UNIVERSIDAD NACIONAL SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA DE FORMACIÓN PROFESIONAL DE INGENIERÍA DE SISTEMAS

"SOFTWARE DE TRAZABILIDAD DE REQUISITOS PARA EL PROCESO ICONIX, 2012"

Tipo de investigación : Aplicada
Área de investigación : Ingeniería de Software
Ejecutor : Bach. Ing. TORRES LOZANO, Juan Carlos

AYACUCHO – PERÚ

2012

"SOFTWARE DE TRAZABILIDAD DE REQUISITOS PARA EL PROCESO ICONIX, 2012"

RECOMENDADO : 26 DE MARZO DEL 2013

APROBADO : 18 DE JULIO DEL 2013

MSc. Ing. CARLOS A. PRADO PRADO
(Presidente)

Ing. EDITH F. GUEVARA MOROTE
(Miembro)

Ing. ELINAR CARRILLO RIVEROS
(Miembro)

Ing. JENNIFER R. PILLACA DE LA CRUZ
(Secretario Docente)

Según el acuerdo constatado en el Acta, levantada el 18 de julio del 2013, en la Sustentación de Tesis presentado por el Bachiller en Ingeniería de Sistemas Sr. Juan Carlos TORRES LOZANO, con el Trabajo Titulado "SOFTWARE DE TRAZABILIDAD DE REQUISITOS PARA EL PROCESO ICONIX, 2012", fue calificado con la nota de QUINCE (15) por lo que se da la respectiva APROBACIÓN.

MSc. Ing. CARLOS A. PRADO PRADO
(Presidente)

Ing. EDITH F. GUEVARA MOROTE
(Miembro)

Ing. ELINAR CARRILLO RIVEROS
(Miembro)

Ing. JENNIFER R. PILLACA DE LA CRUZ
(Secretario Docente)

DEDICATORIA

A mis padres por sus enseñanzas y los valores que me inculcaron, a mis hermanos por su ejemplo, apoyo incondicional, a mis maestros que me han ilustrado en mi vida académica, a mis amigos que sin esperar nada a cambio copartieron sus conocimientos y diversión.

AGRADECIMIENTO

A la Universidad Nacional San Cristobal de Huamanga, por darme oportunidad de estudiar Ingeniería de Sistemas y la posibilidad de brillar en la vida.

A los docentes de la Univerisdad que aportaron con sus conocimientos a mi formación académica como Ingeniero de Sistemas.

Gracias a todas las personas que participaron en la investigacion, ya que inviertieron su tiempo y paciencia para ayudarme a completar mi proyecto de tesis.

CONTENIDO

CAPITULO III

METODOLOGÍA DE LA INVESTIGACIÓN

CAPITULO IV

RESULTADOS DE LA INVESTIGACIÓN

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

RESUMEN

El desarrollo de software con el proceso ICONIX, es una actividad que no cuenta con una herramienta de apoyo para la gestión del cambio de los requisitos de manera que automatice la trazabilidad de requisitos para asegurar la consistencia y la completitud de los requisitos. Realizar el seguimiento a los requisitos a lo largo del proceso de desarrollo no es tarea fácil; todo artefacto de software cambia en el tiempo por la evolución en las necesidades de los usuarios. El esfuerzo invertido en mantener las relaciones de traza, obtiene su retorno cuando se van a realizar modificaciones en el software.

El objetivo es desarrollar un software de trazabilidad de requisitos para apoyar el proceso ICONIX, Ayacucho, 2012. A través de la utilización de herramientas JDK (Java Development Kit) con el IDE (Entorno Integrado de Desarrollo) Netbeans, la base de datos postgreSQL y la metodología ICONIX. Con el propósito de mejorar la calidad del producto software desarrollado con el proceso ICONIX, y la finalidad de tener una herramienta para realizar la trazabilidad de requisitos del proceso ICONIX.

Esta investigación fue realizada en la ciudad de Ayacucho, siendo una investigación de tipo no experimental y transeccional descriptiva. Se utilizan las matrices de trazabilidad y como metodología de desarrollo de software con el proceso ICONIX.

Con los resultados obtenidos en el capítulo IV, el software de trazabilidad de requisitos ayuda en la gestión de la transformación y la evolución de los requisitos en las fases análisis, diseño preliminar, diseño, implementación y pruebas del proceso ICONIX.

Palabras clave: Trazabilidad de requisitos, Consistencia, Completitud, Ingeniería de requisitos, Proceso ICONIX.

INTRODUCCIÓN

La trazabilidad es una práctica de la ingeniería de requisitos que facilita el control de los requisitos por medio de vínculos de trazado entre diferentes artefactos que los representan a través del ciclo de vida de desarrollo (Tabares, 2009). La completitud se refiere a la no existencia de información sin declarar o definir, como objetos o entidades, en una especificación de requisitos (Navarro, Orosco, & Jaramillo, 2007). Según Tabares (2009) la completitud es el conocimiento acerca de que cualquier elemento de entrada en los modelos del problema deberá ser incluido en los modelos de la solución. Un modelo es consistente en un nivel de abstracción Y, con respecto a un nivel de abstracción X, si conserva el mismo significado del nivel de abstracción X (Tabares, Arango, & Anaya, 2006). ICONIX establece un procedimiento sencillo y un conjunto mínimo de pasos que suelen dar lugar a muy buenos resultados (Rosenberg & Stephens, 2007).

La razón para desarrollar el software de trazabilidad de requisitos para el proceso ICONIX, es necesario tener buenas técnicas para separar y especificar correctamente los requisitos, controlar su evolución y transformación. La trazabilidad de requisitos es práctica que permite lograr la verificación de la completitud y consistencia de los requisitos.

En la actualidad la industria del software trae consigo altos costos, alta complejidad, dificultades de mantenimiento y una disparidad entre las necesidades de los usuarios y los productos desarrollados. Para minimizar el impacto causado por la evolución de las necesidades del usuario se requiere de una herramienta para hacer el seguimiento la evolución y la transformación de los requisitos.

Los objetivos específicos planteados son; analizar, diseñar, desarrollar y probar la trazabilidad de requisitos para apoyar el análisis, diseño preliminar, diseño

implementación y las pruebas con finalidad de escribir, identificar, refinar o actualizar los requisitos, el modelo de dominio, el borrador de casos de uso, diagrama de robustez, diagrama de secuencia, controladores, modelo de clases, código fuente, pruebas unitarias, pruebas de aceptación y direccionar los requisitos.

El contenido de este documento se estructura en cinco capítulos, la bibliografía y los anexos, que abarcan la totalidad del trabajo de tesis.

El capítulo 1 denominado, Planeamiento de la Investigación presenta un panorama de la situación actual, el enunciado de los problemas, la definición de los objetivos, la hipótesis, la justificación y delimitación de la investigación.

El capítulo 2 denominado, Revisión Literaria contiene los antecedentes de la investigación y el marco teórico de la ingeniería de requisitos, trazabilidad de requisitos, consistencia, completitud y el proceso ICONIX.

El capítulo 3 denominado, Metodología de la Investigación contiene el tipo y diseño de la investigación, la población y muestra seleccionada, la definición de las variables e indicadores, las técnicas e instrumentos de recolección de datos, las técnicas y herramientas para el procesamiento de la información.

El capítulo 4 denominado, Resultados de la Investigación contiene los resultados de la trazabilidad de requisitos y la documentación de los artefactos de desarrollo del software.

El capítulo 5 denominado, Conclusiones y Recomendaciones contiene las conclusiones obtenidas luego de finalizado del trabajo de tesis y las recomendaciones para futuras investigaciones.

El resto del documento presenta las bibliografías utilizadas en el trabajo de tesis y los anexos entre las que figuran las fichas de análisis documental, el modelo de base de datos y el diseño de los reportes.

CAPITULO I
PLANEAMIENTO DE LA INVESTIGACIÓN

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

La realidad de la industria del software de gestión impone la adopción de procesos ágiles de desarrollo para lograr competitividad, ya que el proceso de desarrollo de software trae consigo altos costos, alta complejidad, dificultades de mantenimiento y una disparidad entre las necesidades de los usuarios y los productos desarrollados.

En éste contexto el proceso ICONIX (Rosenberg & Scott, 2007) se define como un proceso de desarrollo de software práctico. ICONIX está entre la complejidad del RUP y la simplicidad y pragmatismo del XP, sin eliminar las tareas de análisis y de diseño que XP no contempla.

Las características fundamentales de ICONIX son: iterativo e incremental, trazabilidad (cada paso está referenciado por algún requisito) y la dinámica de UML (Artefactos). Rosenberg y Scoot destacan como principales tareas de ICONIX el análisis de requisitos, diseño preliminar, diseño, implementación y las pruebas.

> "La trazabilidad en la ingeniería de software es una práctica de control que ayuda a obtener el producto en el dominio de la solución lo más exacto y fiable posible a las necesidades expresadas por el cliente en el dominio del problema" (Tabares, 2006, p. 34).

El proceso de la ingeniería de requisitos utiliza técnicas sistémicas y herramientas para el descubrimiento, documentación y mantenimiento de los requisitos para asegurar que los requerimientos del sistema estén completos y sean consistentes.

Realizar el seguimiento a los requisitos a lo largo del proceso de desarrollo de

software no es tarea fácil; todo artefacto de software cambia en el tiempo por la evolución en las necesidades de los usuarios. Para minimizar el impacto causado por dicha evolución se requiere de una herramienta para hacer el seguimiento al proceso ICONIX, que integre la ingeniería de requisitos y las técnicas de trazabilidad.

1.2. DEFINICIÓN DEL PROBLEMA

1.2.1. PROBLEMA PRINCIPAL

¿Cómo Implementar la trazabilidad de requisitos para apoyar el proceso ICONIX, Ayacucho, 2012?

1.2.2. PROBLEMAS SEGUNDARIOS

a. ¿Cómo realizar la trazabilidad de requisitos durante el análisis?

b. ¿Cómo realizar la trazabilidad de requisitos durante el diseño preliminar?

c. ¿Cómo realizar la trazabilidad de requisitos durante el diseño?

d. ¿Cómo realizar la trazabilidad de requisitos durante la implementación?

e. ¿Cómo realizar la trazabilidad de requisitos durante las pruebas?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. OBJETIVO GENERAL

Desarrollar un software de trazabilidad de requisitos para apoyar el proceso ICONIX, Ayacucho, 2012. A través de la utilización de herramientas JDK (Java Development Kit) con el IDE (Entorno Integrado de Desarrollo) Netbeans, la base de datos postgreSQL y la metodología ICONIX. Con el propósito de mejorar la calidad del producto software desarrollado con el proceso ICONIX, y la finalidad de tener una herramienta para realizar la trazabilidad de requisitos del proceso ICONIX.

1.3.2. OBJETIVOS ESPECÍFICOS

a. Analizar, diseñar, desarrollar y probar la trazabilidad de requisitos para apoyar el análisis con la finalidad de refinar los requisitos y el modelo de dominio.

b. Analizar, diseñar, desarrollar y probar la trazabilidad de requisitos para apoyar el diseño preliminar con la finalidad de mejorar los diagramas de robustez, actualizar el modelo de dominio y el borrador de casos de uso.

c. Analizar, diseñar, desarrollar y probar la trazabilidad de requisitos para apoyar el diseño con la finalidad de actualizar el modelo de dominio, realizar los diagramas de secuencia, limpiar el modelo de clases e identificar los controladores.

d. Analizar, diseñar, desarrollar y probar la trazabilidad de requisitos para apoyar la implementación con la finalidad de mejorar el modelo de dominio, escribir el código y las pruebas unitarias.

e. Analizar, diseñar, desarrollar y probar la trazabilidad de requisitos para apoyar las pruebas con la finalidad de escribir las pruebas y direccionar los requisitos.

1.4. HIPÓTESIS DE LA INVESTIGACIÓN

Si desarrollamos el software de trazabilidad de requisitos entonces apoyamos el desarrollo de software con el proceso ICONIX, Ayacucho, 2012.

1.5. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.5.1. IMPORTANCIA Y JUSTIFICACIÓN

El desarrollo de software con metodologías agiles es una actividad caótica caracterizada por la frase "codifica y corrige", con la posibilidad de disminuir el costo del cambio a lo largo del proyecto. El fundamento básico de cualquier software recae sobre sus requisitos. El éxito del software depende de cómo se hayan capturado, entendido y usado los requisitos como base para el desarrollo. Los requisitos definen las propiedades y la estructura del software y de su buen uso determinan el costo del desarrollo del software. Por lo tanto, un producto software de calidad tienen muy claro lo que pretende conseguir y cómo obtenerlo, para lo cual se debe satisfacer las condiciones que en este contexto son los requisitos del producto, para lo cual la trazabilidad de requisitos es muy importante en el desarrollo de software.

Esta investigación generará reflexión y discusión sobre los conceptos, técnicas y herramientas de la ingeniería de requisitos y la trazabilidad de requisitos en el desarrollo de software con el proceso ICONIX.

En lo práctico, esta investigación propone al problema planteado una estrategia de acción que al aplicarla como se hará en el capítulo cuarto de

esta investigación, contribuirá a resolverlo y permitirá sentar las bases para otros estudios que surjan partiendo de la problemática aquí especificada.

