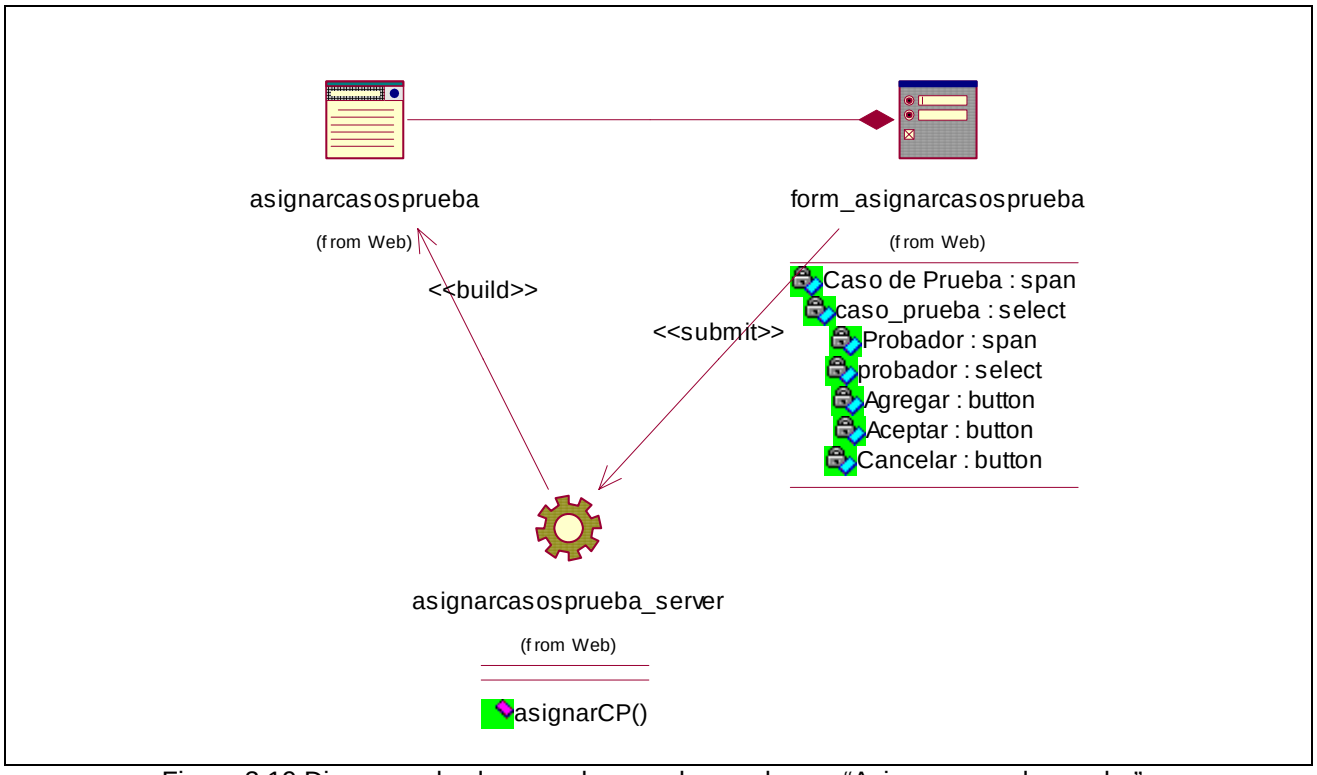


Figura 3.19 Diagrama de clases web para el caso de uso “Asignar caso de prueba”

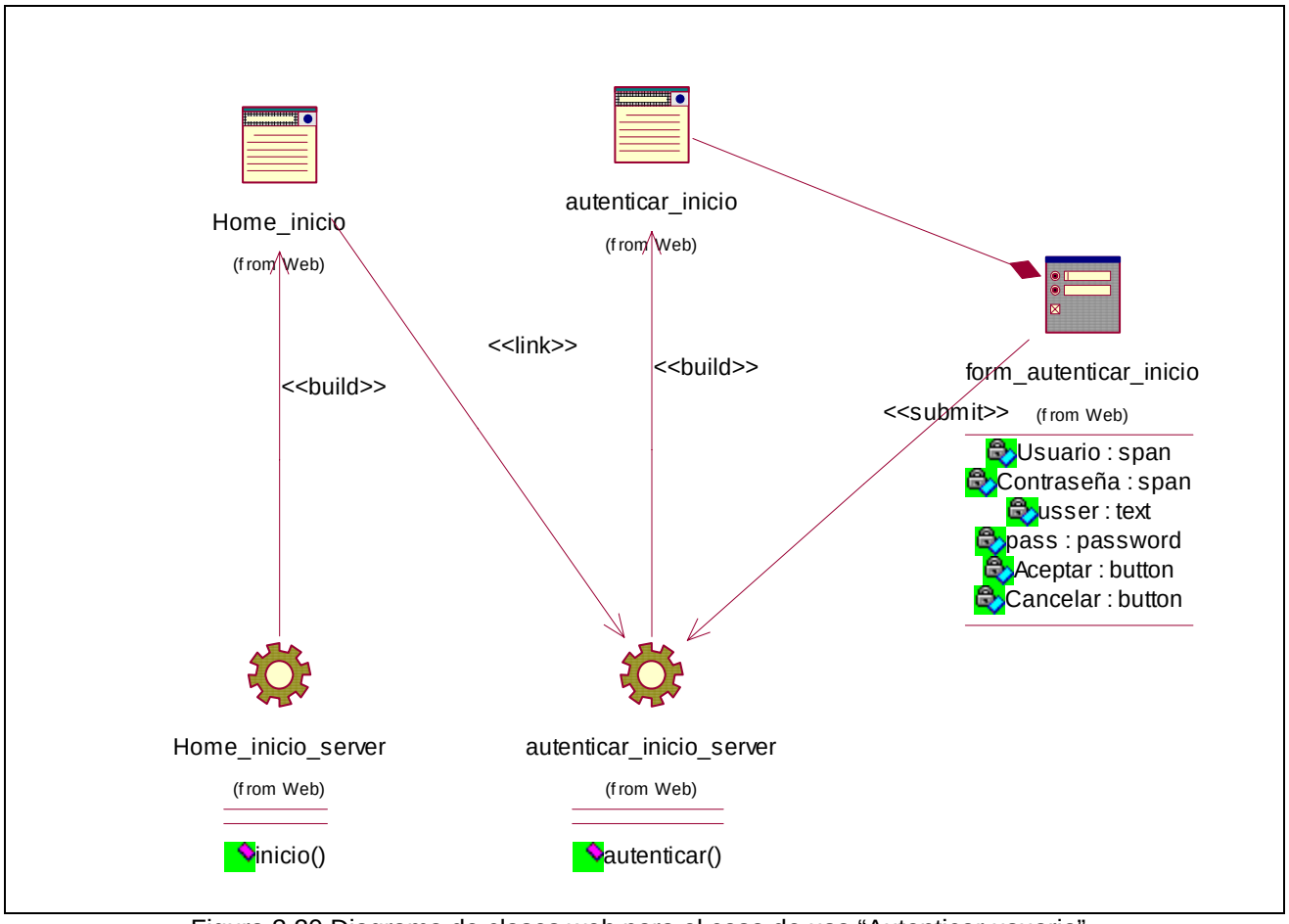


Figura 3.20 Diagrama de clases web para el caso de uso “Autenticar usuario”

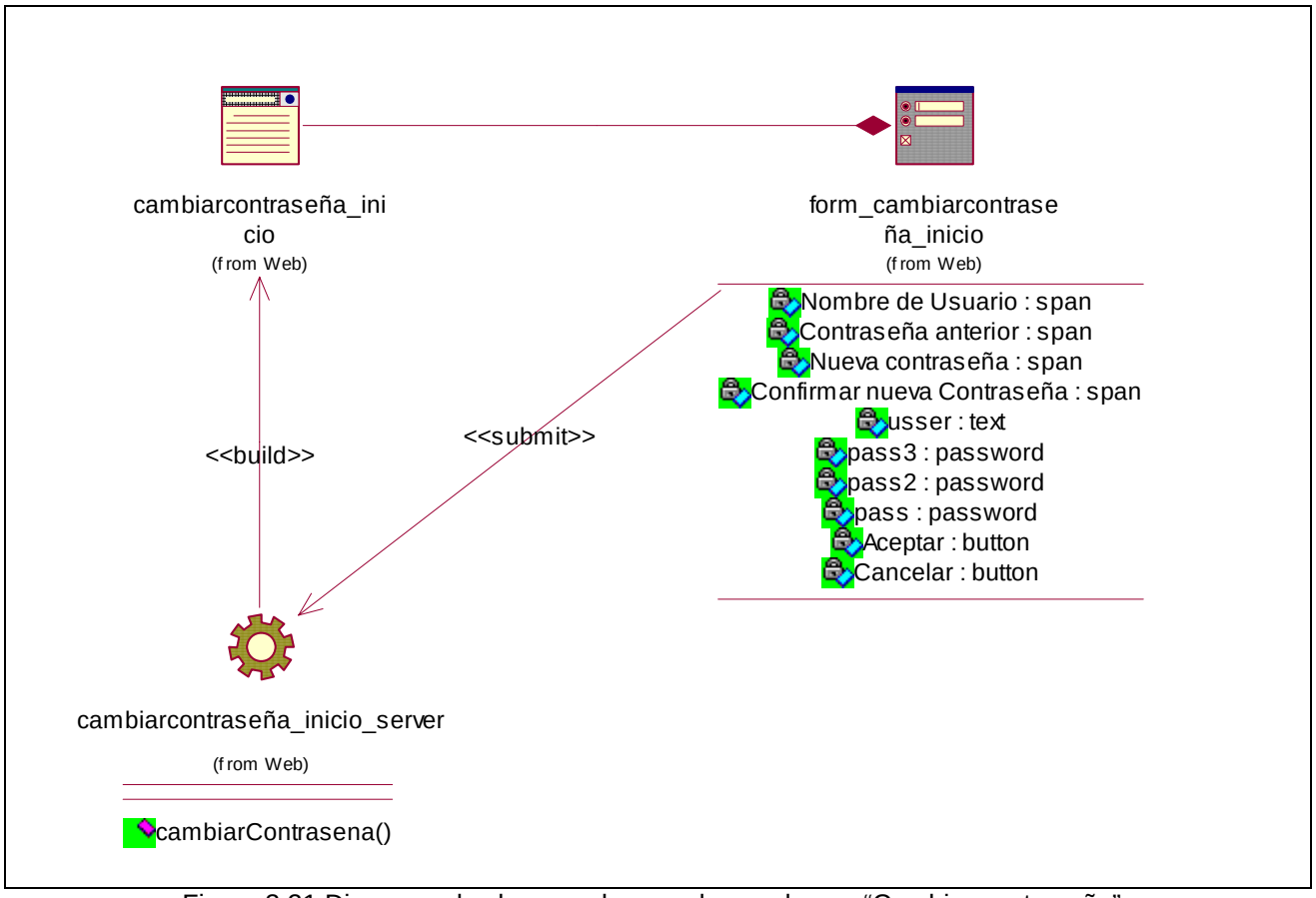


Figura 3.21 Diagrama de clases web para el caso de uso “Cambiar contraseña”

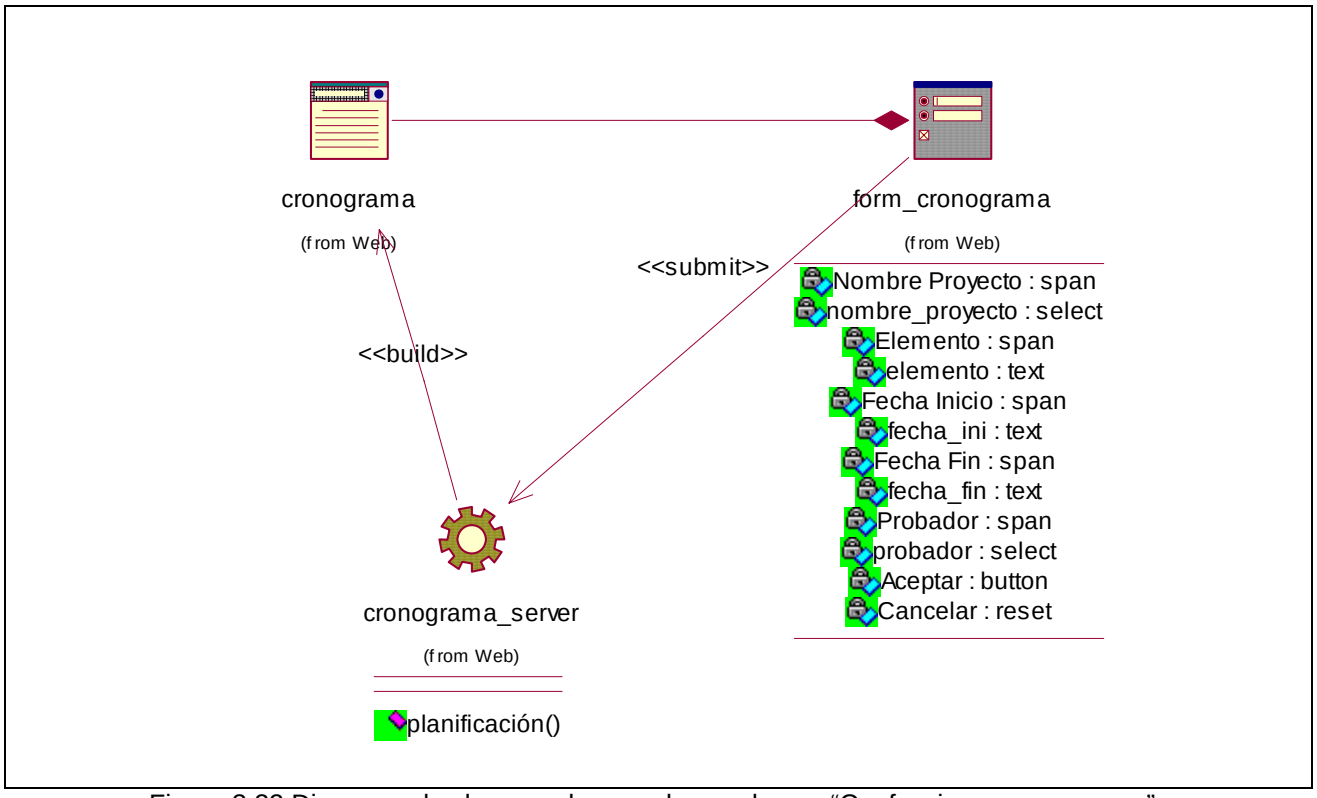


Figura 3.22 Diagrama de clases web para el caso de uso “Confeccionar cronograma”