En lo metodológico, esta investigación generará la integración de las técnicas e instrumentos de la ingeniería de requisitos, la trazabilidad de requisitos y el proceso ICONIX mediante una herramienta que automatice la trazabilidad de requisitos en el desarrollo de software.

Por otra parte, en cuanto a su alcance, esta investigación abrirá nuevos caminos para empresas que presenten situaciones similares a la que aquí se plantea, sirviendo como marco referencial a estas.

1.5.2. DELIMITACIÓN

A. DELIMITACIÓN CONTEXTUAL

El presente trabajo de investigación está dirigido a la integración de las técnicas de trazabilidad de requisitos para el desarrollo de software con el proceso ICONIX.

B. DELIMITACIÓN TEMPORAL

El horizonte temporal será comprendido entre el mes de Abril hasta el mes de Diciembre del 2012.

CAPITULO II
REVISIÓN LITERARIA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Anaya y Letelier (2002) Trazabilidad de Requisitos Adaptada a las Necesidades del Proyecto: Un Caso de Estudio Usando Alternativamente RUP y XP, presenta SharpTrace una herramienta que implementa mediante un metamodelo de trazabilidad compuesta de un conjunto de tareas que guían al usuario. Para ello se consideran los múltiples tipos de enlace de trazabilidad, así como los atributos de calidad de cada tipo de artefacto.

Navarro, Orosco y Jaramillo (2005) en el trabajo "Analizador de Completitud de Requisitos Escritos en Español Restringido" se presenta una herramienta que implementa un método para mejorar la completitud de los requisitos utilizando el enfoque lingüístico, basado en casos semánticos asociado al sentido de un verbo. Los resultados obtenidos permiten retroalimentar el proceso para el propósito de mejorar la completitud de las especificaciones de requisitos y, en última instancia, para el mejoramiento de la calidad del software.

Tabares, Arango y Anaya (2006) en el articulo "Una Revisión de Modelos y Semánticas para la Trazabilidad de Requisitos", concluyen que es conveniente realizar la práctica de la trazabilidad teniendo en cuenta factores tales como: complejidad del software, nivel de especificación de los requisitos y de los modelos, identificación de dependencias entre los requisitos, lenguaje de modelado utilizado para representar los requisitos, empleo de artefactos para la reutilización, nivel de soporte de los modelos y las estructuras para las pruebas y los cambios, dependencias entre los elementos de cada modelo y nivel de abstracción, lenguaje de representación utilizado en los modelos, entre otros.

Zapata, Villegas y Arango (2006) "Reglas de Consistencia entre Modelos de Requisitos de un Método", concluye que la consistencia constituye un nuevo

campo de estudio que cobra mayor importancia en el ámbito de la ingeniería de software, dado que los métodos de desarrollo de software se componen de artefactos que pueden ser desarrollados por diferentes personas (generando puntos de vista diferentes) y por la evolución de los requisitos del interesado. La consistencia debe ser un aspecto fundamental a tener en cuenta en cualquier método de desarrollo de software, porque de ella depende en gran medida la calidad de los resultados que se obtienen.

Tabares, Barrera, Arroyave y Pineda (2007) en el artículo "Un Método para La Trazabilidad de Requisitos en el Proceso Unificado de Desarrollo", para poder automatizar la evolución de los requisitos y los artefactos en diferentes niveles de granularidad (o fases) hace que la trazabilidad esté totalmente orientada a la propagación de los cambios tanto hacia adelante (forward) como hacia atrás (backward) en el proceso de desarrollo, a la evaluación del impacto del cambio y a la verificación de la completitud y consistencia de los modelos de desarrollo.

INTECO (2008) "Guía Práctica de Gestión de Requisitos", menciona que el coste de una buena recogida de requisitos y análisis del sistema a desarrollar es menor comparado con el coste resultante de tener requisitos pobres, es decir, el coste de reparar productos deficientes o de poca calidad, el coste de los proyectos cancelados y el coste de haber perdido la oportunidad de tener el producto correcto en el momento correcto.

2.2. MARCO TEÓRICO

2.2.1. INGENIERÍA DE REQUISITOS

La necesidad de una ingeniería de requisitos dentro de la ingeniería del software es imprescindible para obtener productos de calidad. "Ingeniería de Requerimientos se utiliza para definir todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requerimientos para un producto determinado" (Gomez, 2006, pág. 2).

"La ingeniería de requisitos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán" (Pressman, 2005, pág. 155).
La calidad no puede implementarse en las etapas finales de los procesos de

desarrollo, sino que es una característica intrínseca al propio producto, iniciándose en las especificaciones del producto. La ingeniería de requisitos es la base para el aseguramiento de la calidad del proyecto.

A. REQUISITO

"Un requerimiento es un atributo necesario dentro de un sistema, que puede representar una capacidad, una característica o un factor de calidad del sistema de tal manera que le sea útil a los clientes o a los usuarios finales" (Gomez, 2006, pág. 5).

"Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas" (Sommerville, 2005, pág. 122).

Según (INTECO, 2008, pág. 6) "un requisito es algo que el producto debe hacer o una característica que debe tener. Un requisito existe por el tipo de demanda que tiene el producto o porque el cliente quiere que el requisito sea parte del producto entregado".

La tarea de todo analista de requisitos es hablar con el cliente, escuchar y entender lo que necesita.

B. CLASIFICACIÓN DE LOS REQUISITOS

b.1. REQUISITOS FUNCIONALES

"Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares" (Sommerville, 2005, pág. 109).

Por su parte (León, 1996, pág. 44) define los requisitos funcionales como "aquellos ligados a la relación entre datos de entrada y resultados (datos de salida) que debe presentar el sistema, incluidos los derivados de restricciones temporales cuando éstas están cuantificadas".

En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.

b.2. REQUISITOS NO FUNCIONALES

"Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares" (Sommerville, 2005, pág. 109).

En cambio (León, 1996, pág. 44) manifiesta que los requisitos funcionales son aquellos que incluyen todos los aspectos de calidad del sistema: por ejemplo, mantenibilidad (facilidad para que el sistema evolucione y se modifique una vez entregado al usuario), escalabilidad (posibilidad de incrementar sustancialmente el número de usuarios u otros parámetros), facilidad de uso, etc., que no pueden ligarse a funciones concretas dentro del sistema.

Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Regularmente apenas se aplican a características o servicios individuales del sistema.

C. TAREAS DE LA INGENIERÍA DE REQUISITOS

La meta de las tareas de ingeniería de requisitos es crear y mantener un documento de requisitos del sistema. "El proceso de la ingeniería de requisitos se lleva a cabo a través de siete distintas funciones: inicio, obtención, elaboración, negociación, especificación, validación y gestión" (Pressman, 2005, pág. 157).

Mientras que Sommerville (2005) identificó como procesos la evaluación de si el sistema es útil para el negocio (estudio de viabilidad); el descubrimiento de requerimientos (obtención y análisis); la transformación de estos requerimientos en formularios estándar (especificación), y la verificación de que los requerimientos realmente definen el sistema que quiere el cliente (validación)" (pág. 130).

Algunas de estas tareas de la ingenieria de requisitos se realizan de manera paralela con la finalidad de entregar especificacion de requisitos de software consitente y completa.

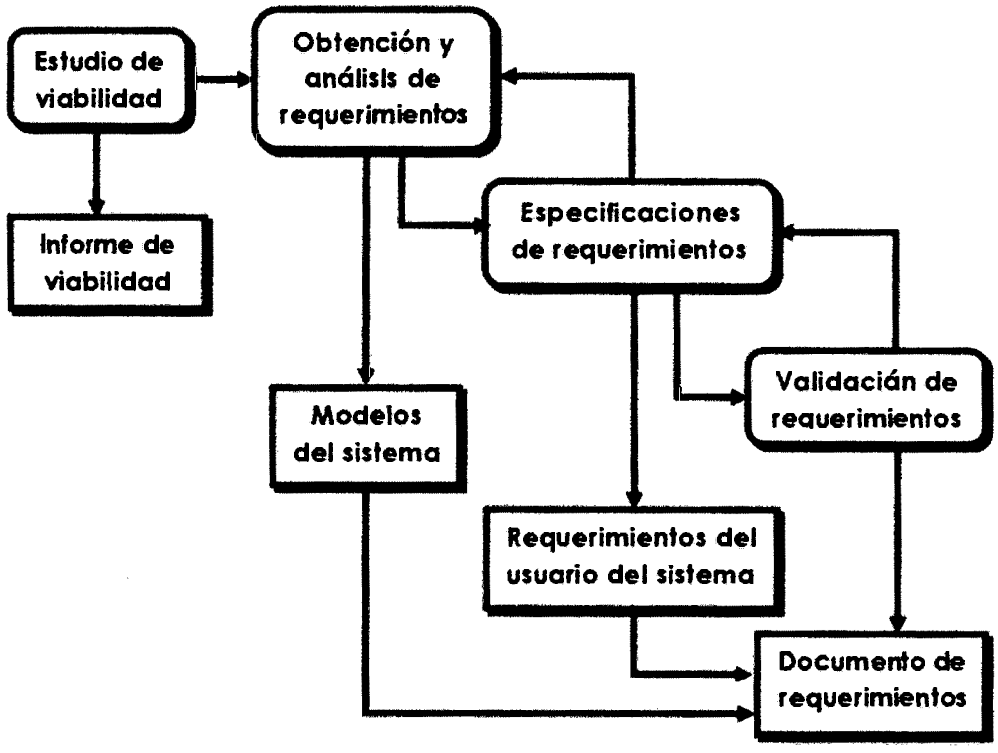


Figura N° 2.1: El proceso de ingeniería de requerimientos (Sommerville, 2005, pág. 130).

2.2.2. TRAZABILIDAD DE REQUISITOS

"Se refiere al soporte que un enfoque provee para controlar la evolución de los requisitos desde fases tempranas de desarrollo hasta la implementación" (Londoño, Anaya, & Tabares, 2008, pág. 43).

"La trazabilidad trata de relacionar cada requisito del sistema con su caso de uso, con su elemento de diseño y con el componente final que implementará dicho requisito" (UPM, 2009, pág. 3).

"la trazabilidad es una práctica de la Ingeniería de Requisitos que facilita el control de los requisitos por medio de vínculos de trazado entre diferentes artefactos que los representan a través del ciclo de vida de desarrollo" (Tabares, 2009, pág. 37).

La trazabilidad es un aspecto imprescindible en el modelado, los requisitos deben ser trazables. Según INTECO (2008, pág. 15) "un requisito es trazable si se pueden identificar todas las partes del producto existente relacionadas con ese requisito".

Los requisitos deberían ser trazables para mantener consistencia y completitud entre los distintos documentos y la para asegurar la calidad en el desarrollo del proyecto.

Es importante conocer aspectos de los requisitos tales como:

a. Su origen (Quién los propuso)
b. Necesidad (Por qué existe)
c. Relación con otros requisitos (Dependencias)
d. Relación con otros elementos (Dependencias)

A. COMPLETITUD DE REQUISITOS

"La completitud se refiere a la no existencia de información sin declarar o definir, como objetos o entidades, en una especificación de requisitos" (Boehm, 1984, citado en Navarro, Orosco, & Jaramillo, 2005).

"La completitud se refiere al grado en que un modelo cumple las necesidades de los usuarios en cualquier nivel de abstracción. Esto se cumple si todos los significados del nivel de abstracción X se hallan en el nivel de abstracción Y, y viceversa" (Tabores, Arango, & Anaya, 2006, pág. 36).

"La completitud es el conocimiento acerca de que cualquier elemento de entrada en los modelos del problema deberá ser incluido en los modelos de la solución" (Tabares, 2009, pág. 47).

La completitud no necesariamente significa que cada elemento producido se conserve, es decir, si el desarrollador determina que algún elemento de un modelo no tiene sentido, dichos elementos deben ser eliminados de los modelos que los representan sin generar incompletitud en el sistema.

La escritura de requisitos que no cumplen con la propiedad de completitud tiene como consecuencia que se omitan, al momento de la abstracción y diseño, elementos estructurales fundamentales en el funcionamiento de un sistema. La incompletitud afecta la calidad del software y se ve reflejado en el

nivel de concordancia del software desarrollado con las necesidades del cliente.

B. CONSISTENCIA DE REQUISITOS

"Un modelo es consistente en un nivel de abstracción Y, con respecto a un nivel de abstracción X, si conserva el mismo significado del nivel de abstracción X" (Tabares, Arango, & Anaya, 2006, pág. 36).

"La consistencia de los requisitos consiste en mantener su significado a través de los modelos que los representan en diferentes niveles de abstracción. Una teoría es consistente si ésta nunca prueba una contradicción" (Tabares, 2009, pág. 45).

La consistencia o caherencia de requisitos significa que no hay contradicciones ni acoplamientos redundantes. Un modelo es consistente si na hay cantradicciones entre requisitas. Al hacer una verificación de consistencia es posible clasificar los resultados en tres niveles:

a. Consistencia total: ocurre cuando todas los elementos que representan un requisita, interés u objetivo del sistema mantienen las relaciones y el significada entre ellos.

b. Semi-consistencia: acurre cuando parte de los elementos que representan un requisito, interés u objetiva del sistema mantienen las relaciones y el significado entre ellos.

c. Inconsistencia: ocurre cuando los elementos que representan un requisito no están semánticamente relacianadas dentro del dominio del modelo.

Una inconsistencia puede ser que más de un casa de uso realice al mismo requisito, un prototipo no realice a ningún caso de uso, un requisito no sea trazada a ningún caso de uso, una interfaz na trace ninguna clase del análisis.

2.2.3. MODELO DE TRAZABILIDAD DE LINDVALL

Presenta un modelo que ofrece diferentes tácticas para realizar el trazado entre artefactos de software que resultan del procesa software en tres niveles de abstracción: el dominio, el análisis y el diseño. Aplica la trazabilidad

hacia delante *(forward)* y hacia atrás *(backward)*, vertical y horizontal. Distingue entre artefactos permanentes y temporales para jerarquizar el trazado.

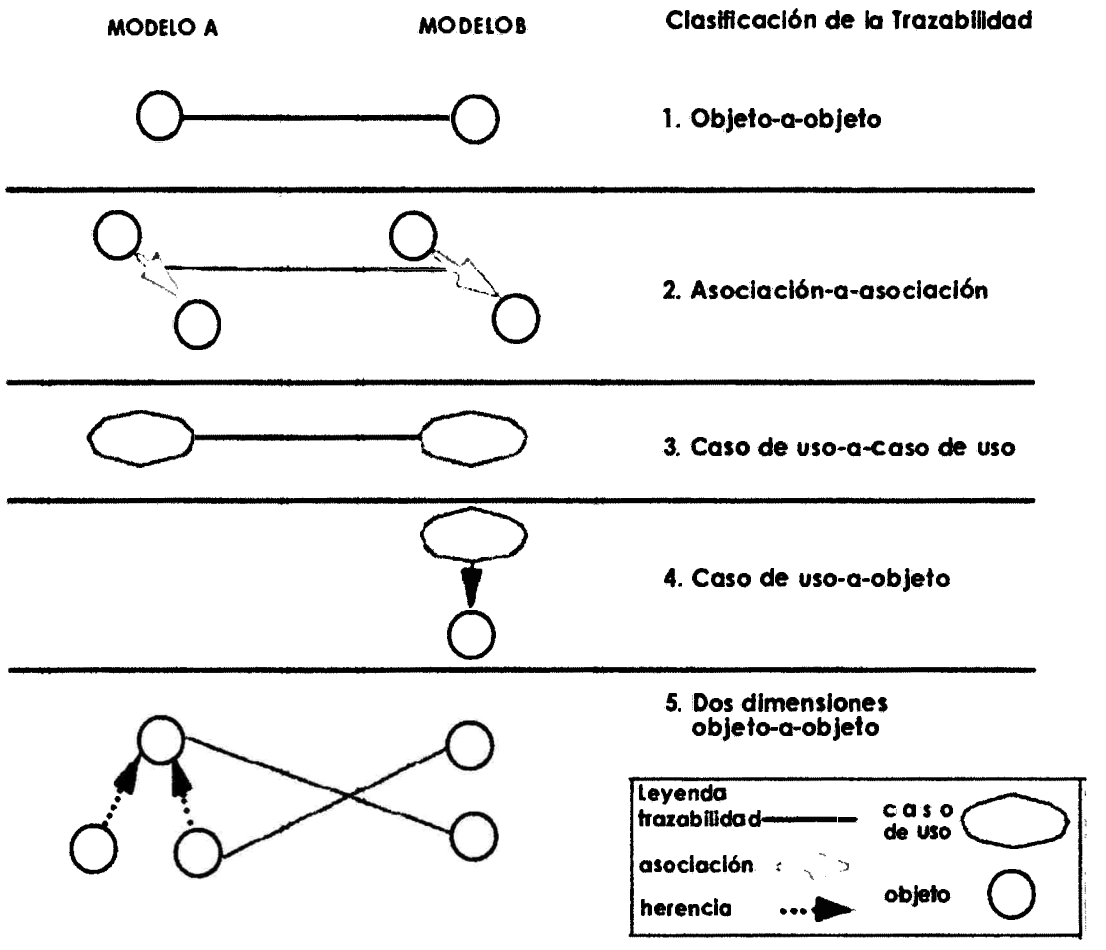


Figura N° 2.2: Clasificación de la trazabilidad (Lindvall, 1996).

DIMENSIONES DE LA TRAZABILIDAD

a. En la primera, se identifican los vínculos de trazados entre los diferentes elementos del modelo, por ejemplo: de objeto a objeto *(object-to-object)*, relación a relación *(association-to-association)*, caso de uso a caso de uso (use-case-to-use-case), requisitos a casos de uso, etc.

b. En la segunda, se determina cómo se realiza el trazado: por vínculos explícitos (o relaciones de los modelos de desarrollo), por referencia, por nombre o por la experiencia del desarrollador y el conocimiento que tenga del dominio y del sistema para distinguir otros ítems que estarían relacionados.

Verificación de consistencia; evalúa en el modelo de desarrollo la posibilidad que haya algún ítem o conjunto de ítems y de asociaciones que se contradigan entre sí o que de una u otra forma hagan el modelo inconsistente. Procura la consistencia a partir del trazado por nombre. La inspección es informal y no tiene forma de validar si todos los ítems y asociaciones están involucrados en la inspección.

Verificación de Completitud; evalúa si el material de especificación de requisitos utilizados en los diferentes niveles de abstracción está completo. Es una inspección intuitiva que depende de la importancia que cada participante les dé a los documentos producidos en el ciclo de vida.

2.2.4. TÉCNICAS DE TRAZABILIDAD

El uso de matrices de trazabilidad es una buena técnica para llevar a cabo esta actividad de forma eficiente. A continuación se presentan algunas matrices de trazabilidad más comunes.

A. MATRIZ DE TRAZABILIDAD DE REQUISITOS Y CASOS DE USO

Esta matriz permite verificar en qué casos de uso son representados y especificados los requisitos identificados en el dominio del problema.

Casos de Uso / Requisitos	CU1	CU2	CU3	CU4
Requisito 1	✓			✓
Requisito 2		✓		
....				
Requisito n	✓			

Tabla N° 2.1: Matriz de trazado, requisitos y casos de uso (Tabores, Barrera, Arroyave, & Pineda, 2007, pág. 75).

También se pueden representar de diversas maneras como la matriz de trazabilidad de casos de uso y requisitos, pero se teniendo en cuenta la consistencia de requisitos y casos de uso, es decir, un caso de uso no puede implementar más de un requisito funcional, en cambio un requisito no

funcional puede ser implementado en uno o más casos de uso.

REQUERIMIENTO	CASO DE USO
RF-001	CU-001
RF-002	CU-002
RF-003	CU-003
RF-004	CU-004
RF-005	CU-005
RF-006	CU-006
RF-007	CU-007
RNF-001	No aplicable
RNF-002	CU-006 CU-007
RNF-003	No aplicable
RNF-004	No aplicable
RNF-005	No aplicable

Tabla N° 2.2: Matriz de trazabilidad, requerimientos-casos de uso (Maniasi, 2005, pág. 359).

CASO DE USO	REQUERIMIENTO
CU-001	RF-001
CU-002	RF-002
CU-003	RF-003
CU-004	RF-004
CU-005	RF-005
CU-006	RF-006 RNF-002
CU-007	RF-007 RNF-002

Tabla N° 2.3: Matriz de trazabilidad, casos de uso-requerimientos (Maniasi, 2005, pág. 359).

B. MATRIZ DE TRAZABILIDAD DE CASOS DE USO-INTERFACES

Permite realizar la trazabilidad entre los casos de uso identificados y las interfaces definidas.

CASO DE USO	INTERFAZ
CU-001	INT-001 INT-002 INT-003 INT-004

	INT-008
CU-002	INT-005 INT-006 INT-007 INT-012
CU-003	INT-012 INT-013 INT-016
CU-004	INT-006 INT-007 INT-008 INT-009
CU-005	INT-010 INT-011
CU-006	INT-014 INT-015
CU-007	INT-015

Tabla N° 2.4: Matriz de trazabilidad, casos de uso-interfaces (Maniasi, 2005, pág. 360).

INTERFAZ	CASO DE USO
INT-001	CU-001
INT-002	CU-001
INT-003	CU-001
INT-004	CU-001
INT-005	CU-001 CU-002
INT-006	CU-002 CU-004
INT-007	CU-002 CU-004
INT-008	CU-001 CU-004
INT-009	CU-004
INT-010	CU-005
INT-011	CU-005
INT-012	CU-002 CU-003
INT-013	CU-003
INT-014	CU-006
INT-015	CU-006 CU-007
INT-016	CU-003

Tabla N° 2.5: Matriz de trazabilidad, interfaces-casos de uso (Maniasi, 2005, pág. 361).

C. MATRIZ DE TRAZABILIDAD CASOS DE USO-DIAGRAMAS DE ROBUSTEZ

Permite realizar la trazabilidad entre los casos de uso identificados y los diagramas de robustez definidos.

CASO DE USO	DIAGRAMA DE ROBUSTEZ
CU-001	DR-001
CU-002	DR-002
CU-003	DR-003
CU-004	DR-004
CU-005	DR-005
CU-006	DR-006
CU-007	DR-007

Tabla N° 2.6: Matriz de trazabilidad, casos de uso-diagramas de robustez (Maniasi, 2005, pág. 362).

Siguiendo la trazabilidad en ambas direcciones también se puede realizar la trazabilidad entre los diagramas de robustez definidos y los casos de uso identificados.

DIAGRAMA DE ROBUSTEZ	CASO DE USO
DR-001	CU-001
DR-002	CU-002
DR-003	CU-003
DR-004	CU-004
DR-005	CU-005
DR-006	CU-006
DR-007	CU-007

Tabla N° 2.7: Matriz de trazabilidad, diagramas de robustez-casos de uso (Maniasi, 2005, pág. 362).

D. MATRIZ DE TRAZABILIDAD DIAGRAMAS DE ROBUSTEZ-CLASES

Esta matriz permite realizar la trazabilidad entre los diagramas de robustez definidos y las clases identificadas.

DIAGRAMA DE ROBUSTEZ	CLASES
DR-001	Proyecto Sesión Indicador Identificación

	Riesgo Categoría
DR-002	Proyecto Sesión Indicador Identificación Riesgo
DR-003	Proyecto Sesión Indicador Identificación Riesgo
DR-004	Proyecto Sesión Indicador Identificación
DR-005	Proyecto Sesión Indicador Identificación
DR-006	Riesgo Categoría
DR-007	Riesgo Categoría

Tabla N° 2.8: Matriz de trazabilidad, diagramas de robustez-clases (Maniasi, 2005, pág. 362).

CLASES	DIAGRAMA DE ROBUSTEZ
Proyecto	DR-001 DR-002 DR-003 DR-004
Sesión	DR-001 DR-002 DR-003 DR-004
Identificación	DR-001 DR-002 DR-003 DR-004
Categoría	DR-001 DR-006 DR-007
Riesgo	DR-001 DR-002 DR-003 DR-006 DR-007
Indicador	DR-001 DR-002

	DR-003 DR-004

Tabla N° 2.9: Matriz de trazabilidad, clases-diagramas de robustez (Maniasi, 2005, pág. 363).

E. MATRIZ DE TRAZABILIDAD DE REQUERIMIENTOS-CASOS DE PRUEBA

Esta matriz permite realizar la trazabilidad entre los requerimientos definidos y los casos de prueba especificados.

REQUERIMIENTO	CASO DE PRUEBA
RF-001	CP-007 CP-008 CP-009 CP-010 CP-011 CP-012 CP-013 CP-014 CP-027
RF-002	CP-015
RF-003	CP-024
RF-004	CP-016 CP-017 CP-023
RF-005	CP-018 CP-021
RF-006	CP-006
RF-007	CP-006
RNF-001	CP-001 CP-002 CP-003 CP-004 CP-005 CP-019 CP-020 CP-023 CP-026 CP-029
RNF-002	CP-005
RNF-003	CP-029
RNF-004	CP-028
RNF-005	No aplicable
RNF-006	No aplicable

Tabla N° 2.10: Matriz de trazabilidad de requerimientos-casos de prueba (Maniasi, 2005, pág. 363).

En modo hacia atrás se puede realizar la trazabilidad entre los casos de prueba especificados y los requerimientos definidos.

CASO DE PRUEBA	REQUERIMIENTO
CP-001	RNF-001
CP-002	RNF-001
CP-003	RNF-001
CP-004	RNF-001
CP-005	RNF-001 RNF-002
CP-006	RF-006 RF-007
CP-007	RF-001
CP-008	RF-001
CP-009	RF-001
CP-010	RF-001
CP-011	RF-001
CP-012	RF-001
CP-013	RF-001
CP-014	RF-001
CP-015	RF-002
CP-016	RF-004
CP-017	RF-004
CP-018	RF-005
CP-019	RNF-001
CP-020	RNF-001
CP-021	RF-005
CP-022	RNF-001
CP-023	RF-004 RNF-001

Tabla N° 2.11: Matriz de trazabilidad de casos de prueba requerimientos (Maniasi, 2005, pág. 363).