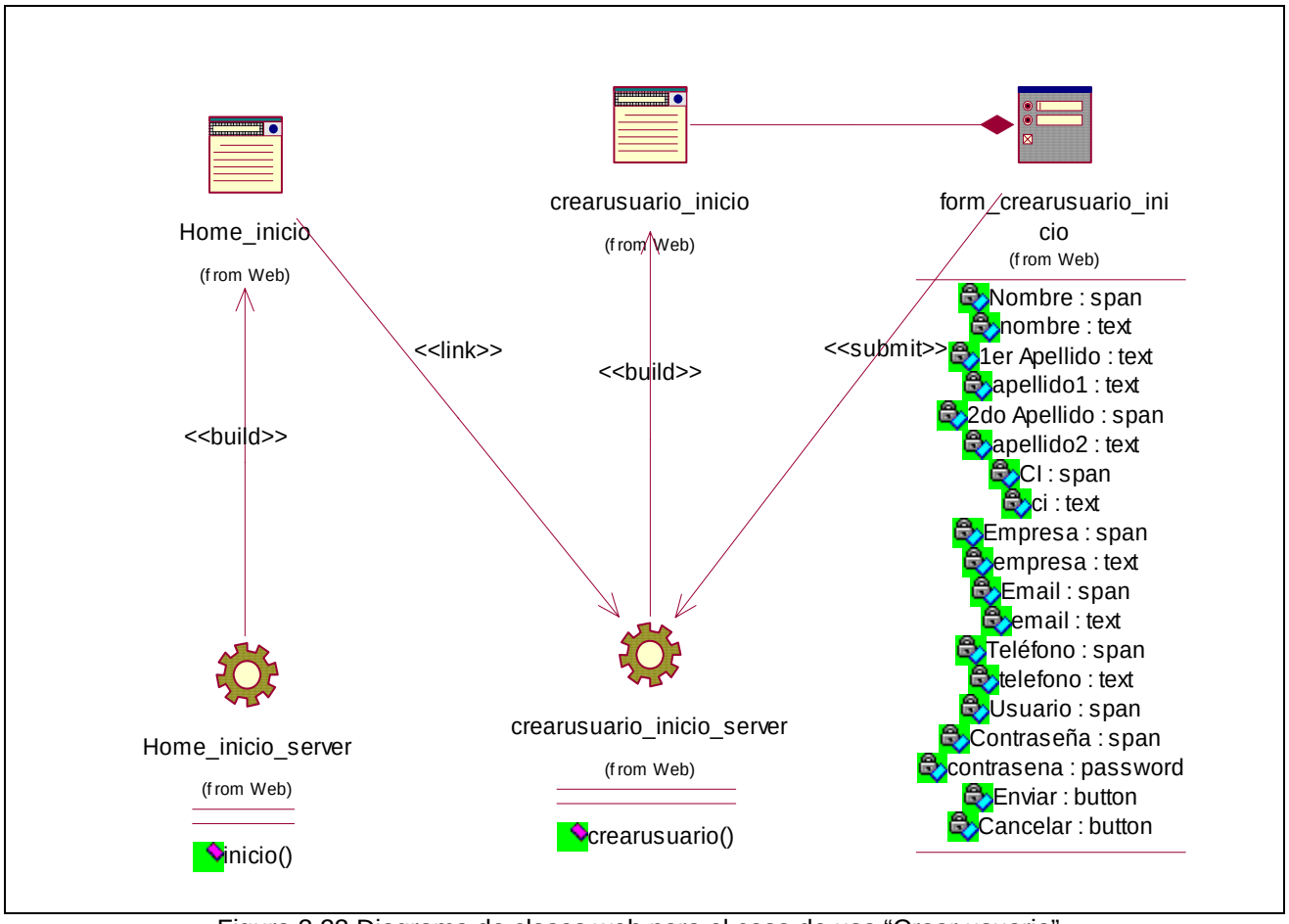


Figura 3.23 Diagrama de clases web para el caso de uso “Crear usuario”

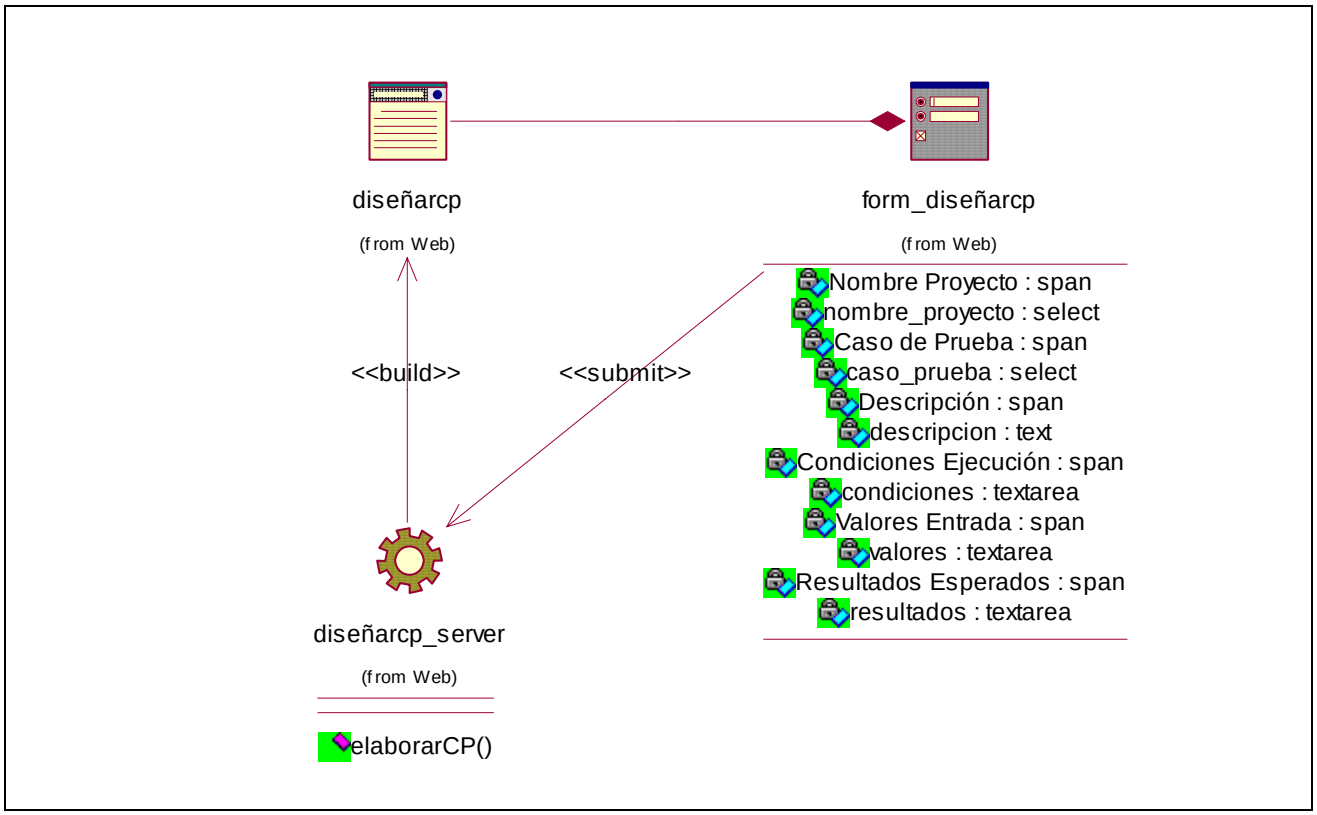


Figura 3.24 Diagrama de clases web para el caso de uso “Diseñar caso de prueba”

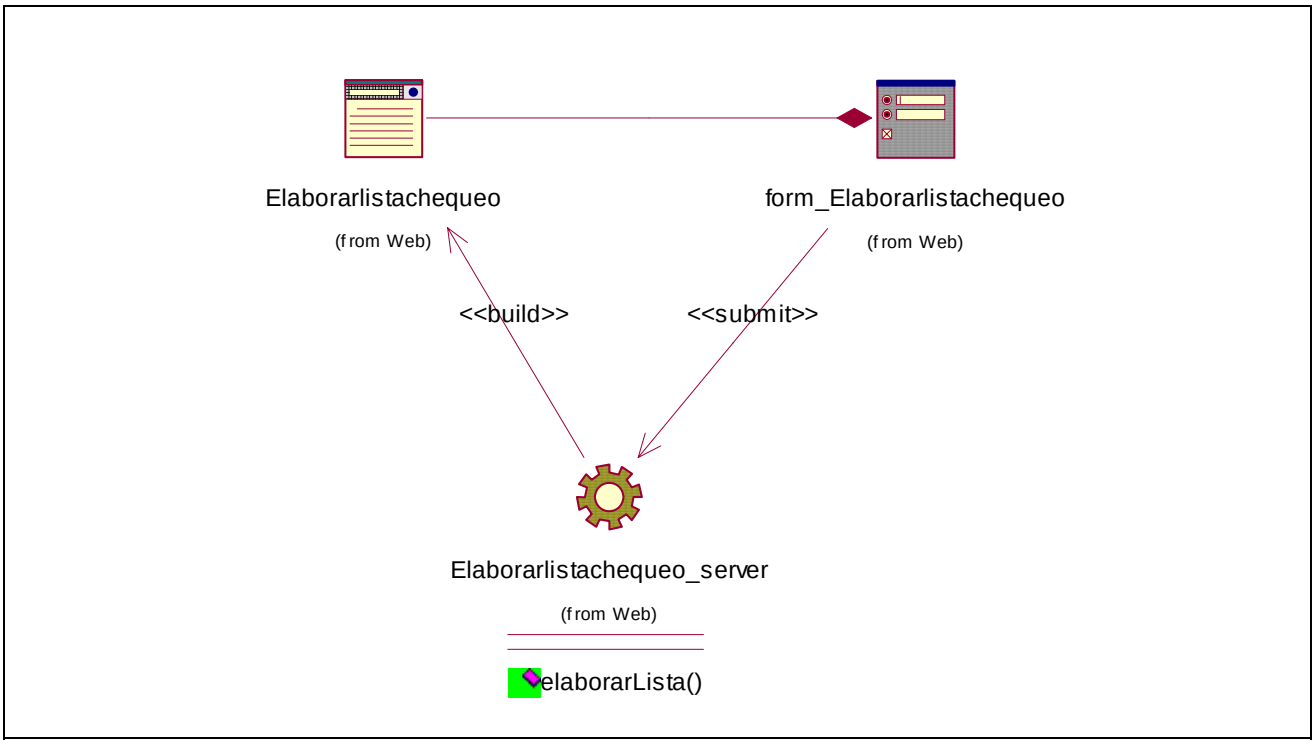


Figura 3.25 Diagrama de clases web para el caso de uso “Elaborar lista de chequeos”

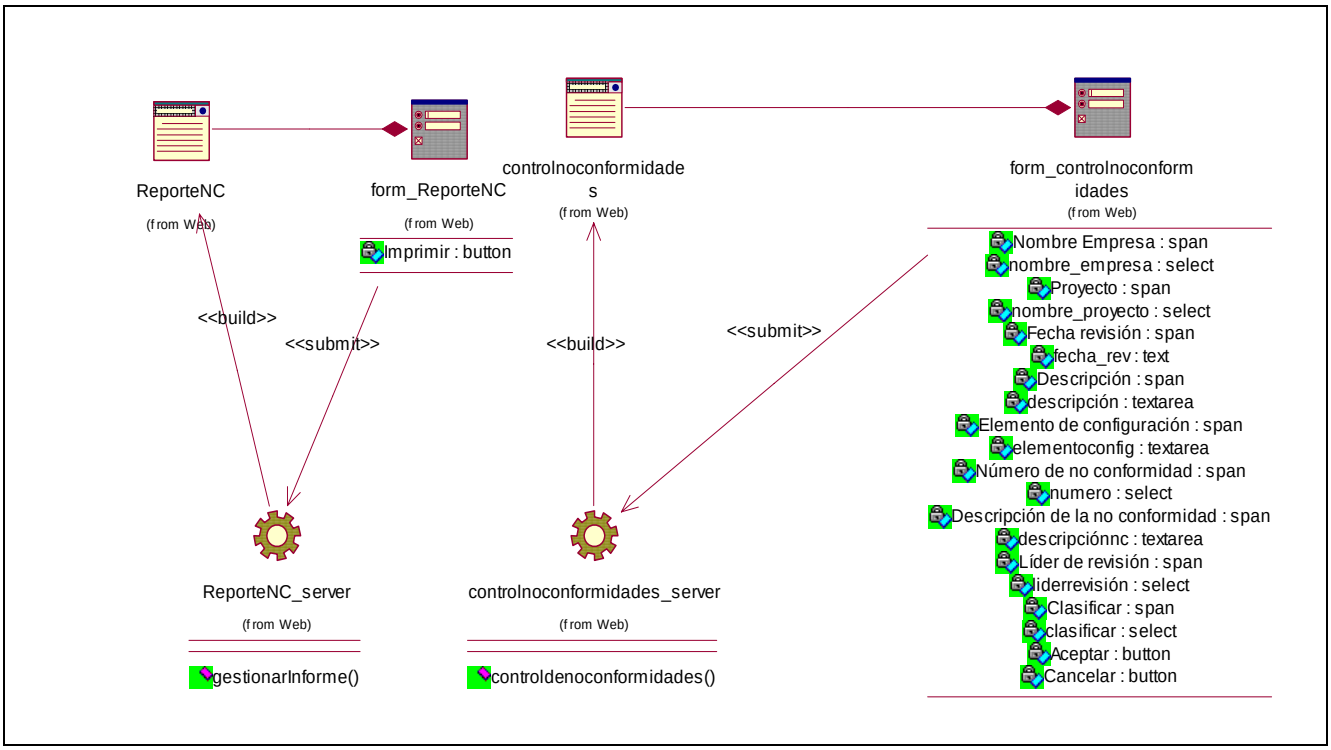


Figura 3.26 Diagrama de clases web para el caso de uso “Gestionar no conformidades”

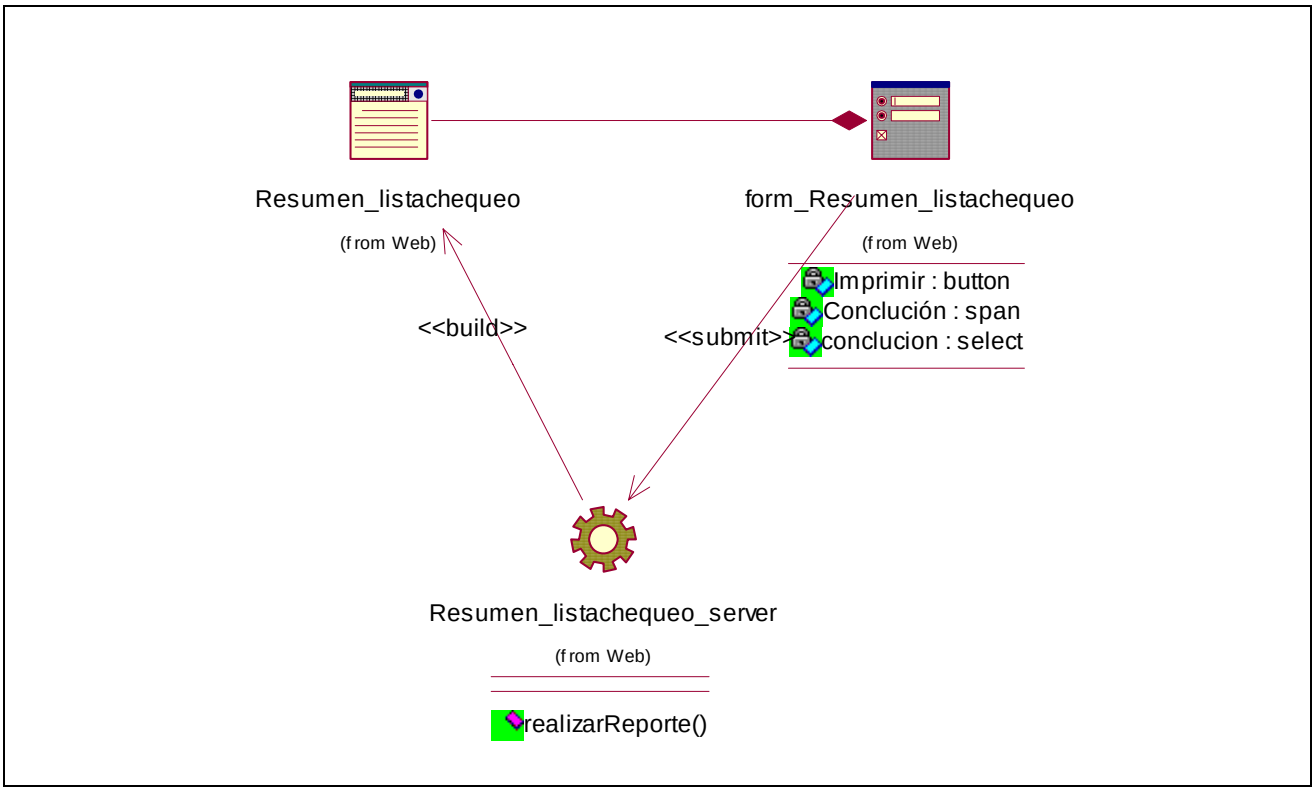


Figura 3.27 Diagrama de clases web para el caso de uso “Realizar reporte”