F. MATRIZ HACIA ATRÁS/HACIA DELANTE

"La matriz de trazabilidad hacia atrás/ hacia adelante es la habilidad para describir y seguir la vida de un requisito en ambas direcciones hacia delante (forward) y hacia atrás (backward)" (INTECO, 2008, pág. 16).

Esta trazabilidad se realiza desde su origen, a través de su desarrollo y especificación, hasta su construcción y uso, y a través de todos los períodos de

refinamiento en curso e iteración en alguna de estas fases.

Matriz de trazabilidad de Requisitos										
Requisitos				Diseño alto nivel	Diseño detallado	Código	ID Caso prueba unitario	ID Caso prueba integración	ID Caso prueba sistema	Petición de cambio
Req negocio	Req usuario	Req. Sistema /SW	Caso de uso							

Tabla Nº 2.12: Matriz hacia atrás/hacia delante (INTECO, 2008, pág.16).

G. MATRIZ DE DEPENDENCIAS

En la matriz de dependencias se irán registrando los requisitos de negocio y por cada requisito de negocio se identificarán los requisitos de usuario correspondientes.

		Requisitos (A)							
		Req 1	Req2	Req3	Req4	Req5	Req6	Req7	Req8
Requisitos (B)	Req 1		X			X			X
	Req2							X	
	Req3					X	X		
	Req4								X
	Req5	X							
	Req6		X						
	Req 7					X			
	Req8		X						

Tabla Nº 2.13: Matriz de dependencias (INTECO, 2008, pág. 17).

H. MATRIZ CRUD DE OBJETOS Y CASOS DE USO

La matriz CRUD(Create, Retrieve, Update y Delete) permite analizar las operaciones que deben realizarse sobre la base de datos a partir de la correlación entre tablas y funciones y entre otros elementos de la base de datos.

"A partir de ellas es posible analizar o inferir información acerca del nivel de especificación de los requisitos, el nivel de participación de los usuarios, el costo asociado a cada fase de desarrollo, la arquitectura requerida, el plan de las pruebas, etc." (Tabares, 2009, pág. 41). La construcción de estas matrices trae beneficios más allá de un simple registro de la correlación o dependencia entre los elementos de los modelos.

Crear objeto	**C**
Consultar objeto	**R**
Modificar objeto	**U**
Eliminar objeto	**D**

Tabla N° 2.14: Operaciones de la matriz CRUD.

	1	2	3	4	5	6	7	8	9	10
Objeto de Negocio	CU Realizar Presupuesto	CU Entrar Item Presupuesto	CU Consultar Presupuesto	CU Seleccionar Presupuesto	CU Consultar Item de Catálogo	CU Seleccionar Item de Catálogo	CU Consultar Detalles Item de Catálogo	CU Validar Habilitación Cliente	CU Autorizar Presupuesto	CU Calcular Importe Presupuesto
Presupuesto										
ItemDePresupuesto										
ModeloDeVehiculo										
Servicio										
Catalogo										
ItemDeServicio										
Reserva										
Contrato										
UsuarioAnónimo										

Tabla N° 2.15: Matriz CRUD de objetos y casos de uso (Aprende UML, 2012).

2.2.5. INGENIERÍA DE SOFTWARE

"La ingeniería del software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software" (Sommervile, 2005, p. 4). Por su parte Pressman(2005) sustenta que la ingeniería de software es una tecnología estratificada en herramientas, métodos, proceso y un enfoque de calidad, cualquier enfoque de la ingeniería de software debe estar sustentado

en un compromiso con la calidad.

La ingeniería del software ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo.

PROCESO DEL SOFTWARE

"Un proceso del software es un conjunto de actividades que conducen a la creación de un producto software" (Pressman, 2005, pág. 60).

Los procesos del software son complejos y, como todos los procesos intelectuales y creativos, dependen de las personas que toman decisiones y juicios. Debido a la necesidad de juzgar y crear, los intentos para automatizar estos procesos han tenido un éxito limitado.

Los modelos de desarrollo del software han evolucionado desde los modelos tradicionales hasta tener en la actualidad los métodos agiles de desarrollo.

Métodos ágiles	Métodos tradicionales
Se basan en heurísticas para la producción de código	Se basan en normas
Aceptan e incluso fomentan el cambio	Mayor o menor resistencia a los cambios
El cliente forma parte del equipo	El cliente no forma parte del equipo, sólo mantiene reuniones con éste
Equipos de trabajo pequeños o medianos	Equipos grandes (>15-20 miembros)
Procesos con pocas reglas	Procesos con muchas normas
Poca documentación	Mucha documentación
Poco análisis y diseño	Mucho análisis y diseño
Conceden poca importancia a la arquitectura de los sistemas	Conceden mucha importancia a la arquitectura de los sistemas

Tabla N° 2.16: Algunas diferencias significativas entre los dos tipos de métodos (Abián, 2006, pág. 18).

2.2.6. PROCESO ICONIX

Es un proceso simplificado en comparación con otros procesos más

tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto.

Presenta claramente las actividades de cada etapa y exhibe una secuencia de pasos que deben ser seguidos. Está entre la complejidad del RUP (Rational Unified Processes) y la simplicidad de XP (Extreme Programming).

"ICONIX establece un procedimiento sencillo y un conjunto mínimo de pasos que suelen dar lugar a muy buenos resultados. Estos resultados han demostrado ser coherentes y repetibles en los últimos 12 años" (Rosenberg & Stephens, 2007, pág. 2).

A. CARACTERÍSTICAS DE ICONIX

a. Iterativo e Incremental.- varias iteraciones ocurren entre el desarrollo del modelo del dominio y la identificación de los casos de uso. El modelo estático es incrementalmente refinado por los modelos dinámicos.

b. Trazabilidad.- cada paso está referenciado por algún requisito. Se define trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos de software.

c. Dinámica del UML.- La metodología ofrece un uso dinámico del UML por que utiliza algunos diagramas del UML, sin exigir la utilización de todos, como en el caso de RUP.

B. FASES DE ICONIX

FASE 1: ANÁLISIS DE REQUISITOS.

a. Requisitos funcionales: Definir lo que el sistema debe ser capaz de hacer. Dependiendo de la forma en que su proyecto está organizado, ya sea que usted participe en la creación de los requisitos funcionales o que los requisitos serán "dictados desde lo alto" de un cliente o un equipo de analistas de negocios.

b. Modelo de Dominio: Entender el espacio del problema en términos inequívocos (sin ambigüedad).

c. Requisitos de comportamiento: Define la forma en que el usuario y el sistema interactúan (es decir, escribir el primer proyecto de casos de uso). Le recomendamos que empezar con un prototipo GUI (lo que llamamos "guión" GUI) e identificar todos los casos de uso que vamos a poner en práctica, o, al menos, llegar a una primera lista de casos de uso, que espera razonablemente cambiar a medida que se explora los requisitos con mayor profundidad.

Hito 1. Revisión de Requisitos: Asegúrese de que la descripción de los casos de uso coincidan con las expectativas de sus clientes. Tenga en cuenta que se deben revisar los casos de uso en pequeños lotes, justo antes de diseñarlos.

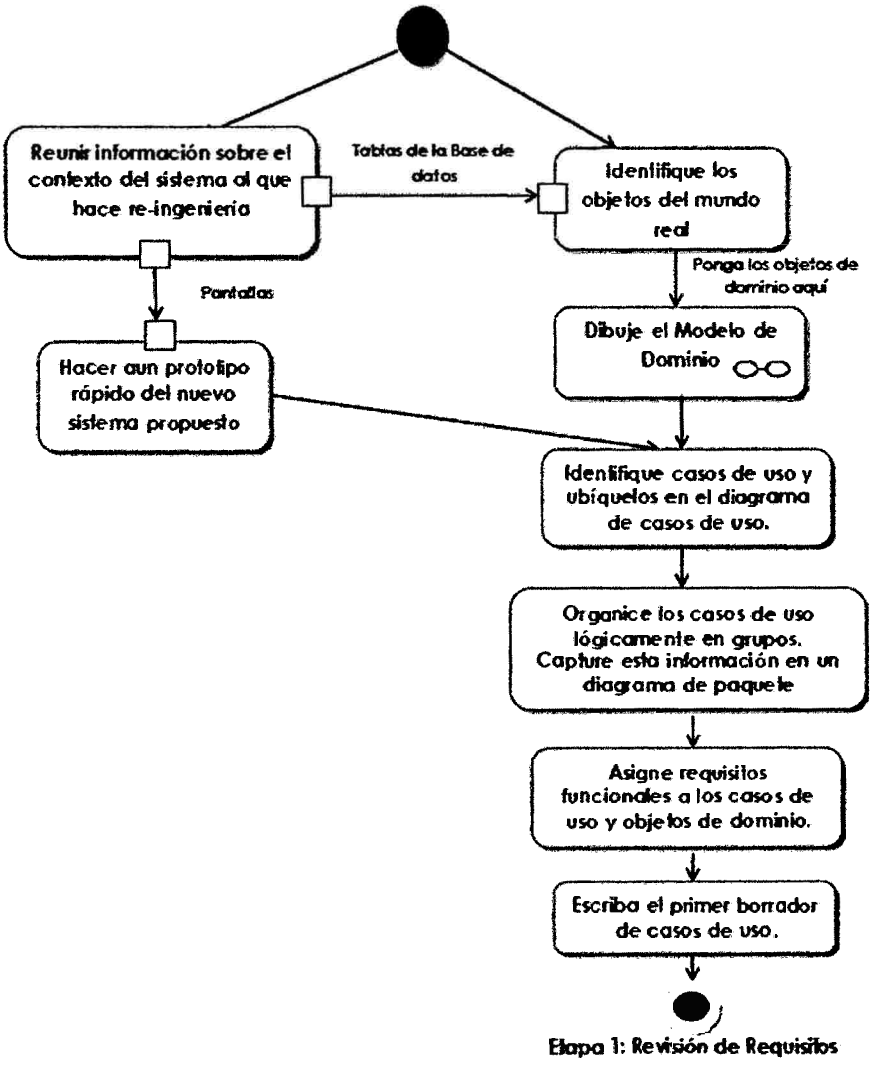


Figura N° 2.3: Análisis de Requisitos (Rosenberg & Stephens, 2007, pág. 5).

FASE 2. DISEÑO PRELIMINAR

a. Análisis de robustez: Dibuje un diagrama de robustez (una "imagen del

objeto" de los pasos en un caso de uso), reescribiendo los casos de uso a medida que avanza.

b. Actualice el modelo de dominio mientras escribe los casos de uso y dibuje el diagrama de robustez. Aquí descubrirá algunas clases "perdidas", corregirá las ambigüedades, y añadirá atributos a los objetos de dominio (por ejemplo: identificar que un libro tiene un Título, autor, sinopsis, etc.).

c. Nombre todas las funciones lógicas del software (controladores) necesitadas para hacer que los casos de uso funcionen.

d. Reescriba el primer borrador de casos de uso.

Hito 2: Revisión del Diseño Preliminar (RDP).

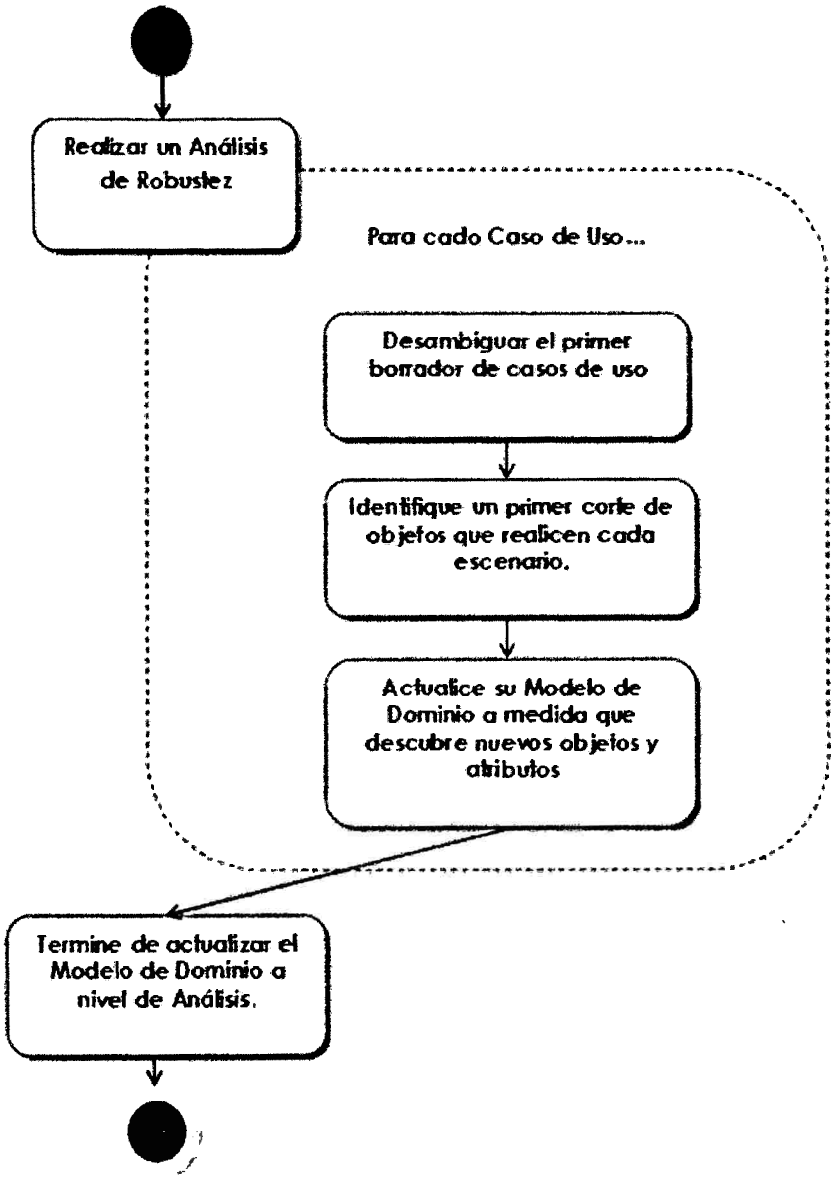


Figura N° 2.4: Análisis y diseño preliminar (Rosenberg & Stephens, 2007, pág. 10).

FASE 3. DISEÑO

a. Diagrama de Secuencia: Dibuje un diagrama de secuencia (un diagrama de secuencia por cada caso de uso) para mostrar en detalle cómo va a implementar el caso de uso caso. La función fundamental de los diagramas de secuencia es asignar comportamiento para sus clases.

b. Actualice el modelo de dominio mientras se dibujan los diagramas de secuencia, y añada operaciones a los objetos de dominio. En esta fase, los objetos de dominio son realmente entidades del modelo estático.

c. Depurar el modelo estático.

Hito 3: Revisión del diseño crítico (RDC).

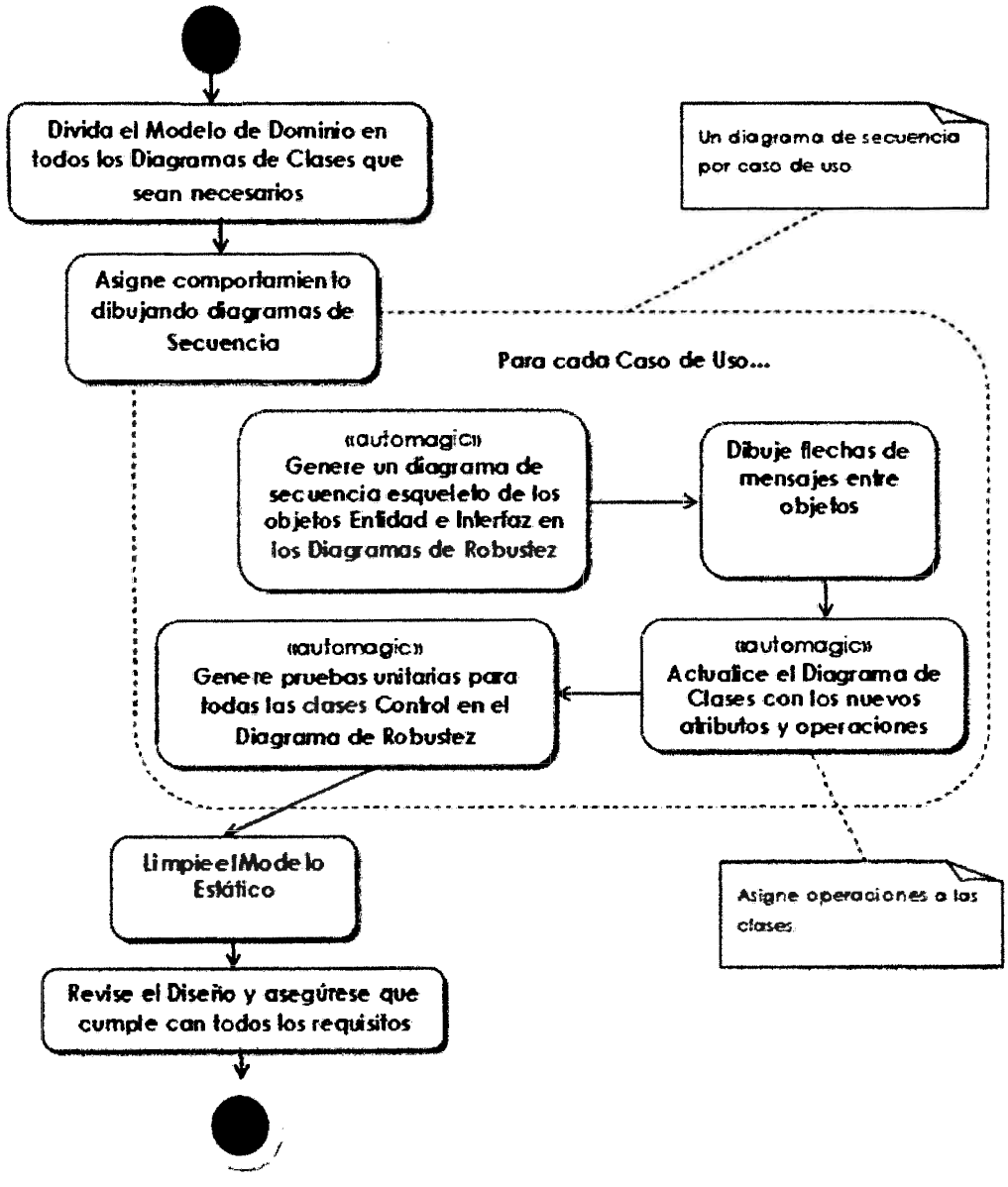


Figura N° 2.5: Diseño detallado (Rosenberg & Stephens, 2007, pág. 13).

FASE 4. IMPLEMENTACIÓN

a. Código/pruebas unitarias: Escriba el código y las pruebas unitarias. (o, dependencia de sus preferencias, escriba las pruebas unitarias y luego el código).

b. Integración y pruebas de escenario: Base las pruebas de integración en los casos de uso, de modo que así probara ambos cursos, el básico y el alterno.

c. Ejecute una revisión de código y actualice el modelo para prepararse para la próxima ronda de trabajo.

Hito 4: Entregable.

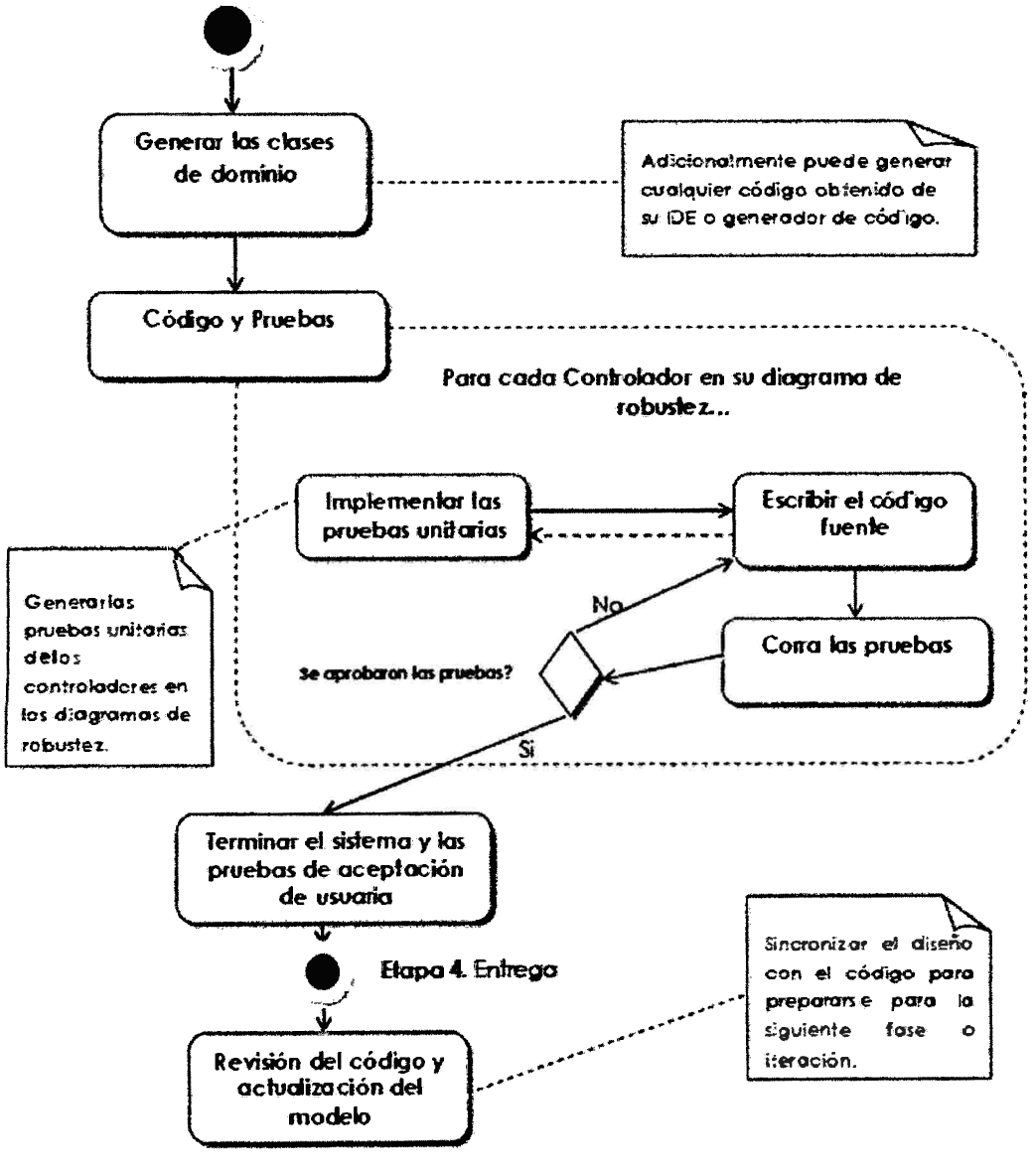


Figura N° 2.6: Implementación (Rosenberg & Stephens, 2007, pág. 16).

FASE 5. PRUEBAS

a. Prueba basada en el diseño: Aquí se aplican las diferentes tipos de pruebas como pruebas unitarias, prueba de integración, prueba de compatibilidad, pruebas de sistema, prueba de aceptación, prueba del Beta, prueba de versión, prueba de rendimiento, prueba de regresión, prueba de tensión y la prueba de cantidad o volumen.

b. Direccionamiento de requisitos.

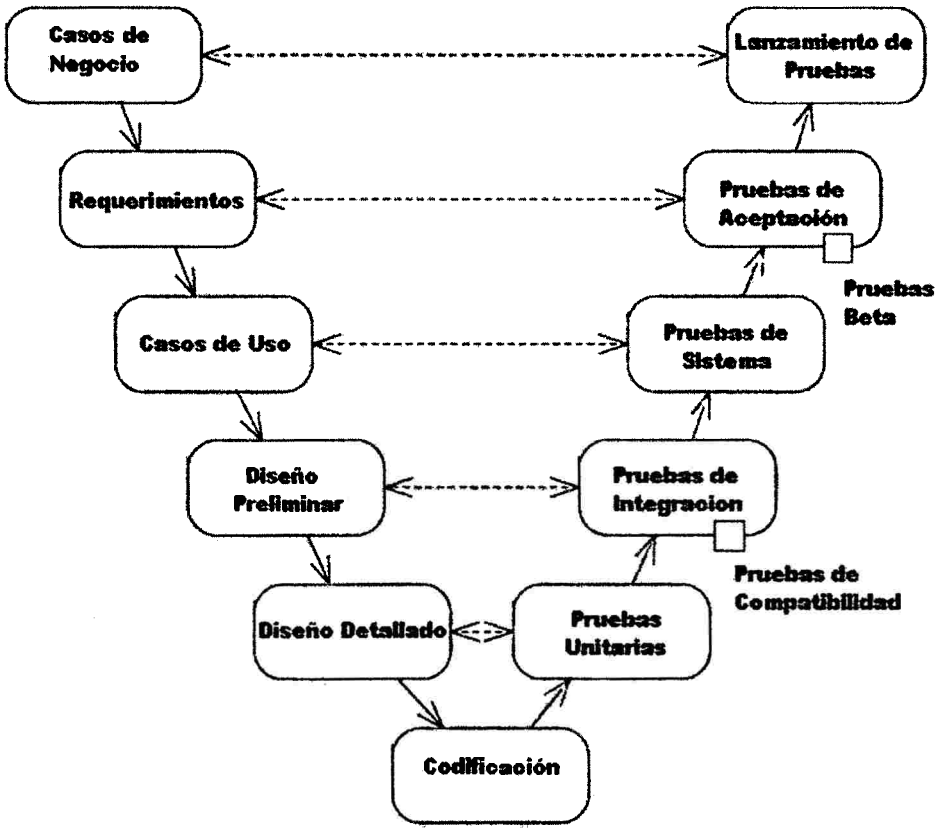


Figura N° 2.7: Modelo "V" de pruebas de software aplicado al proceso ICONIX (Rosenberg & Stephens, 2007, pág. 332).

2.2.7. LENGUAJE DE PROGRAMACIÓN

"Un lenguaje de programación puede definirse como una notación para escribir instrucciones u órdenes para el ordenador y necesarias para la de un determinado proceso" (Garcia & Roque, 2007, pág. 62).