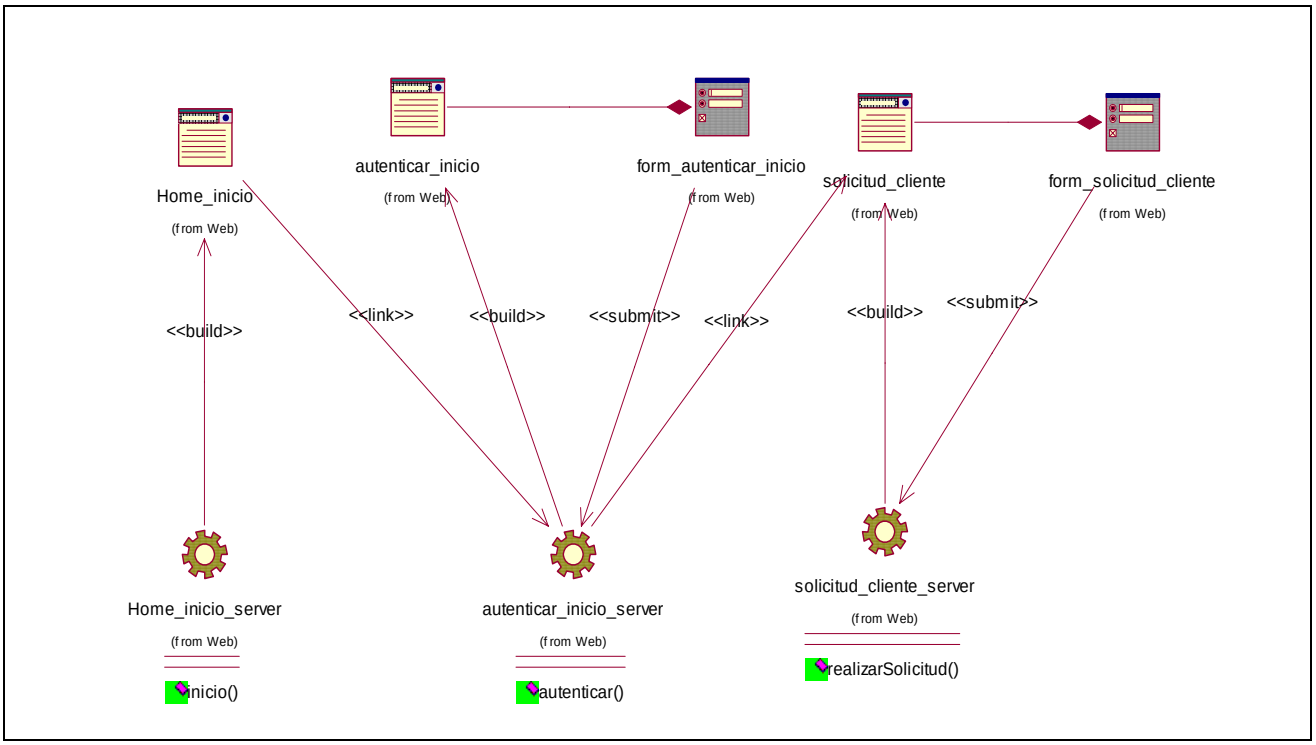


Figura 3.28 Diagrama de clases web para el caso de uso “Realizar solicitud”

3.3.3.1 Descripción de las clases (Ver anexos 3)

3.3.4 Diseño de la BD

La base de datos es el sistema utilizado para el almacenamiento de datos y acceso controlado a los datos almacenados. En este epígrafe se muestra el diseño de la base de datos del sistema propuesto a través del diagrama de clases persistentes y el esquema de la base de datos generados a partir de éste, el modelo de datos.

Diagrama de clases persistentes

Las clases persistentes son las clases que necesitan ser capaz de guardar su estado en un medio permanente, la necesidad de guardar su estado está dado por al almacenamiento físico permanente de la información de la clase, para la copia de seguridad en caso del fracaso del sistema, o para el intercambio de información. A continuación se muestra el diagrama de clases persistentes.

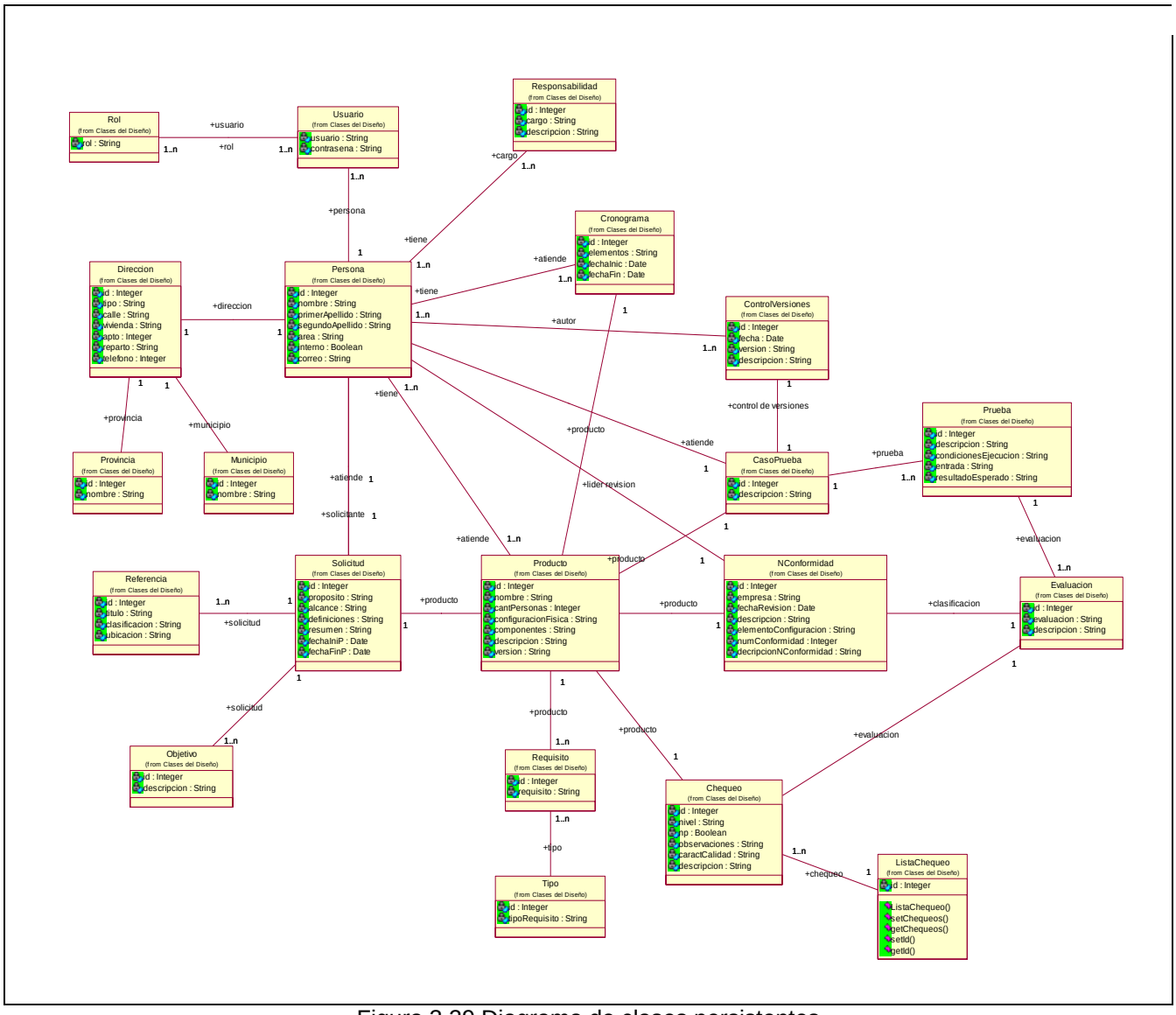


Figura 3.29 Diagrama de clases persistentes

Modelo de Datos

El modelo de los datos describe la representación lógica y física de datos persistentes en el sistema. A continuación se muestra el modelo de datos

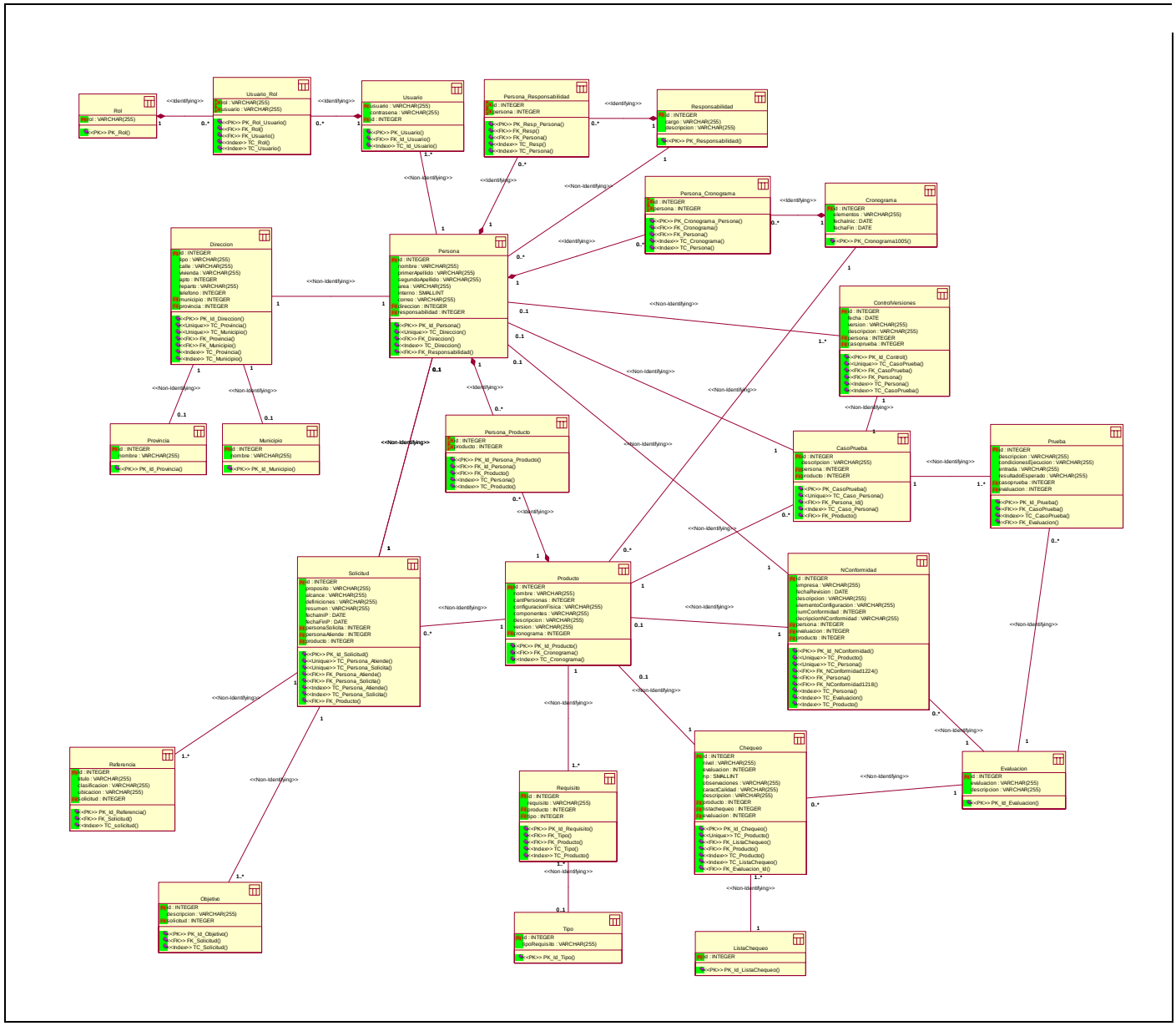


Figura 3.30 Modelo de Datos

3.3.4.1 Descripción de las tablas de la BD (Ver Anexos 4)

3.3.5 Diagrama de despliegue

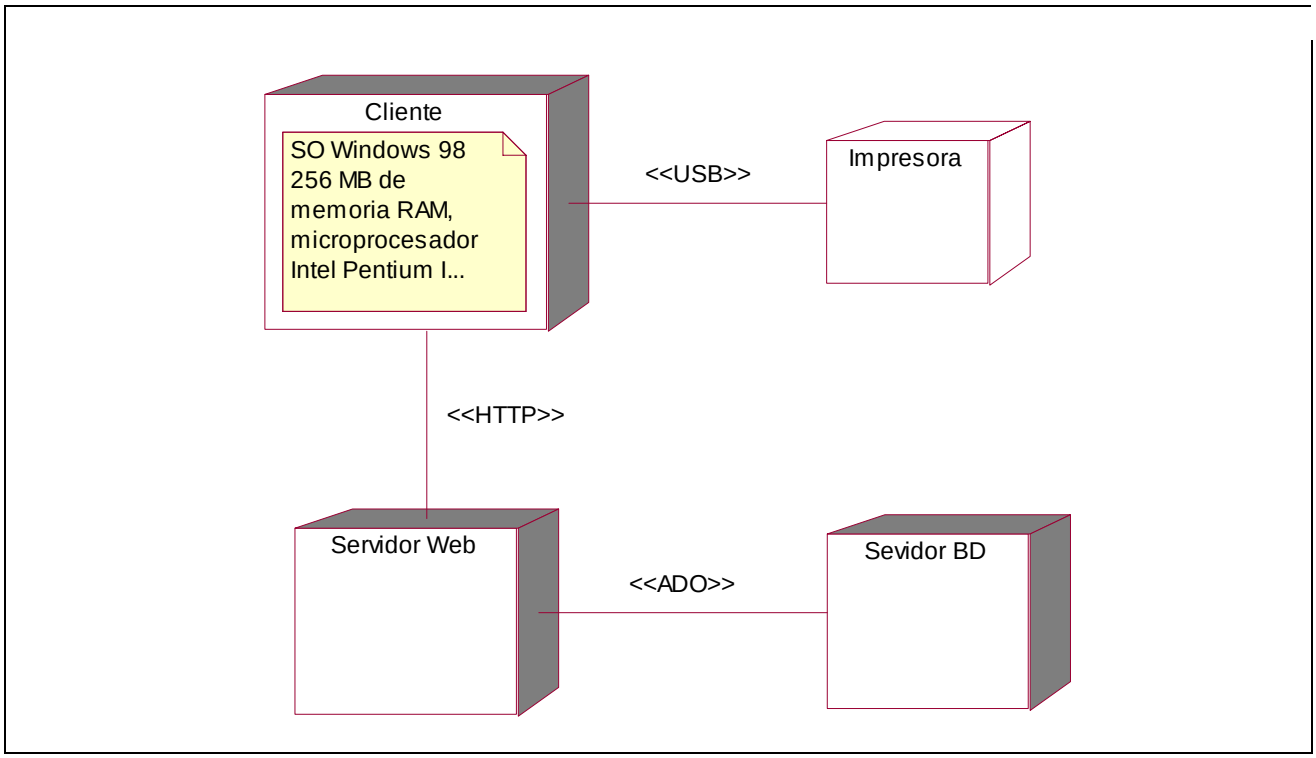


Figura 3.31 Diagrama de Despliegue

3.4 Diagramas de Implementación

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. De forma más específica los propósitos de la implementación son:

- Planificar las integraciones de sistema necesarias en cada iteración.
- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue.
- Implementar las clases y subsistemas encontrados durante el diseño.
- Probar los componentes individualmente, y a continuación integrarlos compilándolos y enlazándolos en uno o más ejecutables.

3.4.1 Herramientas, sistema gestor de base de datos y lenguaje propuestos para la implementación

Las herramientas que se proponen para la implementación del sistema, son: Macromedia Dreamweaver 8 y Adobe Photoshop CS2 por las mismas razones planteadas de por que se utilizaron para la confección del prototipo, como sistema gestor de base de datos: MySQL por poseer una arquitectura que lo hace extremadamente rápido y fácil de personalizar, además brinda un sistema de administración de la base de datos veloz, compactado y estable. Y como lenguaje de programación Java por las características que presenta de: orientado a objetos, distribuido y dinámico, robusto, seguro, multitarea y portable; este lenguaje fue diseñado para la creación de páginas web, y lo que se plantea para el sistema es una aplicación web ya que: no requieren de instalación, pues usan tecnología web, lo cual permite el aprovechamiento de todas las características de Internet; son fáciles de usar porque no necesitan conocimientos avanzados de computación y poseen una alta disponibilidad debido a que pueden ser consultadas desde cualquier parte del mundo gracias a Internet y a cualquier hora.

3.4.2 Diagrama de Componentes

Los diagramas de componentes muestran las dependencias del compilador y del "*runtime*" entre los componentes del software.

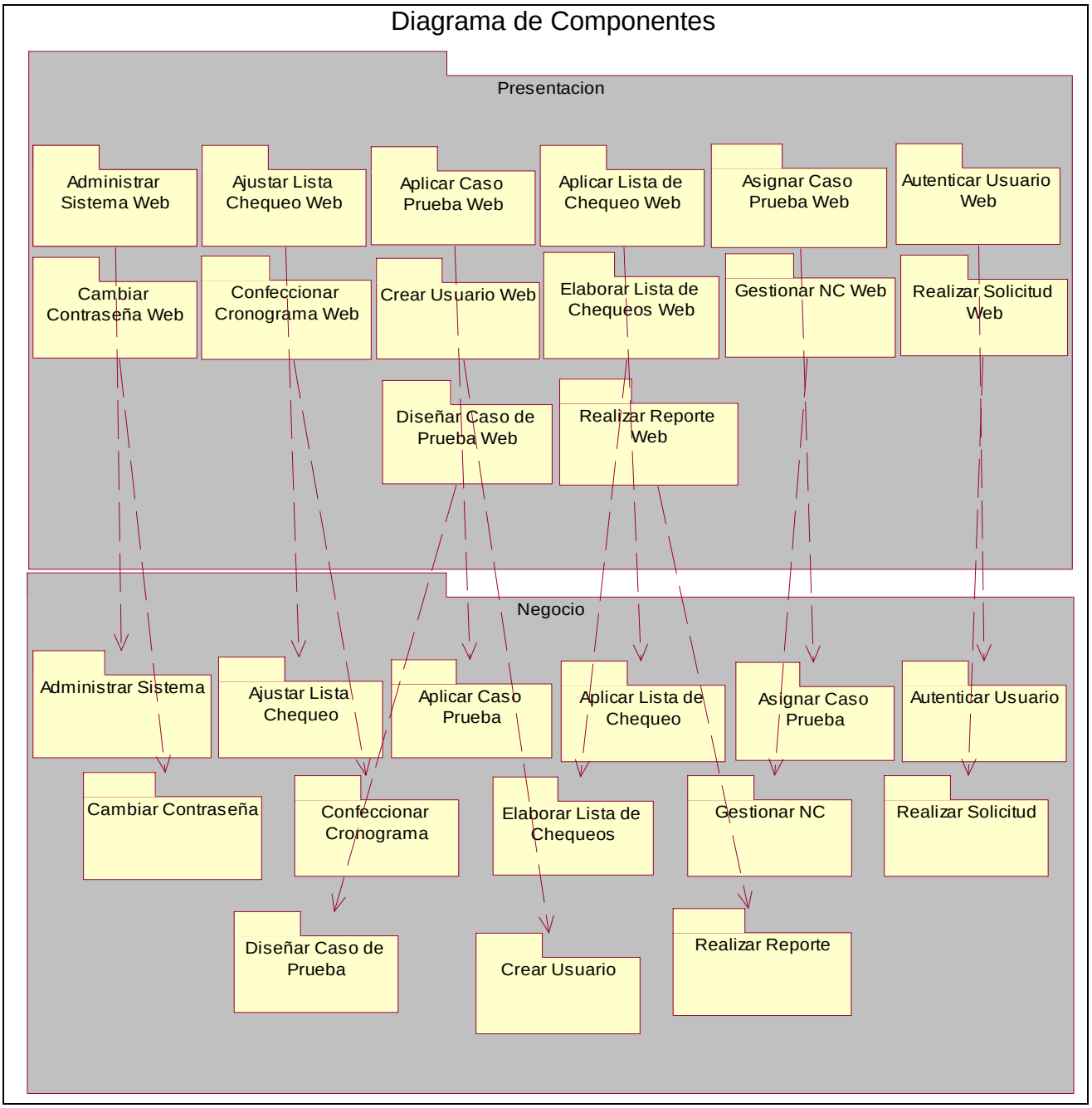


Figura 3.32 Diagrama de Componentes (Vista de la relación entre presentación y negocio)

Diagramas de componentes por caso de uso (presentación)

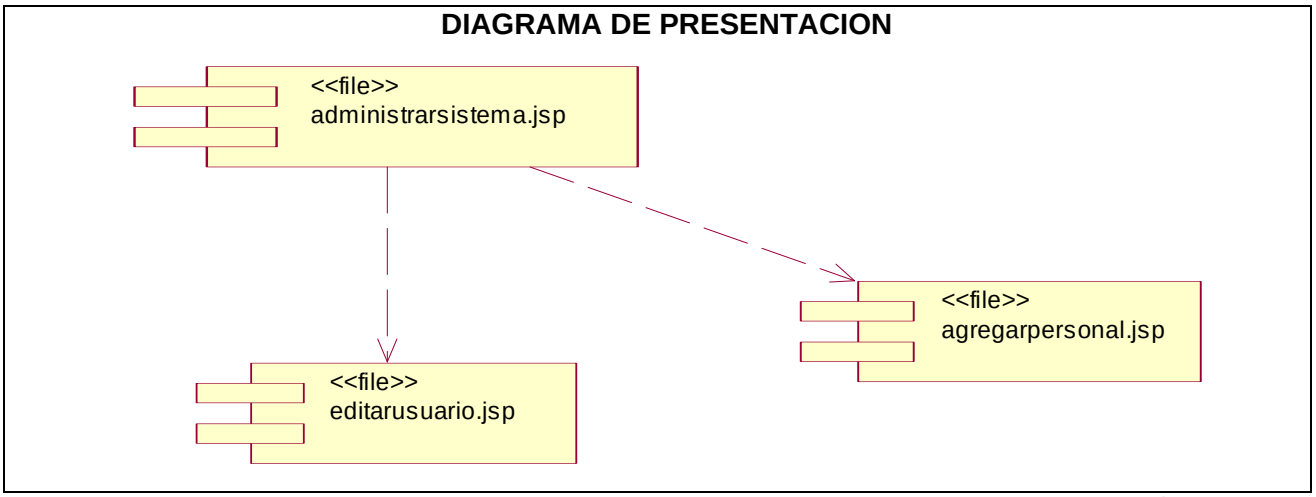


Figura 3.33 Diagramas de componentes Caso de uso “Administrar sistema” (presentación)

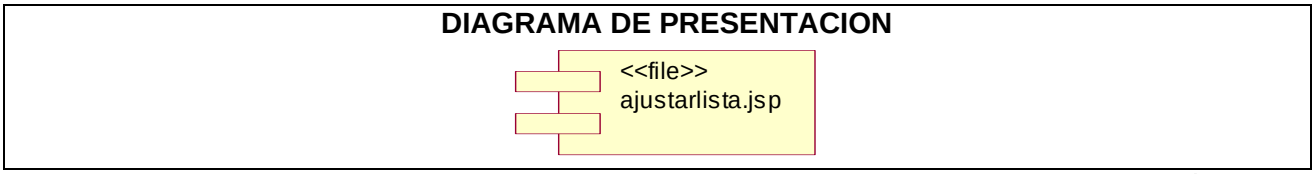


Figura 3.34 Diagramas de componentes Caso de uso “Ajustar lista de chequeos” (presentación)

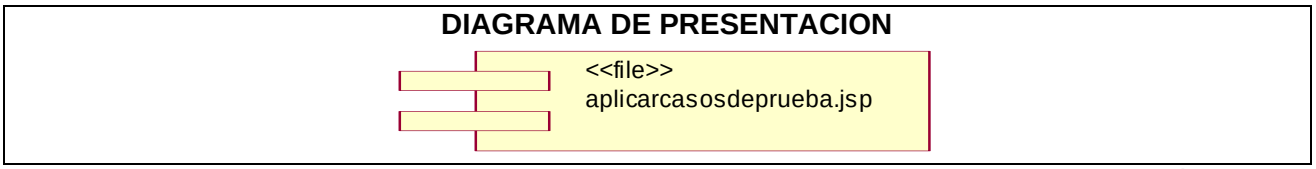


Figura 3.35 Diagramas de componentes Caso de uso “Aplicar caso de prueba” (presentación)

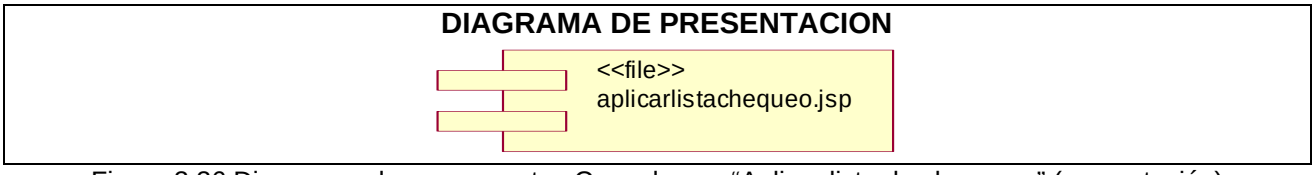


Figura 3.36 Diagramas de componentes Caso de uso “Aplicar lista de chequeos” (presentación)

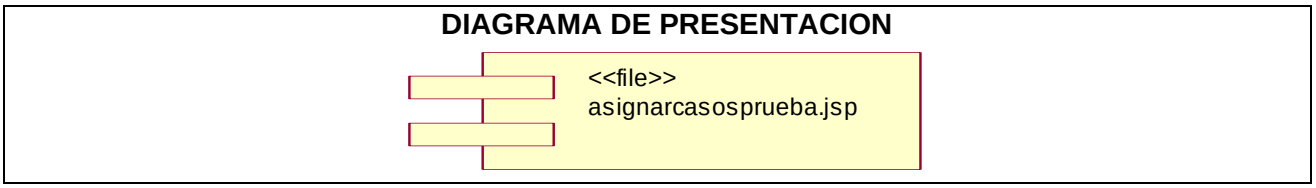


Figura 3.37 Diagramas de componentes Caso de uso “Asignar casos de prueba” (presentación)

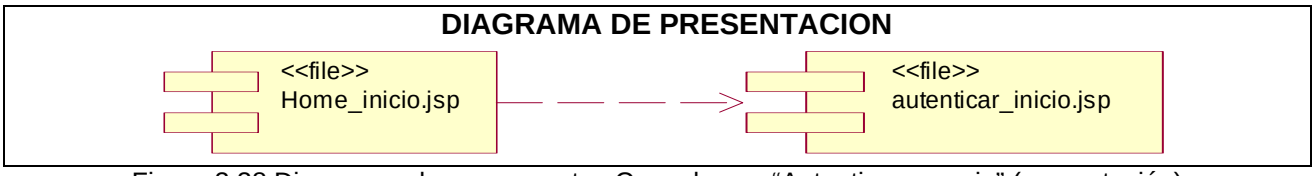


Figura 3.38 Diagramas de componentes Caso de uso “Autenticar usuario” (presentación)

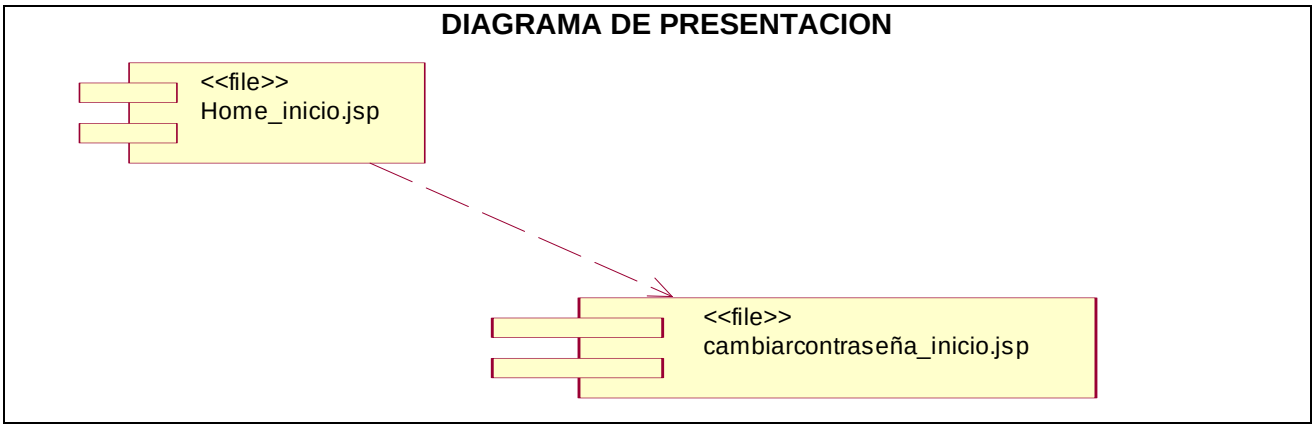


Figura 3.39 Diagramas de componentes Caso de uso “Cambiar contraseña” (presentación)