"Los lenguajes de programación son como idiomas que constituyen el sistema de comunicación entre el hombre y el ordenador, mediante el cual se transmiten a éste las instrucciones e información en un formato comprensible para la maquina" (de Pablos, López, Romero, & Medina, 2004, pág. 110).

Los lenguajes de programación que se presentan como código máquina (ceros y unos) se denominan lenguajes de de "bajo nivel", mientras que los que se parecen a los usuarios (lenguaje natural o humano) se denominan de "alto nivel".

A. PARADIGMAS DE PROGRAMACIÓN

"Un paradigma de programaciones una colección de patrones conceptuales que moldean la forma de razonar sobre problemas, de formular soluciones y de estructura programas" (Rodríguez, Santamaría, Rabasa, & Martínez, 2003, pág. 4).

"Un paradigma de programación es un modelo básico de construcción de programas. Un modelo que permite producir programas conforme con unas directrices, tales como diseñar un programa mediante una secuencia de instrucciones que operan sobre unos datos de entrada y producen un resultado de salida" (Alonso, Martínez, & Segovia, 2005, pág. 3).

Por lo tanto un paradigma de programación es un modelo o patrón para la construcción de programas. Los paradigmas de programación que tienen mayor relevancia en la son:

a. Paradigma imperativo.
b. Paradigma funcional.
c. Paradigma lógica.
d. Paradigma orientado a objetos.

Pero en la actualidad existe paradigmas nuevos que empiezan a tener adeptos como:

a. Paradigma orientado a aspectos.
b. Paradigma orientado a eventos.
c. Paradigma orientado a agentes.

B. PARADIGMA DE PROGRAMACIÓN ORIENTADA A OBJETOS.

Según (Rodríguez, Santamaría, Rabasa, & Martínez, 2003) el paradigma orientado a objetos (OO) se refiere a un estilo de programación. Un lenguaje de programación orientado a objetos (LOO) puede ser tanto imperativo como

funcional o lógico. Lo que caracteriza un LOO es que se basa en clases, objetos y herencia.

"Si una aplicación no incluye abstracción, encapsulación, jerarquías, mensajes y polimorfismo no está orientada a objetos" (Abián, 2006, pág. 6).

El paradigma orientado a objetos es una filosofía de desarrollo y empaquetamiento de programas que permite crear unidades funcionales genéricas, de forma que el usuario las pueda utilizar según sus necesidades y de acuerdo con las especificaciones del problema a desarrollar.

C. LENGUAJES DE PROGRAMACIÓN ORIENTADA A OBJETOS

"El análisis y diseño OO puede aplicarse a cualquier sistema, independientemente de que se implemente al final en un lenguaje orientado a objetos o no. En cuanto a la POO, es posible incluso en lenguajes no orientado a objetos" (Abián, 2006, pág. 49).

El paradigma orientado a objetos no depende de ningún lenguaje; un lenguaje para ser OO debe tener como mínimo:

a. Encapsulación.
b. Herencia.
c. Polimorfismo.

Un LOO para ser considerada como un lenguaje OO puro debe tener además:

d. Los tipos de datos predefinidos en el lenguaje son todos objetos.
e. Los tipos de datos que puedan ser definidos por los programadores son todos objetos.
f. Todas las operaciones se realicen enviando mensajes entre objetos.

Característica	Fortran 77	C++	C#	Java	Smalltalk	Eiffel
a)	NO	SI	SI	SI	SI	SI
b)	NO	SI	SI	SI	SI	SI
c)	NO	SI	SI	SI	SI	SI
d)	NO	NO	NO	NO	SI	SI
e)	NO	NO	NO	SI	SI	SI
f)	NO	NO	NO	NO	SI	SI

Tabla N° 2.17: Ejemplos de lenguajes no OO, OO "híbridos" y OO "puros" (Abián, 2006, pág. 56).

D. EL LENGUAJE JAVA

Java es un lenguaje de programación orientada a objetos creado por Sun Microsystems, Inc. que permite crear programas que funcionan en cualquier tipo de ordenador y sistema operativo.

"La clave consistió en desarrollar un código "neutro" el cual estuviera preparado para ser ejecutado sobre una "máquina hipotética o virtual", denominada Java Virtual Machine (JVM)" (García, Rodríguez, 2006, pág. 23). "JDK (Java Development Kit) es el Kit de desarrollo de Java. Se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java" (Abián, 2006, pág. 28).

Las aplicaciones Java se compilan en un formato llamado código de byte o bytecode (con extensión .class), El código compilado de Java puede correr en casi cualquier computadora porque los intérpretes de Java y los ambientes de ejecución (Java Virtual Machine o JVM).

E. ARQUITECTURA MVC (Model-View-Controller)

"La arquitectura MVC propuesta por SUN es una solución de desarrollo web en el lado servidor que permite separar la parte lógica de la parte de presentación de una aplicación web" (Aumaille, 2002, pág. 21).

En la arquitectura MVC:

a. El modelo está representado por los EJBs y/o los JavaBeans.
b. La vista está representada por los JSPs.
c. El controlador está representado por los Servlets.

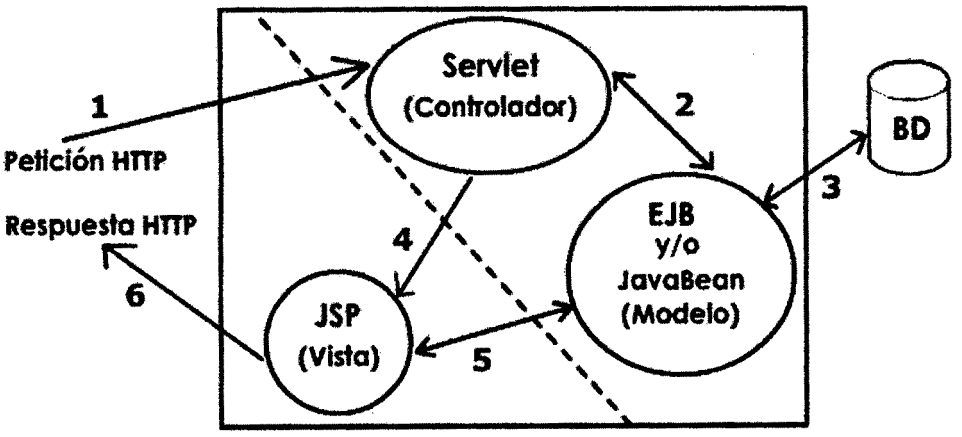


Figura N° 2.8: Arquitectura MVC (Aumaille, 2002, pág. 22).

2.2.8. BASE DE DATOS

"Una base de datos de un SI es la representación integrada de los conjuntos de entidades instancia correspondientes a las diferentes entidades tipo del SI y de sus interrelaciones" (Camps, Casillas, Costal, Gibert, Martín, & Pérez, 2005, pág. 8).

Esta representación informática (o conjunto estructurado de datos) debe *poder* ser utilizada de forma compartida por muchos usuarios de distintos tipos.

Una base de datos es un conjunto estructurado de datos que representa entidades y sus interrelaciones. La representación será única e integrada, a pesar de que debe permitir utilizaciones varias y simultáneas.

A. SISTEMA GESTOR DE BASE DE DATOS(SGBD)

"La aparición de los SGBD relacionales supone un avance importante para facilitar la programación de aplicaciones con BD y para conseguir que los programas sean independientes de los aspectos físicos de la BD" (Camps, Casillas, Costal, Gibert, Martín, & Pérez, 2005, pág. 11).

Los SGBD relacionales están en plena transformación para adaptarse a tres tecnologías de éxito reciente, fuertemente relacionadas: la multimedia, la de orientación a objetos (OO) e Internet y la web.

Los usuarios podrán hacer consultas de cualquier tipo y complejidad directamente al SGBD. El SGBD tendrá que responder inmediatamente sin que estas consultas estén preestablecidas; es decir, sin que se tenga que escribir, compilar y ejecutar un programa específico para cada consulta.

B. POSTGRESQL

"Es un gestor de bases de datos orientadas a objetos (SGBDOO o ORDBMS en sus siglas en inglés) muy conocido y usado en entornos de software libre porque cumple los estándares SQL92 y SQL99" (Camps, Casillas, Costal, Gibert, Martín, & Pérez, 2005, pág. 382).

PostgreSQL implementa las características necesarias para competir con cualquier otra base de datos comercial, con la ventaja de tener una licencia de libre distribución BSD.

La migración de bases de datos alojadas en productos comerciales a PostgreSQL se facilita gracias a que soporta ampliamente el estándar SQL. Por si esto fuera poco PostgreSQL es extensible. Es posible agregar nuevos tipos de datos y funciones al servidor que se comporten como los ya incorporados.

CAPÍTULO III
METODOLOGÍA DE LA INVESTIGACIÓN

3.1. TIPO DE INVESTIGACIÓN

Se desarrollará un "software de trazabilidad de requisitos para el proceso ICONIX", usando como metodología el proceso ICONIX. "La investigación descriptiva es uno de los tipos o procedimientos investigativos más populares y utilizados por los principiantes en la actividad investigativa. Los trabajos de grado, en los pregrados y en muchas de las maestrías, son estudios de carácter eminentemente descriptivo. En tales estudios se muestran narran reseñan o identifican hechos, situaciones, rasgos, características de un objeto de estudio, o se diseñan productos, modelos, prototipos, guías, etc. Pero no se dan explicaciones o razones del porqué de las situaciones, los hechos, los fenómenos etc." (Bernal, 2006, pág 112).

En este estudio se selecciona los conocimientos sobre la trazabilidad de requisitos y el proceso ICONIX; se describe cada una de las variables independientemente; para luego modelar y realizar un prototipo, por estas consideraciones el tipo de investigación es descriptivo.

3.2. DISEÑO DE LA INVESTIGACIÓN

En el diseño de esta investigación se consideran que los "diseños de investigación no experimentales; son aquellos cuyas variables independientes carecen de manipulación intencional, y no poseen grupo de control, ni mucho menos experimental. Analizan y estudian los hechos y fenómenos de la realidad después de su ocurrencia" (Carrasco, 2006, pág.75). Por dicho el diseño de esta investigación es no experimental.

"Los diseños de investigacion transeccional o transversal recocolectan datos en un solo momento, en un tiempo único. Su propósito es describir variables, y analizar su incidencia e interrelación en un momento dado." (Hernández, Fernández, & Baptista, 2006, pág. 120). Por lo tanto, este estudio es

transeccional descriptivo.

En esta investigación, se usará el proceso ICONIX, para el desarrollo del producto software. Los instrumentos de recolección de datos se presentan en los anexos. Por las consideraciones anteriores, el diseño de la investigación es no experimental y transeccional descriptivo.

3.3. POBLACIÓN Y MUESTRA

POBLACIÓN.- La población está constituida por todas las tareas del proceso de desarrollo de software con ICONIX, 2012.

MUESTRA.- La muestra son todas las tareas del proceso de desarrollo de software con ICONIX, 2012.

3.4. DEFINICIÓN DE VARIABLES E INDICADORES

3.4.1. DEFINICIÓN CONCEPTUAL DE VARIABLES

A. VARIABLE INDEPENDIENTE

Trazabilidad de Requisitos.- La trazabilidad de requisitos es una práctica de la ingeniería de requisitos que facilita la gestión de la transformación y la evolución de los requisitos para mantener su consistencia y completitud por medio de vínculos de trazado entre diferentes artefactos que se representan a través del ciclo de desarrollo de software.

B. INDICADORES DE LA VARIABLE INDEPENDIENTE

Completitud de requisitos.- La completitud de requisitos significa que un requisito de entrada en el modelo del sistema debe estar presente en la solución del sistema, es decir la especificación de un requisito es completa al grado que todas sus partes están presentes y cada parte es totalmente desarrollada. Una especificación de software debe exponer varias propiedades para asegurar su completitud.

Consistencia de requisitos.- Un requisitos es consecuente, cuando no hay contradicciones y redundancias entre los requisitos, un requisito entra en conflicto con otro o con especificaciones gobernantes y objetivos cuando no están relacionados adecuadamente en el modelo.

C. VARIABLE DEPENDIENTE

Proceso ICONIX.- Es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto.

D. INDICADORES DE LA VARIABLE DEPENDIENTE

Análisis.- Es la etapa de reunión de requisitos, se intensifica y se centra especialmente en el software. El proceso de análisis es fundamental porque través de este se capturan los requisitos funcionales y no funcionales, y se describe el comportamiento del software.

Diseño preliminar.- Es la etapa preliminar al diseño donde se identifican los controladores y se dibuja el diagrama de robustez y se actualizan los artefactos de la etapa de análisis.

Diseño.- La etapa del diseño se refiere al establecimiento de las estructuras de datos, la arquitectura general del software, se representan los diagramas de secuencia y se actualiza el modelo de dominio.

Implementación.- Esta actividad consiste en traducir el diseño en una forma legible para la máquina, es decir, se genera código. La generación de código se refiere tanto a la parte de código fuente como a las pruebas unitarias, a la integración y las pruebas de escenario.

Pruebas.- Una vez que se ha generado código, comienzan las pruebas del software que se ha desarrollado. La etapa de pruebas se centra en los procesos lógicos internos del software, y en los procesos externos funcionales, es decir, la realización de las prueba para la detección de errores.