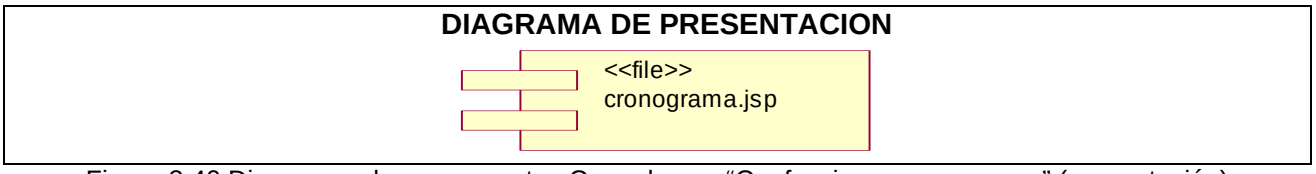


Figura 3.40 Diagramas de componentes Caso de uso “Confeccionar cronograma” (presentación)

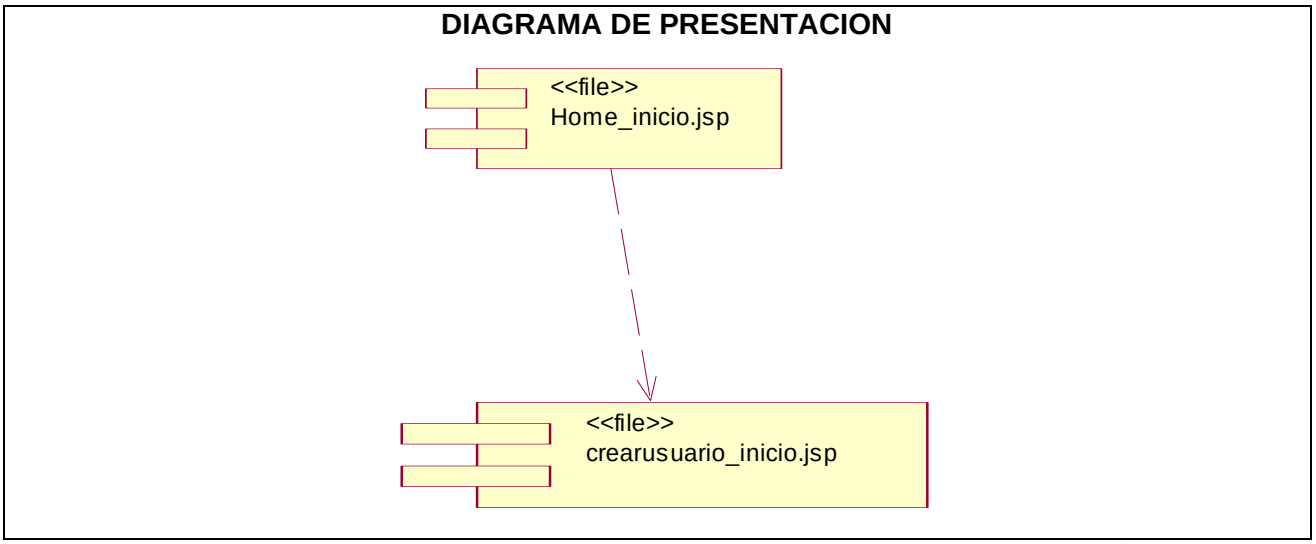


Figura 3.41 Diagramas de componentes Caso de uso “Crear usuario” (presentación)

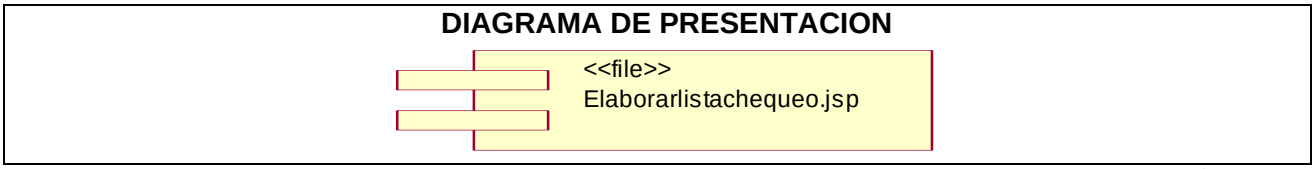


Figura 3.42 Diagramas de componentes Caso de uso “Elaborar lista de chequeos” (presentación)

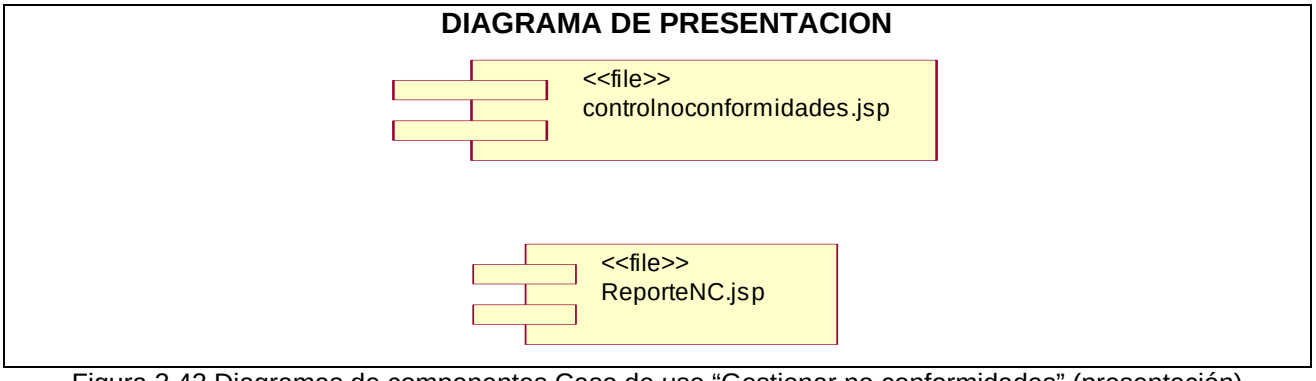


Figura 3.43 Diagramas de componentes Caso de uso “Gestionar no conformidades” (presentación)

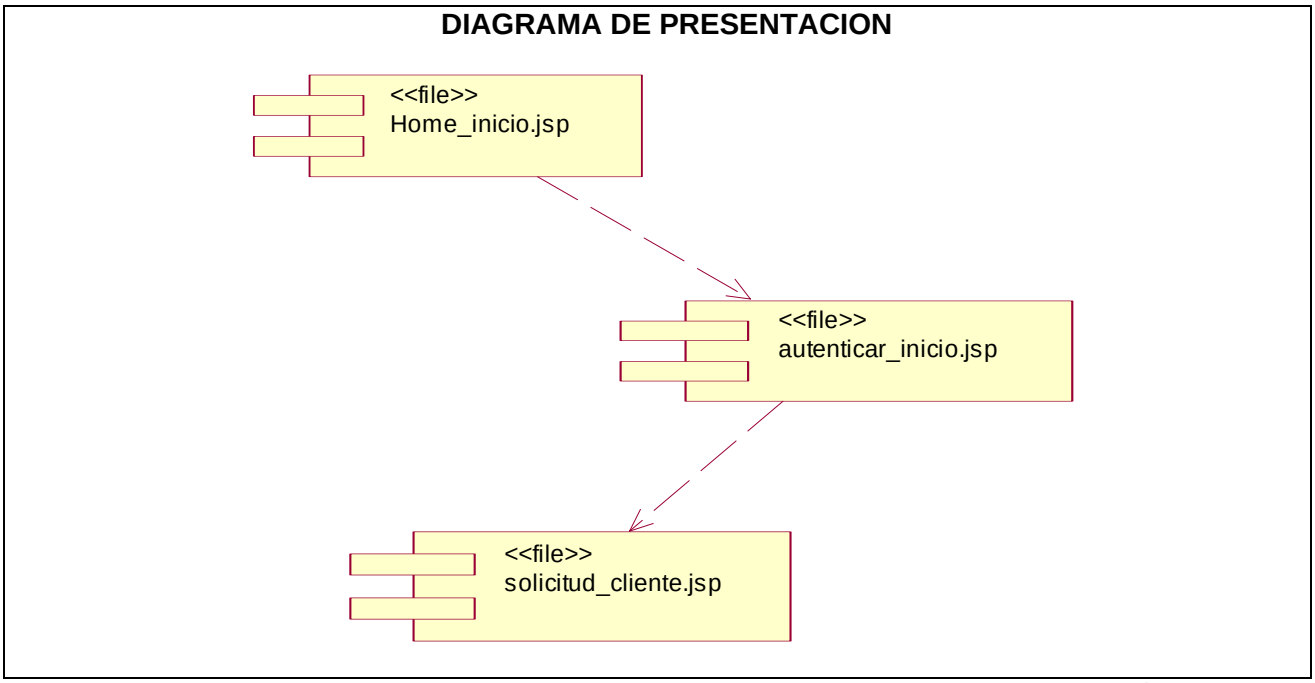


Figura 3.44 Diagramas de componentes Caso de uso “Realizar solicitud” (presentación)

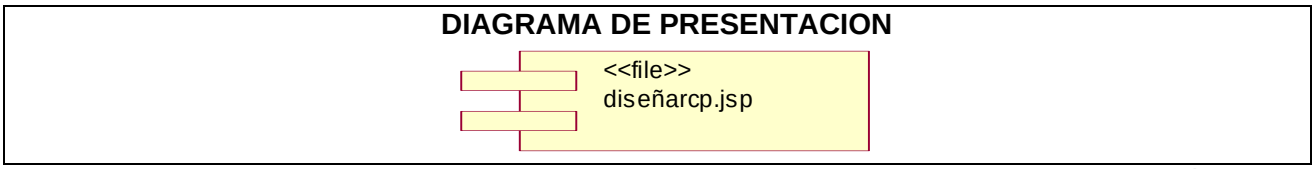


Figura 3.45 Diagramas de componentes Caso de uso “Diseñar casos de prueba” (presentación)

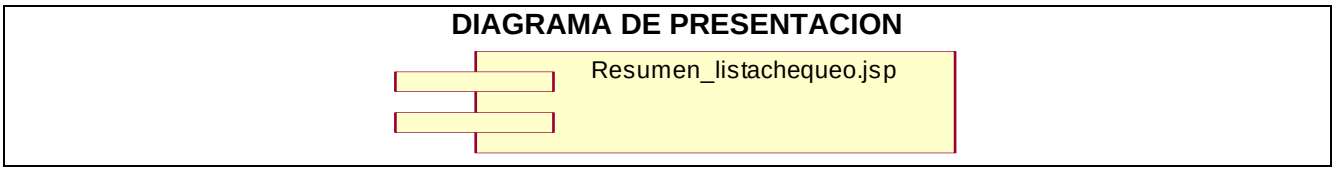


Figura 3.46 Diagramas de componentes Caso de uso “Realizar reporte” (presentación)

Diagramas de componentes por caso de uso (Negocio)

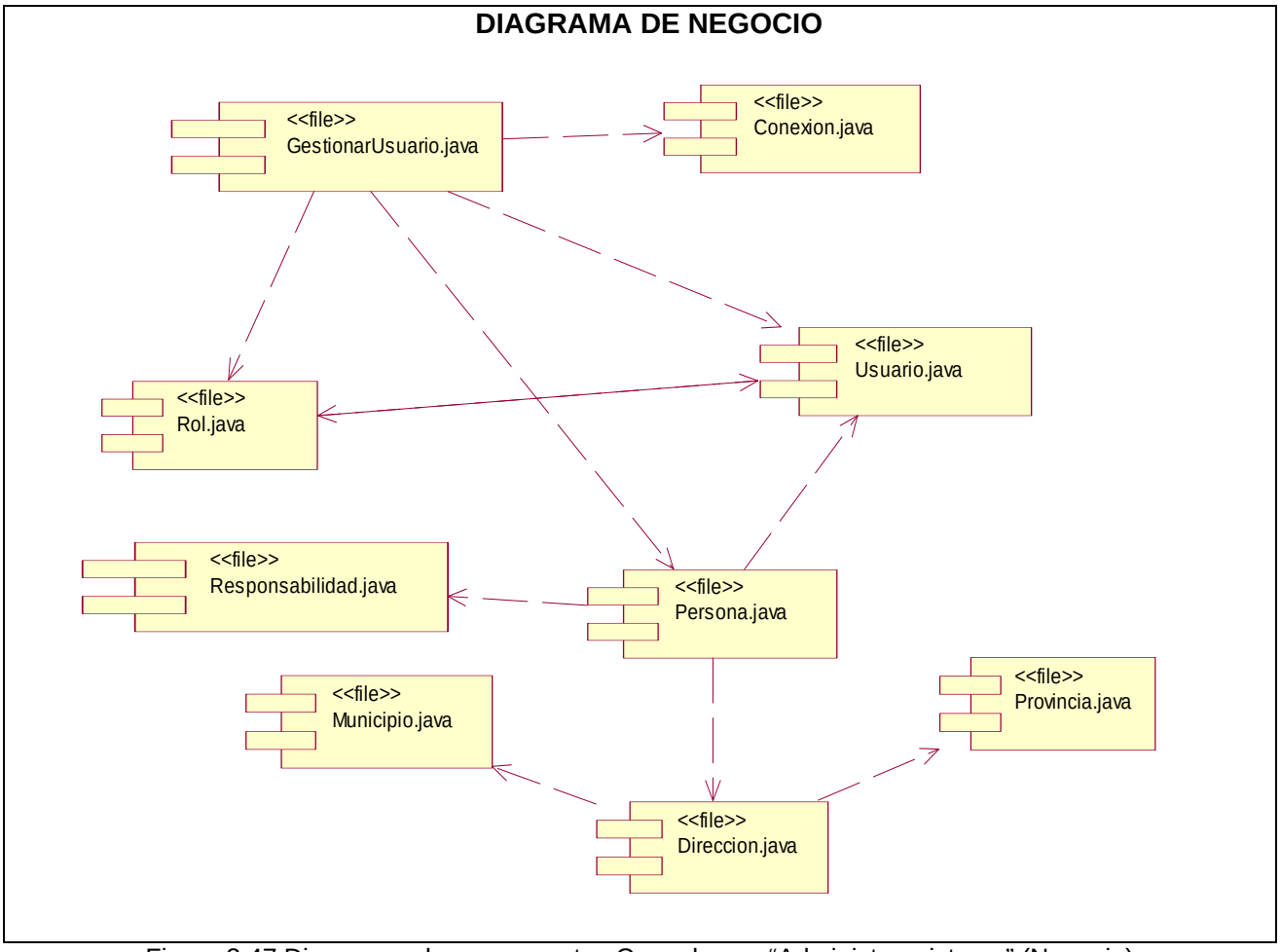


Figura 3.47 Diagramas de componentes Caso de uso “Administrar sistema” (Negocio)

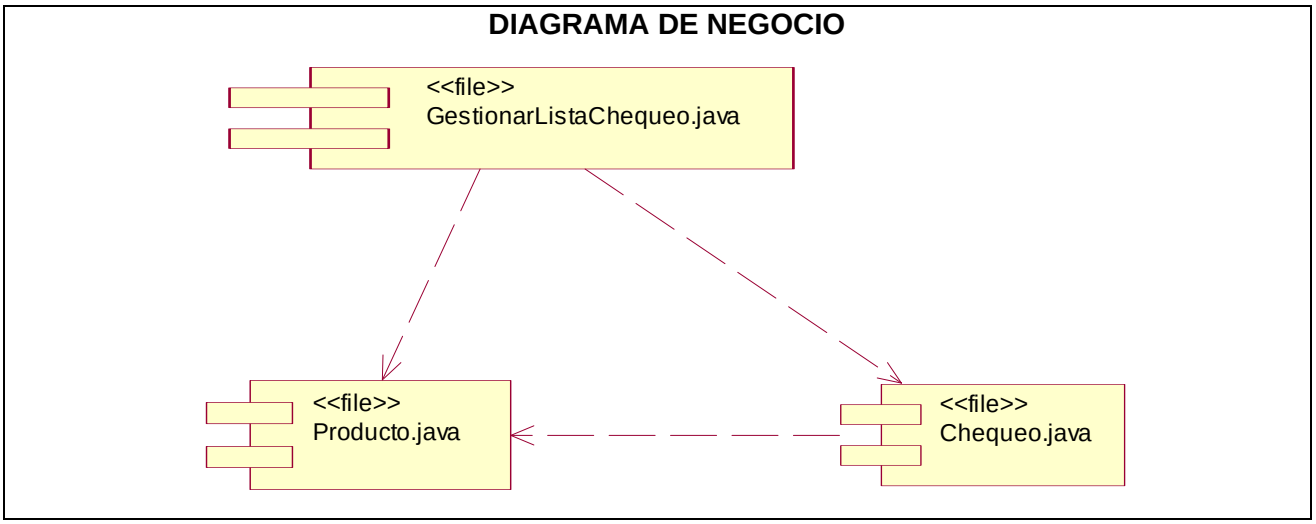


Figura 3.48 Diagramas de componentes Caso de uso “Ajustar lista de chequeos” (Negocio)

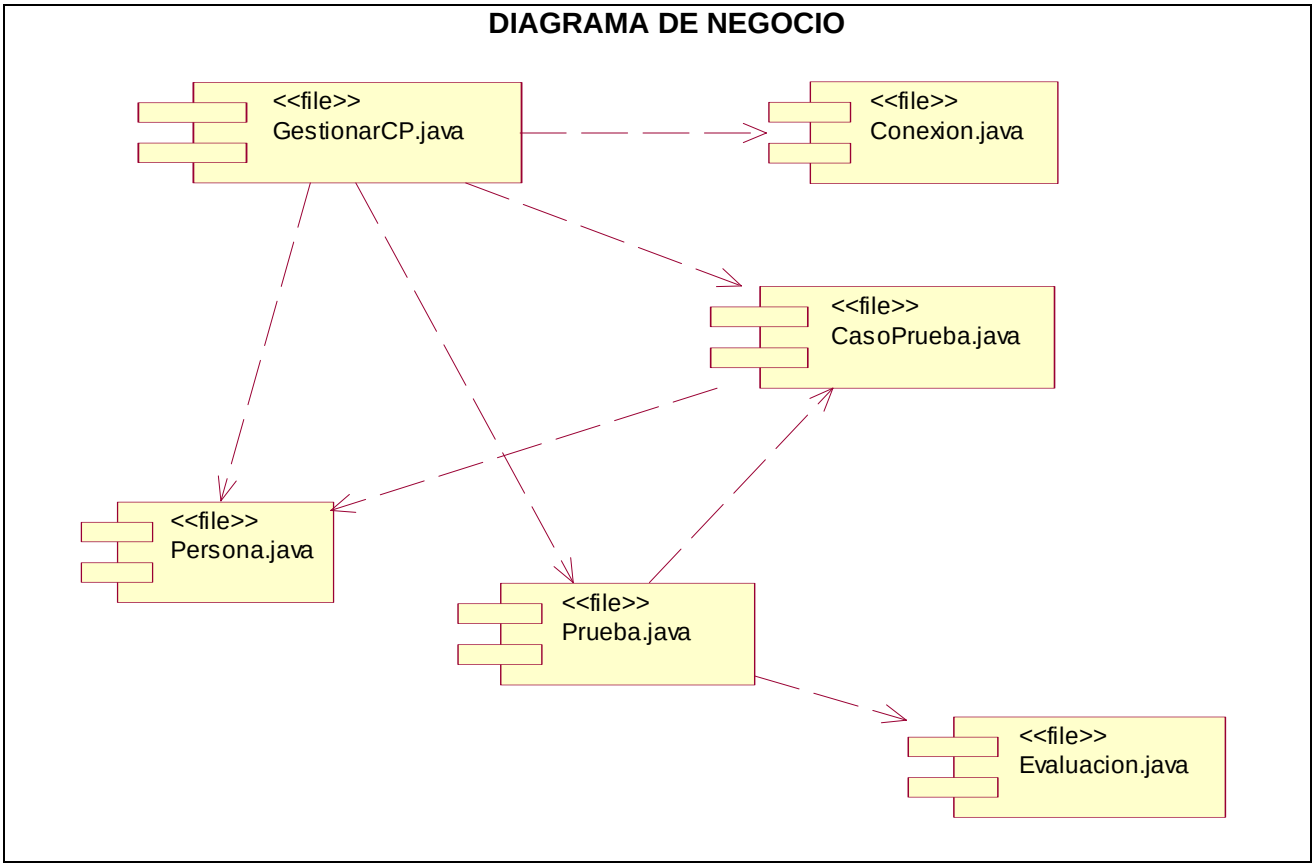


Figura 3.49 Diagramas de componentes Caso de uso “Aplicar casos de prueba” (Negocio)

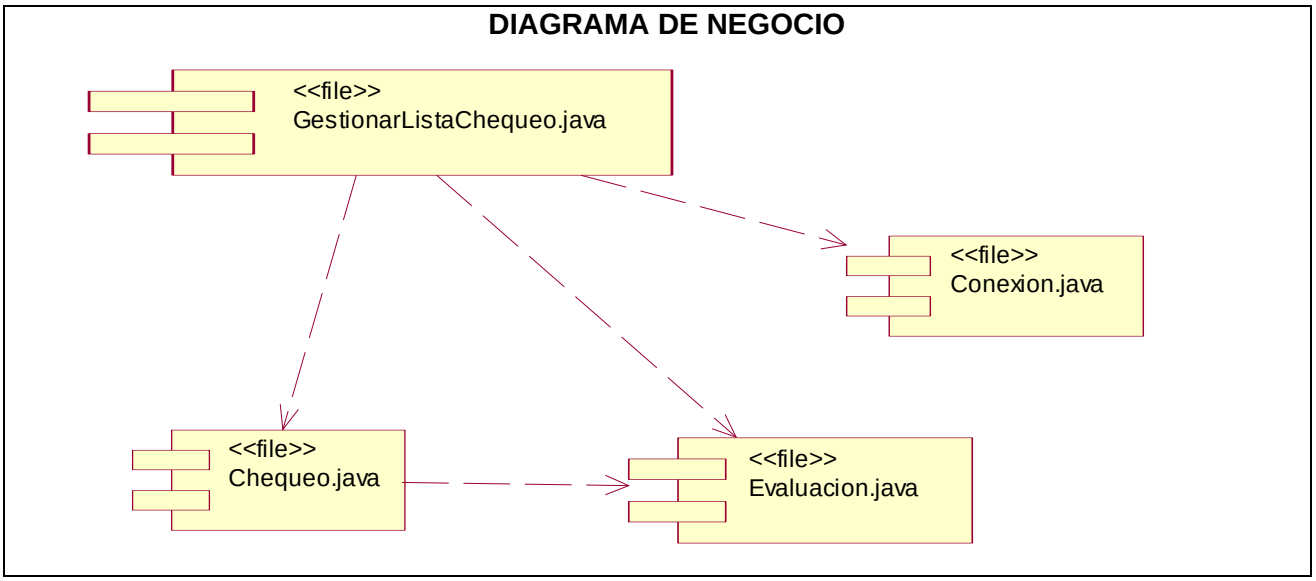


Figura 3.50 Diagramas de componentes Caso de uso “Aplicar lista de chequeos” (Negocio)

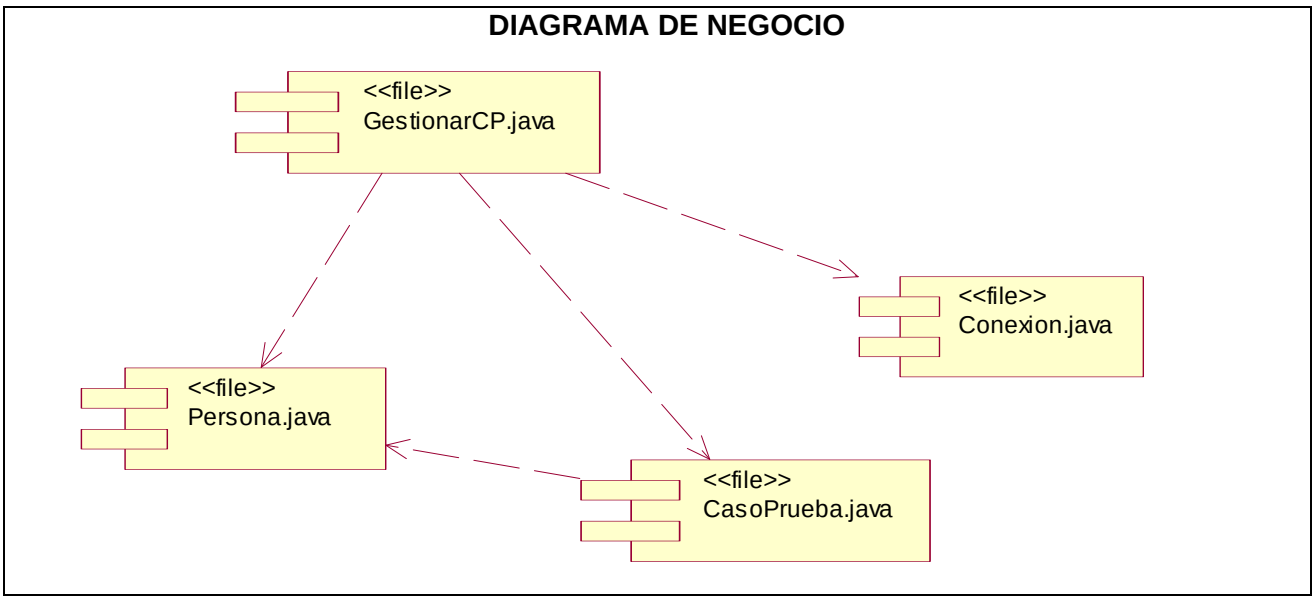


Figura 3.51 Diagramas de componentes Caso de uso “Asignar casos de prueba” (Negocio)

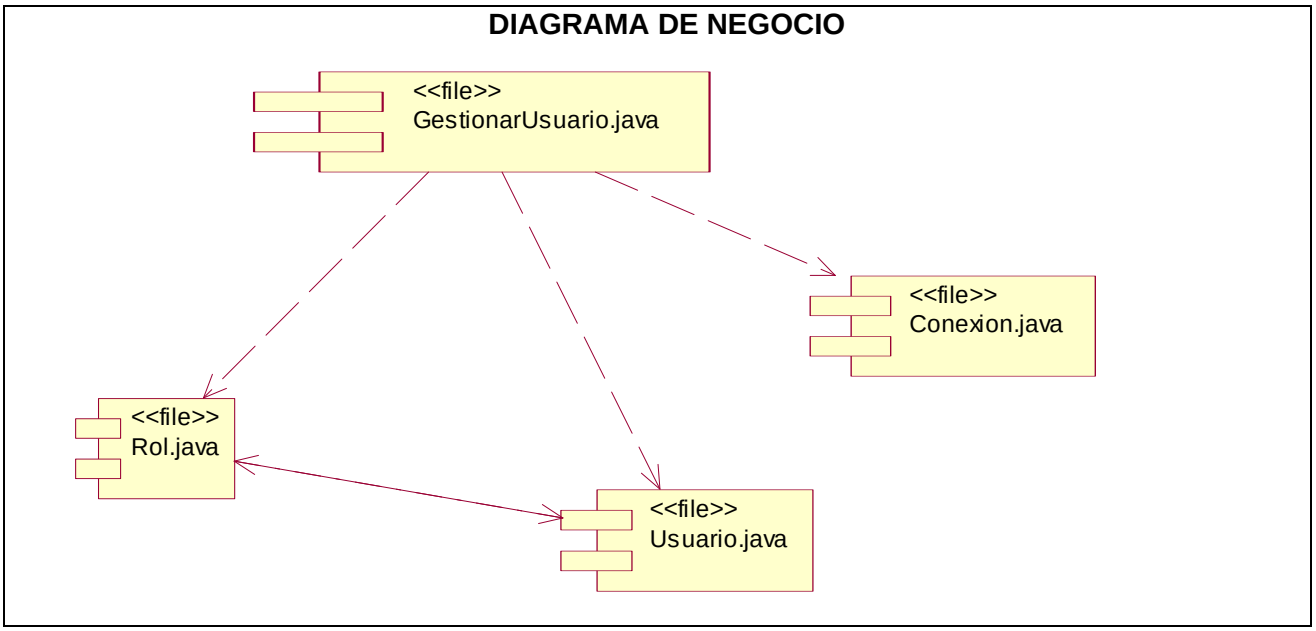


Figura 3.52 Diagramas de componentes Caso de uso “Autenticar usuario” (Negocio)

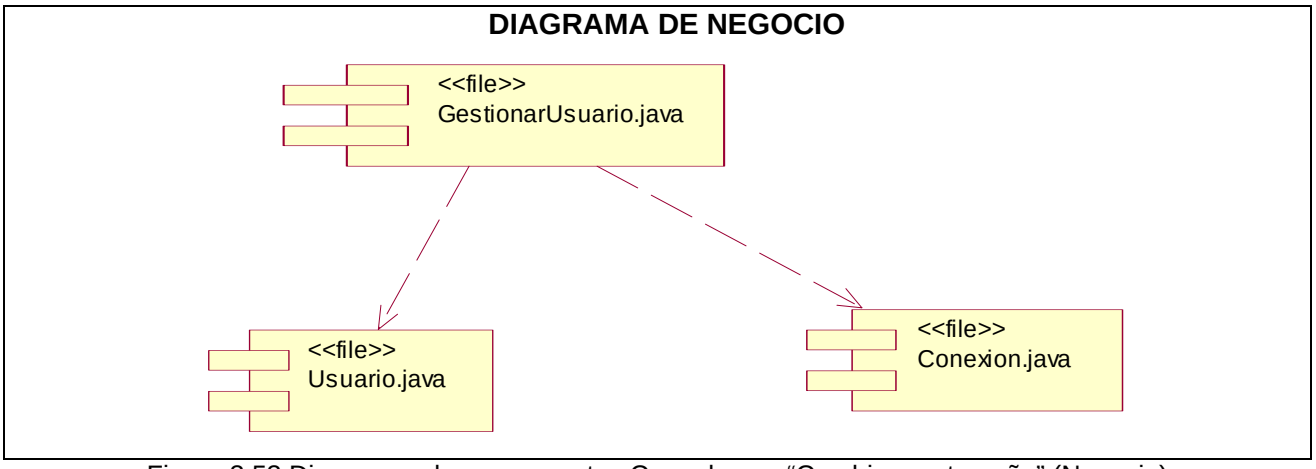


Figura 3.53 Diagramas de componentes Caso de uso “Cambiar contraseña” (Negocio)