3.4.2. DEFINICIÓN OPERACIONAL DE VARIABLES E INDICADORES

VARIABLE INDEPENDIENTE

X. Trazabilidad de requisitos

INDICADORES DE LA VARIABLE INDEPENDIENTE

X1. Completitud de requisitos

X2. Consistencia de requisitos

VARIABLE DEPENDIENTE

Y. Proceso ICONIX

INDICADORES DE LA VARIABLE DEPENDIENTE

Y1. Análisis

Y2. Diseño preliminar

Y3. Diseño

Y4. Implementación

Y5. Pruebas

3.5. TÉCNICAS E INSTRUMENTOS PARA LA RECOLECCIÓN DE DATOS

3.5.1. TÉCNICAS

Las técnicas que se utilizaran para la investigación es el análisis documental de libros digitales, libros físicos, revistas, tesis, informes de investigación, audios y videos.

3.5.2. INSTRUMENTOS PARA LA RECOLECCIÓN DE DATOS

En el análisis de datos se utilizara tablas para la síntesis y evaluación de los datos existentes sobre las variables de investigación, con la finalidad de seleccionar los factores deseables para la investigación.

FICHA DE TRABAJO

La ficha de trabajo es la unidad de registro en investigación que consigna datos obtenidos del contenido de un documento. Estas fichas se subdividen en conceptuales, textuales, sinópticas, personales y mixtas.

Ficha N° ____

Autor: **Tópico:** _____

Título: _____

Capítulo: **Página:**

_____

_____

_____

_____

Figura N° 3.1: Ficha de trabajo.

En esta ficha de análisis documental, se identificarán las tareas para cada fase del proceso ICONIX y los artefactos que estas tareas generan; se determina si los artefactos son trazables y el nivel de trazabilidad, mediante los conceptos extraídos con la ficha de trabajo, la información recolectada se muestra en el Anexo B.

FASE	TAREA	ARTEFACTO	ARTEFACTO TRAZABLE(SI/NO)	NIVEL DE TRAZABILIDAD

Tabla N° 3.1: Ficha de análisis documental de la trazabilidad para el proceso ICONIX.

En esta ficha de análisis documental identificarán las reglas de consistencia para los artefactos trazables, mediante los conceptos extraídos de las matrices de trazabilidad, la información recolectada se muestra en el Anexo C.

ARTEFACTO	RELACIÓN (trace/realize)	REGLA DE CONSISTENCIA

Tabla N° 3.2: Ficha de análisis documental de la consistencia del trazado.

En esta ficha de análisis, se identificarán cómo evolucionan los artefactos del proceso ICONIX, determinando en qué fase se crean, se modifican o se revisan dichos artefactos, la información recolectada se muestra en el Anexo A.

ARTEFACTO	FASES DE ICONIX (Crea / Modifica / Revisa)				
	Análisis	Diseño preliminar	Diseño	Implementación	Pruebas

Tabla N° 3.3: Ficha de análisis documental de la evolución de los artefactos del proceso ICONIX.

3.6. TÉCNICAS PARA PROCESAMIENTO DE DATOS E INFORMACIÓN

3.6.1. PROCESO DE LA INGENIERÍA DE REQUISITOS

El proceso de la ingeniería de requisitos propuesto por Ian SOMMERVILLE (Capitulo 2, titulo tareas de la ingeniería de requisitos), es el más adecuado para la metodología ICONIX por ser el más sencillo y preciso para la metodología.

ACTIVIDADES	TAREAS	ARTEFACTOS	TÉCNICA
Obtención y análisis de requisitos	1. Descubrimiento de requerimientos. 2. Clasificación y organización de requerimientos. 3. Ordenación por prioridades. 4. Documentación de requerimientos.	⌂ Modelos del sistema	⌂ Entrevista
Especificación de requisitos	1. Especificación de los requisitos del negocio. 2. Especificación de los requisitos del usuario. 3. Especificación y modelado de los requisitos del sistema.	⌂ Requisitos del usuario del sistema	⌂ Entrevista
Validación de requisitos	1. Verificaciones de validez. 2. Verificaciones de consistencia. 3. Verificaciones de completitud. 4. Verificaciones de realismo. 5. Verificabilidad.	⌂ Informe de viabilidad. ⌂ Documento de requisitos	⌂ Construcción de prototipos. ⌂ Revisiones. ⌂ Generación de casos de prueba.

Tabla N° 3.4: Proceso de la ingeniería de requisitos (Sommerville, 2005).

3.6.2. EL PROCESO ICONIX

A. ANÁLISIS

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Identificar requisitos	⌂ Requisitos funcionales y no funcionales ⌂ Casos de prueba	⌂ Entrevistas ⌂ Definir lo que el sistema debe hacer ⌂ Escribir un caso de prueba para cada requisito	Usuario Cliente Analista

2. Identificar objetos del mundo real y dibujar el modelo de dominio	⌂ Glosario de objetos ⌂ Modelo de dominio	⌂ Identificar clases clave del dominio ⌂ Identificar sustantivos y depurar ⌂ Identificar objetos en requisitos funcionales ⌂ Utilizar agregación y generalización	Analista
3. Realizar prototipo de interfaz grafica	⌂ Prototipo GUI	⌂ Utilizar historia de eventos de usuario ⌂ Utilizar los requisitos funcionales ⌂ Diseñar interfaz gráfica básica	Analista Programador
4. Descubrir casos de uso	⌂ Lista de casos de uso	⌂ Utilizar requisitos funcionales ⌂ Entrevista	Usuario Cliente Analista
5. Dibujar y empaquetar los casos de uso	⌂ Diagrama de casos de uso ⌂ Paquete de casos de uso	⌂ Identificar roles y responsabilidades de los actores ⌂ Asociar actores con los casos de uso ⌂ Relacionar los casos de uso ⌂ Agrupar lógicamente los casos de uso	Analista
6. Asigne requisitos funcionales a los casos de uso	⌂ Relación entre requisitos funcionales y casos de uso	⌂ Asignar requisitos funcionales a los casos de uso	Analista
7. Escribir el primer borrador de casos de uso	⌂ Primer borrador de casos de uso	⌂ Utilizar glosario de objetos del modelo de dominio ⌂ Utilizar la regla de dos párrafos ⌂ Escribir el caso de uso como flujo de evento/respuesta ⌂ Escribir el caso de uso con estructura sustantivo-verbo-sustantivo ⌂ Escribir caso de uso en voz activa ⌂ Referenciar pro su nombre las interfaces	Analista

Tabla N° 3.5: Análisis de requisitos (Porras, 2011, pág. 22).

REVISIÓN DE REQUISITOS

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Revisar el modelo de dominio	⌂ Modelo de dominio	⌂ Identificar al menos el 80% de las clase clave del dominio del problema	Usuario Cliente Analista Programador
2. Revisar el prototipo GUI	⌂ Prototipo GUI	⌂ Diseñar con precisión la GUI relacionada al caso de uso	Usuario Cliente Analista Programador
3. Revisar modelo de casos de uso	⌂ Casos de uso revisado	⌂ Eliminar clases fuera del dominio del problema ⌂ Cambiar descripción de voz pasiva a activa ⌂ Describir todos los cursos alternos ⌂ Asociar todos los requisitos de los casos de uso ⌂ Describir que intenta hacer el usuario para cada caso de uso	Usuario Cliente Analista Programador

Tabla N° 3.6: Revisión de requisitos (Primer hito) (Porras, 2011, pág. 24).

B. DISEÑO PRELIMINAR

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Reescriba el primer borrador para cada caso de uso	⌂ Caso de uso desambiguado	⌂ Reescribir el caso de uso durante el análisis de robustez	Analista
2. Identificar el primer corte de objetos que contempla n escenarios para cada caso de uso	⌂ Diagrama de robustez	⌂ Copiar la descripción del caso de uso en el diagrama de robustez ⌂ Usar clases del modelo de dominio ⌂ Crear un objeto interfaz pro cada GUI y nombrarlo ⌂ Transformar los verbos del caso de uso en controladores ⌂ Relacionar un caso de uso al diagrama de robustez cuando es invocado ⌂ Utilizar las reglas para	Analista

		construir el diagrama de robustez	
3. Actualizar el modelo de dominio	⌂ Modelo de dominio actualizado	⌂ Actualizar el modelo de dominio con nuevas clases y atributos durante el análisis de robustez	Analista
4. Actualizar el diagrama de clases de análisis	⌂ Modelo de dominio actualizado	⌂ Actualizar el diagrama de clases de análisis al finalizar el análisis de robustez ⌂ Asignar atributos a las clases entidad	Analista

Tabla N° 3.7: Diseño preliminar (Porras, 2011, pág. 31).

REVISIÓN DE DISEÑO PRELIMINAR

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Revisar descripción de caso de uso	⌂ Caso de uso	⌂ Coincidir la descripción del caso de uso con el diagrama de robustez	Usuario Cliente Analista Programador
2. Revisar diagrama de robustez	⌂ Diagrama de robustez	⌂ Coincidir el diagrama de robustez con descripción del caso de uso ⌂ Verificar que el diagrama cumple las reglas ⌂ Verificar que el diagrama de robustez tiene todos los cursos alternos	Usuario Cliente Analista Programador
3. Revisar modelo de dominio actualizado	⌂ Modelo de dominio actualizado	⌂ Coincidir los objetos entidad del diagrama de robustez con el modelo de dominio actualizado	Usuario Cliente Analista Programador

Tabla N° 3.8: Revisión de diseño preliminar (Segundo hito) (Porras, 2011, pág. 35).

C. DISEÑO

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Dividir el modelo de dominio actualizado para cada	⌂ Parte el modelo de dominio actualizado	⌂ Coincidir las clases entidad del diagrama de robustez con parte del modelo de dominio actualizado y	Diseñador

caso de uso		dibujando	
2. Dibujar un diagrama de secuencia para cada caso de uso	⌂ Diagrama de secuencia	⌂ Copiar la descripción del caso de uso ⌂ Copiar objetos entidad, interfaz y actores del diagrama de robustez ⌂ Verificar que un mensaje del diagrama de secuencia es verbo en el caso de uso ⌂ Hacer refactoring al diagrama de secuencia antes de codificar	Diseñador Programador
3. Actualizar el diagrama de clases de un caso de uso	⌂ Diagrama de clases	⌂ Asignar operaciones a las clases a partir de mensajes del diagrama de secuencia ⌂ Establecer multiplicidad en las clases ⌂ Depurar las clases, operaciones y atributos del diagrama de clases	Diseñador Programador
4. Extraer controlador es para pruebas unitarias	⌂ Lista de controlador es	⌂ Identificar controladores la lógica del negocio desde un diagrama de robustez	Diseñador Programador

Tabla N° 3.9: Diseño (Porras, 2011, pág. 50).

REVISIÓN CRÍTICA DE DISEÑO

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Revisar el diagrama de secuencia	⌂ Diagrama de secuencia	⌂ Verificar que el diagrama de secuencia coincide con la descripción del caso de uso ⌂ Verificar que el diagrama de secuencia representa los cursos básico y alterno ⌂ Verificar en los mensajes que los	Diseñador Programador

		atributos y valores de retorno son correctos	
2. Revisar el diagrama de clases	⌂ Diagrama de clases	⌂ Verificar que el nombre, atributos y operaciones se asignaron correctamente a las clases ⌂ Asignar requisitos no funcionales a los casos de uso y clases	Diseñador Programador
3. Revisar modelo de dominio actualizado	⌂ Modelo de dominio actualizado	⌂ Verificar nombres y atributos del modelo de dominio actualizado	Diseñador Programador
4. Revisar lista de pruebas unitarias	⌂ Lista de controladores	⌂ Actualizar la lista de controladores	Diseñador Programador

Tabla N° 3.10: Revisión crítica de diseño (Tercer hito) (Porras, 2011, pág. 58).

D. IMPLEMENTACIÓN

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Implementar la base de datos física	⌂ Base de datos física	⌂ Escribir el script usando el modelo de dominio actualizado ⌂ Ejecutar el script usando un DBMS	Programador
2. Implementar código para clases entidad	⌂ Código fuente	⌂ Escribir o generar código fuente con una herramienta usando el modelo de dominio actualizado	Programador
3. Implementar código para las interfaces	⌂ Código fuente	⌂ Generar código fuente usando una herramienta	Programador
4. Crear pruebas unitarias para cada controlador	⌂ Prueba unitaria	⌂ Escribir código fuente para una prueba unitaria usando una herramienta	Programador
5. Implementar código fuente para cada controlador	⌂ Código fuente	⌂ Escribir el código fuente siguiendo el flujo normal del diagrama de secuencia usando una herramienta	Programador

		⌂ Actualizar el diagrama de secuencia con la codificación	
6. Ejecutar pruebas unitarias para cada controlador	⌂ Reporte de pruebas unitarias	⌂ Ejecutar el módulo de cada prueba unitaria ⌂ Modificar código fuente si la prueba unitaria muestra resultado incorrecto	Programador
7. Ejecutar pruebas de aceptación para cada caso de uso	⌂ Reporte de pruebas de aceptación	⌂ Utilizar los casos de prueba de aceptación ⌂ Ejecutar el módulo de un caso de uso ⌂ Modificar código fuente si la prueba de aceptación muestra resultado incorrecto	Programador Usuario Cliente

Tabla N° 3.11: Implementación (Porras, 2010).