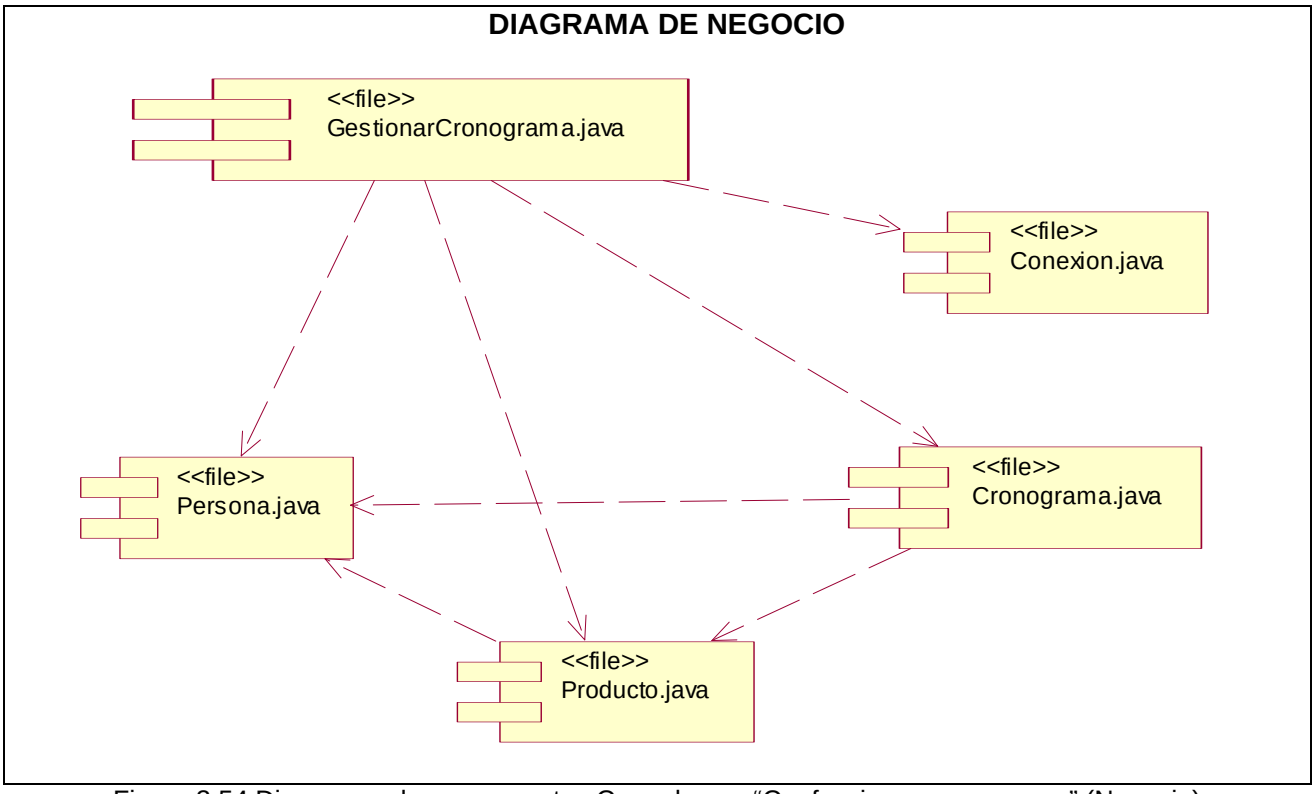


Figura 3.54 Diagramas de componentes Caso de uso “Confeccionar cronograma” (Negocio)

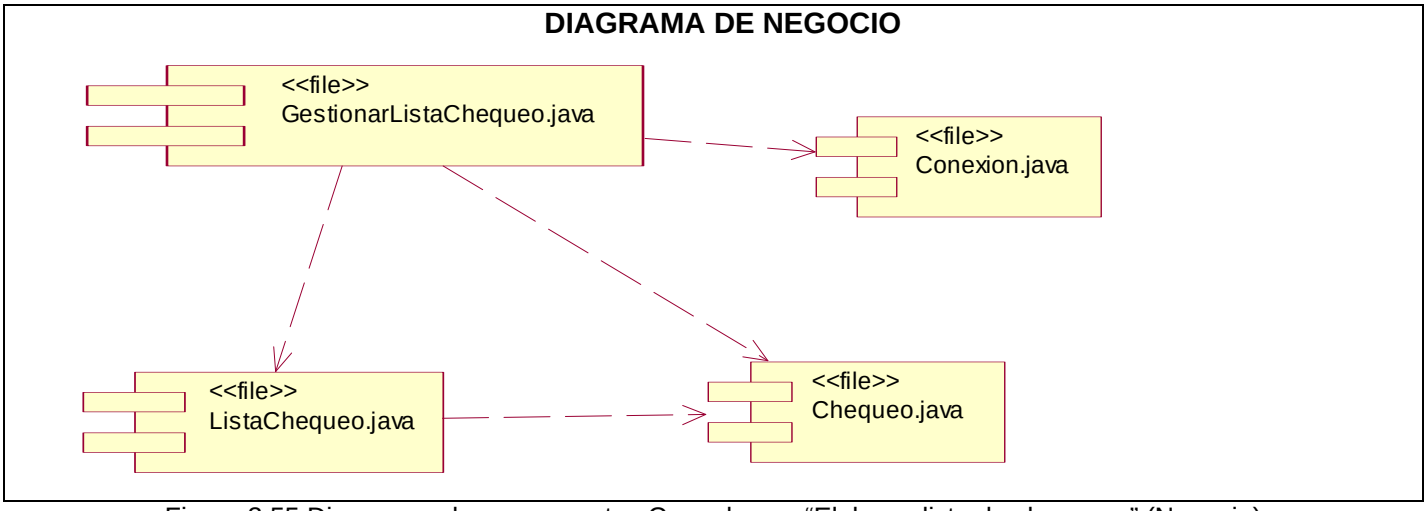


Figura 3.55 Diagramas de componentes Caso de uso “Elaborar lista de chequeos” (Negocio)

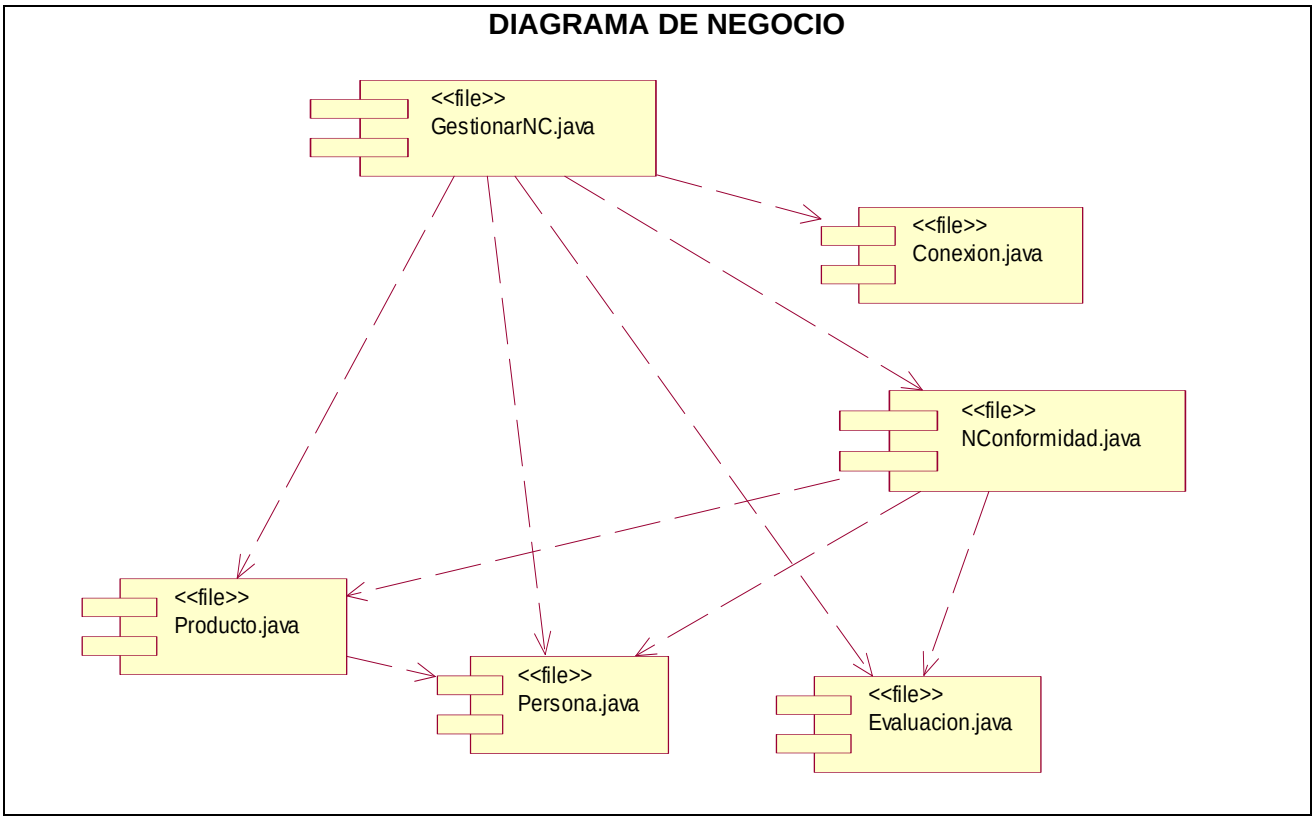


Figura 3.56 Diagramas de componentes Caso de uso “Gestionar no conformidades” (Negocio)

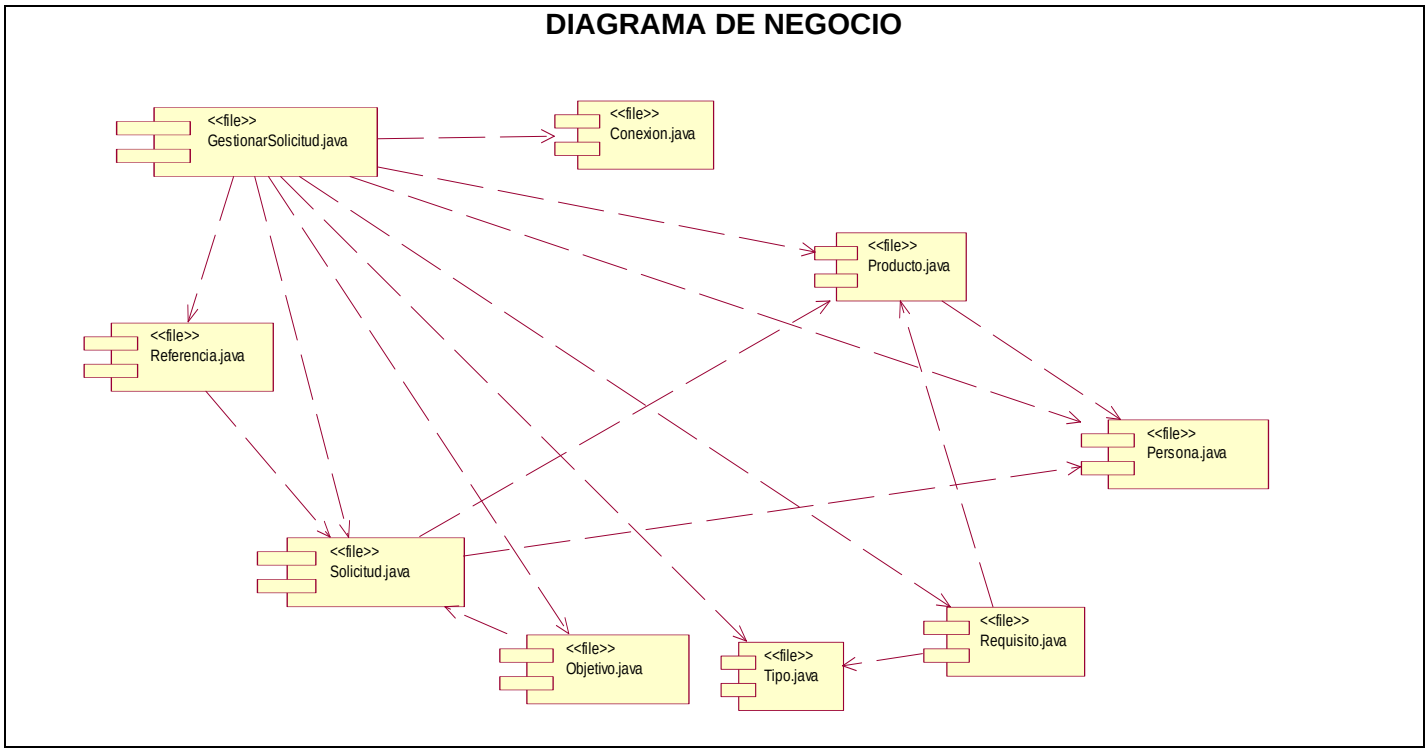


Figura 3.57 Diagramas de componentes Caso de uso “Realizar solicitud” (Negocio)

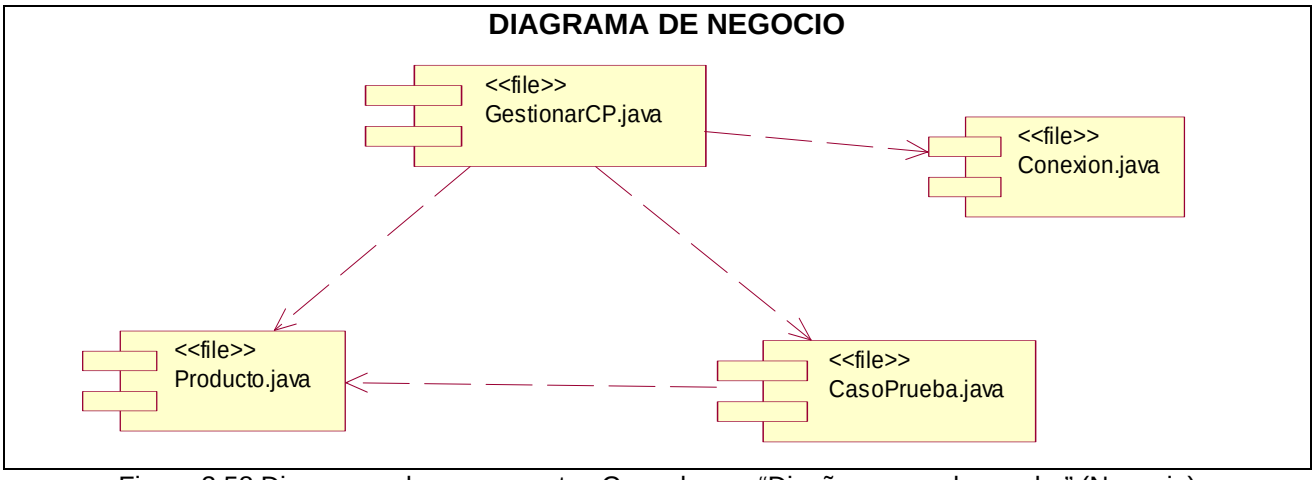


Figura 3.58 Diagramas de componentes Caso de uso “Diseñar casos de prueba” (Negocio)

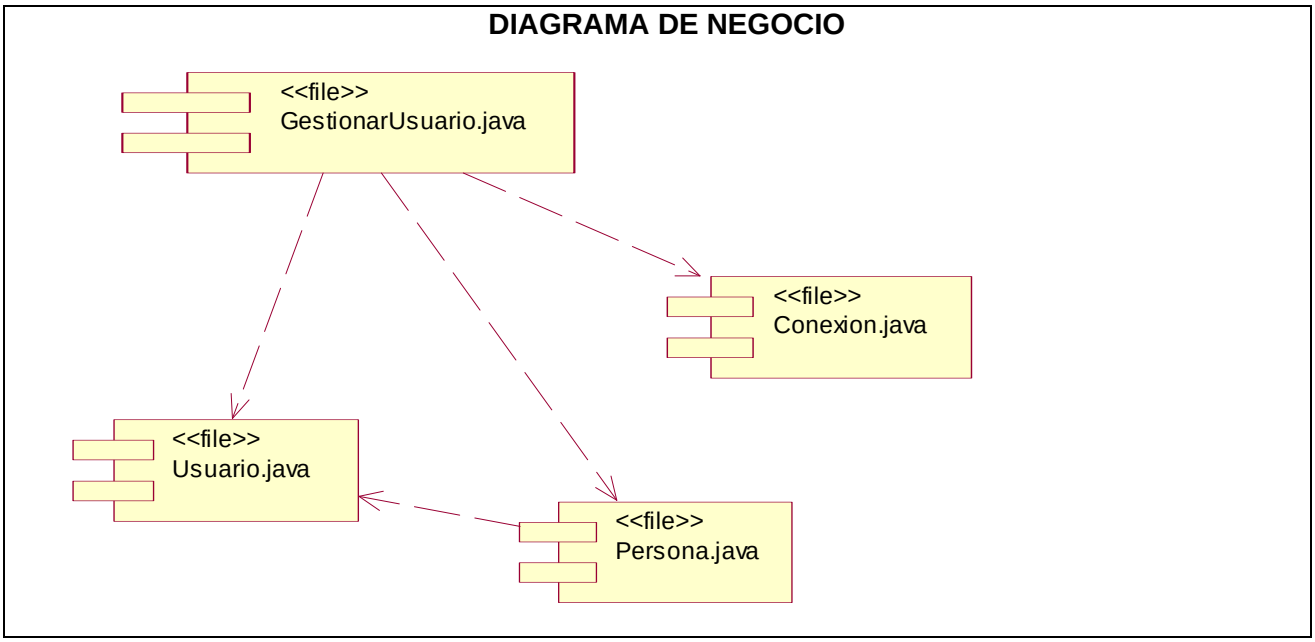


Figura 3.59 Diagramas de componentes Caso de uso “Crear usuario” (Negocio)

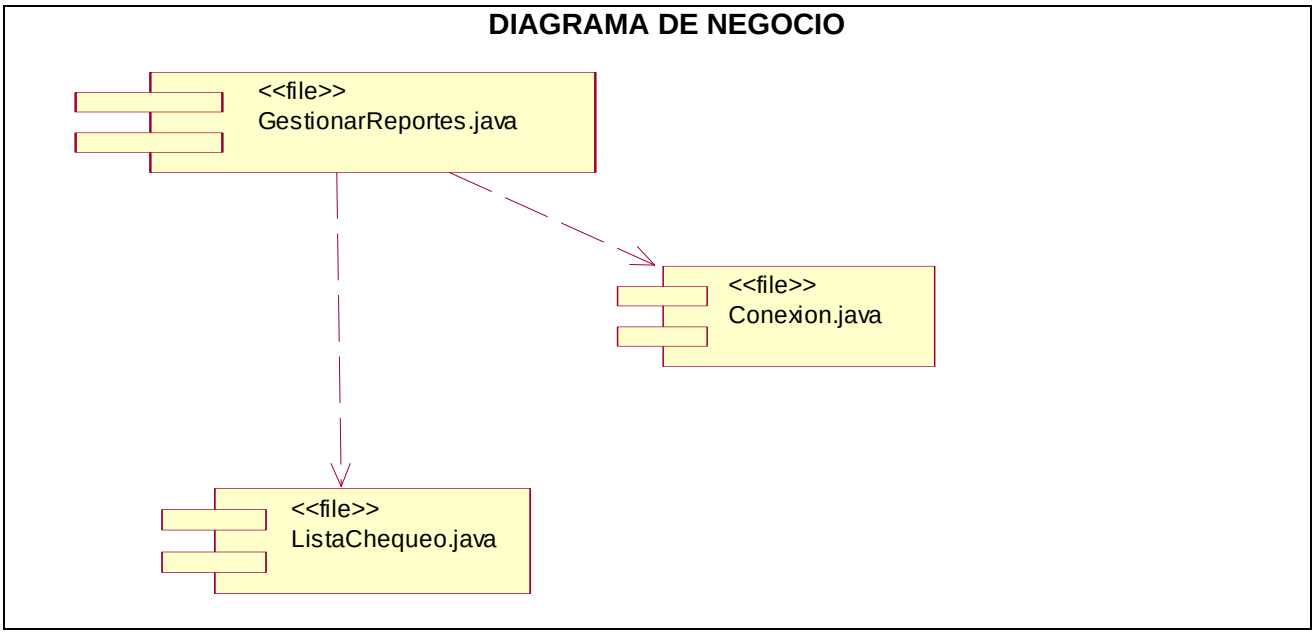


Figura 3. 60 Diagramas de componentes Caso de uso “Realizar reporte” (Negocio)

3.5 Conclusiones

En este capítulo se realizó el análisis y diseño del sistema que en un futuro se desea construir, todos los diagramas; de análisis, de secuencia del diseño, de clases del diseño y web, de clases persistentes, el modelo de datos, de despliegue y componentes son la base fundamental e indispensable para comprender como va a funcionar el Sistema Asistente para gestionar pruebas de caja negra y como se desea hacer.

CONCLUSIONES

Finalizado el trabajo la investigación que se realizó contribuye notablemente a la Dirección de Calidad y Normas de la UCI, aportando un prototipo, análisis y diseño de un Sistema Asistente para gestionar pruebas de caja negra, que garantizará de forma eficiente la realización de las pruebas de caja negra. Se puede decir que se le dio cumplimiento a todos los objetivos propuestos: se conceptualizó sobre modelos y estándares de calidad, calidad de software, pruebas, prototipos, tendencias, tecnologías y herramientas; gracias a ello se pudo, entre otras cosas, determinar las características con las que cuenta el prototipo y seleccionar las herramientas con las que se confeccionó; por otra parte se identificaron los elementos que componen el proceso de pruebas llevadas a cabo en la UCI, esto proporcionó la información y experiencia necesaria para la elaboración del prototipo; se obtuvieron los requisitos primarios, lo que permitió que se construyera el prototipo de un Sistema Asistente para gestionar pruebas de caja negra y finalmente se obtuvo el análisis y diseño de un Sistema Asistente para Gestionar Pruebas de Caja Negra mediante diferentes diagramas como son: los diagramas de secuencia del diseño, los diagramas de clases del diseño y web por mencionar algunos.

RECOMENDACIONES

- Que se realice la implementación de las funcionalidades descritas en el análisis y diseño del sistema.
- Que se utilicen para la implementación las herramientas, gestor de base de datos y lenguaje propuestos.
- Que se haga una ayuda en el sistema.

REFERENCIA BIBLIOGRÁFICA

[1] Pressman, Roger S. Software Engineering: A Practitioner's Approach, 2000, [disponible en: http://www.rspa.com/about/sepa.html]

[2] Shrum, Sandy. CMMI. Guidelines for Process Integration and Product Improvement, Addison Wesley, 2003, 688 pp, Boston

[3] Universidad de Córdoba. Normas y Estándares. Desarrollo de Aplicaciones, Mayo 1993. [Disponible en: http://www.gestion.uco.es/gestion/aplicaciones/docs/NormasyEstandares.pdf]

[4] Pressman, Roger S. Software Engineering: A Practitioner's Approach, 2000, [disponible en: http://www.rspa.com/about/sepa.html]

[5] Fernández, Oscar M. Un enfoque actual sobre la calidad del software, 1995. [Disponible en: http://bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm]

[6] Jacobson, Booch, Rumbaugh. El Proceso Unificado de Desarrollo de Software, Madrid, 2000, Pearson Education S.A.

[7] Hutcheson, Marnie L. Software Testing Fundamentals: Methods an Metrics, 2003 [Disponible en: http://dcc.cucei.udg.mx/academias/planes/cc4/Plan_de_Estudios_Pruebas_de_Software.doc.]

[8] Jacobson, Ivar. El Proceso Unificado de Desarrollo de Software, Madrid,Pearson Education S.A, 2000,

[9] Booch, George. El proceso Unificado de Desarrollo de Software, Madrid, Pearson Education S.A, 2000.

[10] Teruel, Alejandro El Plan de Pruebas, 2001, [Disponible en: http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html]

[11] Booch, George. El proceso Unificado de Desarrollo de Software, Madrid, Pearson Education S.A, 2000.

[12] Pressman, Roger S. Ingeniería de software.Un enfoque Practico, Madrid, McGraw Hill,2003,5,605.

[13] Ramirez, Jaime. Métodos de Prueba del Software. Unidad de Programación,2007, [Disponible en: http://www.lml.ls.fi.upm.es]

[14] Mañas, José A. *Pruebas de Programas*, 1994. [Disponible en: http://www.it.uc3m.es/tsps/testing.htm]

[15] Beizer,Boris. Black Box Testing,2002, [Disponible en: http://www.webopedia.com/TERM/B/Black_Box_Testing.html]

[16] Myers, David G. "The art of software testing", Wiley. 1979.

[17] Pressman, Roger S. "Software Engineering: a practitioner's approach", European Edition. McGraw Hill. 1997.

[18] Ramírez, Javier. Métodos de Prueba del Software Unidad de Programación. [Disponible en: http://lml.ls.fi.upm.es/~jramirez/ed2/pruebas.ppt]

[19] Nielsen, Jakob. Prototipos, 2006, [Disponible en: http://www.albertolacalle.com/hci_prototipos.htm]

[20] Granollers, Toni. ¿Qué es un Prototipo? Octubre 2005 [Disponible en: http://griho.udl.es/mpiua/index.htm]

[21] Maner, Walter. Prototipado, 1997, [Disponible en: http://www.sidar.org/recur/desdi/traduc/es/visitable/maner/Prototipado.htm#car]

[22] Manchón, Eduardo. Prototipos en diseño web: creación y evaluación,, 1999 [Disponible en: http://www.alzado.org/articulo.php?id_art=109]

[23] James Garrett, Jesse. AJAX un nuevo acercamiento a aplicaciones Web, mayo 28 del 2005. [Disponible en: http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php]

BIBLIOGRAFÍA

1. Alvarez, Miguel Angel, Artículos en DesarrolloWeb, 2006 [Disponible en: http://www.desarrolloweb.com/articulos/541.php]
2. Aula con click , Curso de Dreamweaver 8, Agosto 2006 [Disponible en: http://www.aulaclic.es/dreamweaver8/t_1_1.htm]
3. Booch, Rumbaugh, Jacobson. El Proceso Unificado de Desarrollo de Software. Madrid, Pearson Education S.A, 2000.
4. Bryson, Brian. Bridging the gap between black box and white box testing, 2003 [Disponible en: http://www-128.ibm.com/developerworks/rational/library/1147.html].
5. Calidad.com. Historia de la informática, 2007 [Disponible en: http://www.calidad.com.mx/articulos.asp?art=15]
6. Clases de Ingienería del Software I, curso 2005-2006, UCI.
7. CSAE, Normas y estándares aplicables y legislación vigente en materia de las tecnologías de la informática. 2006 [Disponible en: http://www.csae.map.es/csi/silice/Tramit61.html]
8. Dr. Fernández Sanz, Luís. Revista Española de Innovación, Calidad e Ingeniería del Software, 2005 [Disponible en: http://www.ati.es/IMG/pdf/PatriciaRodriguezNum2Vol1.pdf]
9. Fernández Carrasco, Oscar M. Un enfoque actual sobre la calidad del software. Un enfoque actual sobre la calidad del software, 1995 [Disponible en: http://bvs.sld.cu/revistas/aci/vol3_3_95/aci05395.htm.]
10. Franco García, Ángel. Programación en el lenguaje java, 2003 [Disponible en: http://lenguajes-de-programacion.com/programacion-java.html]
11. García Romero. Modelos y estándares de calidad de software, 2001 [Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/garcia_r_ci/capitulo2.pdf]
12. Gutierrez, Javier J. Index of in2test.lsi.uniovi.es. Index of in2test.lsi.uniovi.es, 2007 [Disponible en: http://in2test.lsi.uniovi.es/pris2006/PRIS2006-GutierrezEscalonaMejiasTorres.pdf.]
13. Intellia Technology. Aplicaciones Web a la medida, 2006 [Disponible en: http://www.intellia.com.mx/esp/servicios/aplicaciones_web_a_la_medida.php]
14. Konrad , Shrum, Chrissis. CMMI, Guidelines for Process Integration and Product Improvemente, Addison Wesley, 2003, 688 pp, Boston.

15. Mañas, José A. Pruebas de Aceptación, 2004 [Disponible en: http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/aceptacion.htm.]
16. Mañas, José A. Pruebas de Programas, 1994 [Disponible en: http://www.it.uc3m.es/tsps/testing.htm]
17. Pressman, Roger S. Ingeniería del Software. Un enfoque práctico, McGraw-Hill, 2002, 640 pp.
18. Ramírez, Jaime. Métodos de Prueba del Software, 2006 [Disponible en: http:// www.lml.ls.fi.upm.es/~jramirez/ed2/pruebas.ppt]
19. RedSauce. Pruebas de Hardware y Software, 2005 [Disponible en: http://www.redsauce.net/index.php?folderID=98]
20. Softonic , Hay otros programas de retoque fotográfico pero ninguno como Photoshop, 2005 [Disponible en: http://adobe-photoshop.softonic.com/]
21. Sommerville, Ian, Software Engineering, 2007 [Disponible en: http://www.e-market.cl/dir/umayor/ingsw/cap08.ppt.]
22. Valverde, Jesús. Sistemas de bases de datos,2004 [Disponible en: http://www.unex.es/didactica/Tecnologia_Educativa/info03G.htm]

GLOSARIO DE TÉRMINOS

Calidad: Calidad de software: Satisfacción de las necesidades de los usuarios.

Caso de prueba: Especificación de un caso para probar el sistema, incluyendo que probar, con que entradas y resultados y bajo qué condiciones.

Cliente: Persona, organización o grupo de personas que encarga la construcción de un sistema, ya sea empezarlo desde cero, o mediante el refinamiento de versiones sucesivas.

Interfaz: Una colección de operaciones que se usan para especificar el servicio de una clase o de un componente. Un juego nombrado de operaciones que caracterizan la conducta de un elemento.

Lista de Chequeo: Una serie de pasos ó procedimiento que se deben seguir de forma precisa, pueden ser útil para comprobar un producto o proceso.

Prueba: Ejecución de un programa para verificar si cumple con lo establecido.

Requisito funcional: Requisito que especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas: requisito que especifica comportamiento de entrada y salida de un sistema.

Requisito No funcional: Requisito que especifica propiedades del sistema, restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, mantenibilidad, extensibilidad o fiabilidad. Requisito que especifica restricciones físicas sobre un requisito funcional.

Runtime: Es el período de tiempo que dura la ejecución de un programa.

Usuario: Persona que utiliza normalmente el software.