REVISIÓN DEL CÓDIGO Y ACTUALIZACIÓN DEL MODELO

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
1. Revisar el código	⌂ Código fuente	⌂ Realizar una revisión del código fuente	Programador
2. Actualizar el modelo de dominio	⌂ Modelo de dominio actualizado	⌂ Actualizar el modelo de dominio	Diseñador Programador

Tabla N° 3.12: Entregable (Cuarto hito).

3.7. HERRAMIENTAS PARA PROCESAMIENTO DE DATOS E INFORMACIÓN

Software	Versión	Fabricante	Descripción
NetBeans	7.0	Oracle Corporation	IDE para desarrollar aplicaciones en JAVA.
Jaspersoft iReport Designer	3.7.6	Jaspersoft Corporation	Librerías de JAVA para el diseño e implementación de reportes impresos.
JUnit	4.0	Common Public License	Conjunto de librerías para realizar las pruebas unitarias en aplicaciones Java.
PostgreSQL	9.0.1	PostgreSQL Global Development	Motor de base de datos estable y robusto.
Enterprise Architec	8.0	Sparx Systems	Plataforma de diseño,

			visualización y modelado de alto rendimiento basada en el estándar de UML 2.3.
Macromedia Dreamweaver	8.0	Adobe	Programa para el diseño, programación y edición de sitios y aplicaciones Web basadas en el estándar HTML.
ER/Studio	8.0	Embarcadero Technologies	Herramienta que modela los datos, se usa para el diseño y la construcción lógica y física de base de datos.

Tabla N° 3.13: Herramientas para el procesamiento de la información.

CAPITULO IV
RESULTADOS DE LA INVESTIGACIÓN

4.1. RESULTADOS

4.1.1. TRAZABILIDAD DE REQUISITOS

A. COMPLETITUD DE REQUISITOS

Se plantea el siguiente modelo de trazabilidad para garantizar la completitud de los requisitos, a partir del análisis de la información obtenida de las fichas de análisis documental de la trazabilidad para el proceso ICONIX (tabla N° 3.1) cuyos datos recolectados se muestra en el Anexo B y la ficha de análisis de la evolución de los artefactos del proceso ICONIX (tabla N° 3.3) cuyos datos recolectados se muestra en el Anexo A.

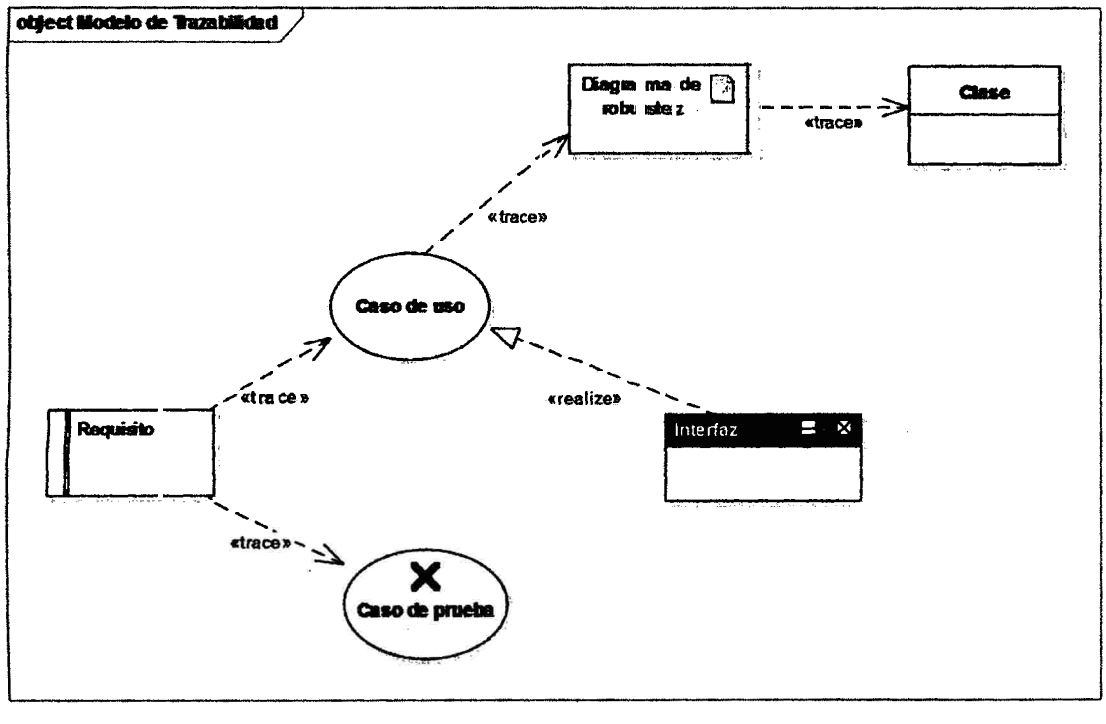


Figura N° 4.1: Modelo de trazabilidad de requisitos.

B. CONSISTENCIA DE REQUISITOS

Para verificar la consistencia de los requisitos se utilizará las matrices de trazabilidad planteados por Maniasi en el capítulo II, para las cuales se utilizan

las reglas de consistencia obtenidas de la ficha de análisis documental de la consistencia del trazado (tabla 3.2) que se detalla en el Anexo C.

ARTEFACTO ORIGEN	RELACIÓN	ARTEFACTO DESTINO	CONSISTENCIA
Requisitos	<<trace>>	Caso de uso	RF[1-1]CU RNF[1-0.n]CU
Caso de uso	<<realice>>	Requisitos	CU[1-1]RF CU[1-1]RF y RNF
Caso de uso	<<trace>>	Interfaz	CU[1-1.n]INT
Interfaz	<<realice>>	Caso de uso	INT[1-1.n]CU
Caso de uso	<<trace>>	Diagrama de robustez	CU[1-1]DR
Diagrama de robustez	<<realice>>	Caso de uso	DR[1-1]CU
Diagrama de robustez	<<trace>>	Clases	DR[1-1.n]CL
Clases	<<realice>>	Diagrama de robustez	CL[1-1.n]DR
Requisitos	<<trace>>	Caso de prueba	RF[1-1.n]CP RNF[1-0.n]CP
Caso de prueba	<<realice>>	Requisitos	CP [1-1.n]RF CP[1-1.n]RNF

Tabla N° 4.1: Reglas de consistencia del trazado.

4.1.2. ARTEFACTOS DEL SOFTWARE APLICANDO EL PROCESO ICONIX

Desarrollando el software con el proceso ICONIX, se generaron los siguientes artefactos para el software de trazabilidad de requisitos para el proceso ICONIX, Ayacucho, 2012. A continuación se detallarán las fases del proceso y los hitos que generan los artefactos del software.

A. ANÁLISIS DE REQUISITOS

Realizando el análisis de los datos recolectados en las fichas de análisis documental del Anexo A, B, C y de los resultados del modelo de trazabilidad de la figura N° 4.1 y de las reglas de consistencia de la tabla N° 4.1, se plantean los siguientes requisitos funcionales y requisitos no funcionales para el desarrollo del software de trazabilidad de requisitos para el proceso ICONIX.

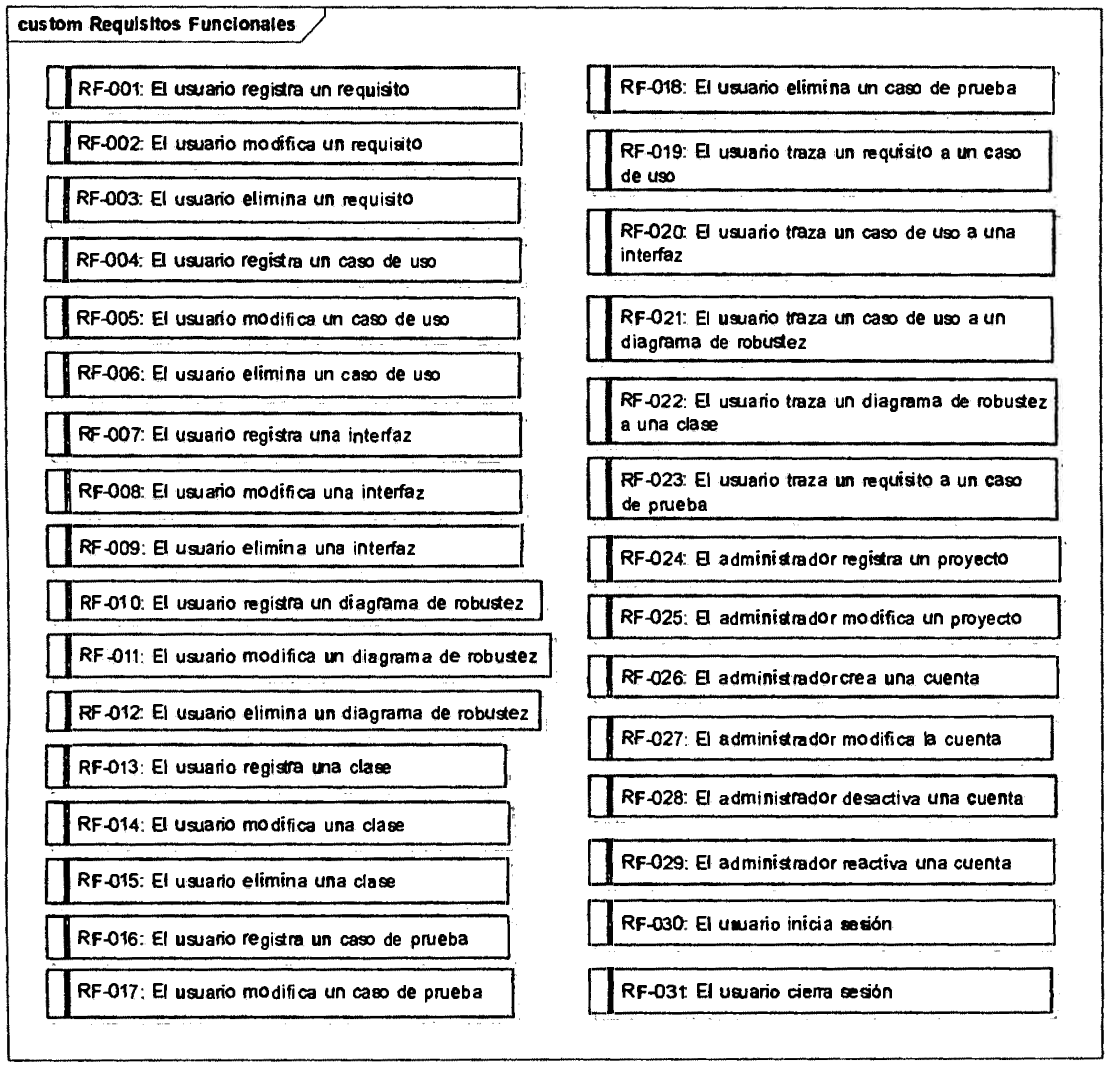


Figura N° 4.2: Requisitos funcionales.

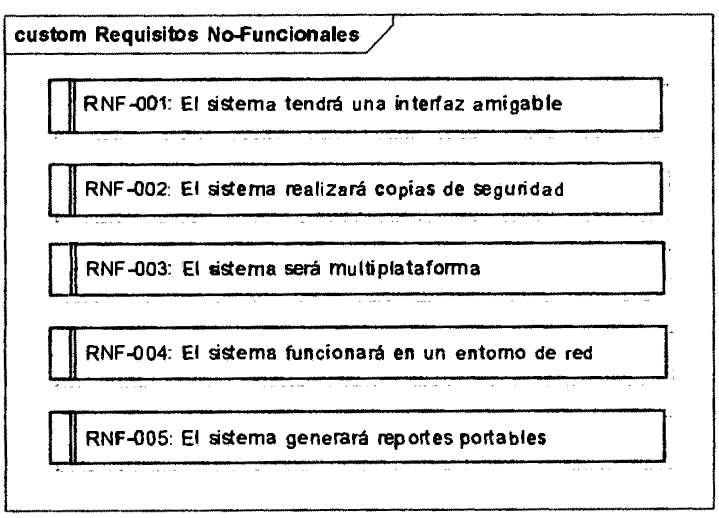


Figura N° 4.3: Requisitos no funcionales.

CASOS DE PRUEBA

CASOS DE PRUEBA	
ID	**Nombre del caso de prueba**
CP-001	Mostrar error de campo nulo
CP-002	Mostrar error de consistencia
CP-003	Mostrar administrar requisitos
CP-004	Listar requisitos
CP-005	Buscar requisito
CP-006	Registrar un requisito
CP-007	Mostrar error requisito no registrado
CP-008	Mostrar requisito registrado satisfactoriamente
CP-009	Modificar un requisito
CP-010	Mostrar error requisito no modificado
CP-011	Mostrar requisito modificado satisfactoriamente
CP-012	Eliminar requisito
CP-013	Mostrar error requisito no eliminado
CP-014	Mostrar requisito eliminado satisfactoriamente
CP-015	Registrar un caso de uso
CP-016	Mostrar error caso de uso no registrado
CP-017	Mostrar caso de uso registrado satisfactoriamente
CP-018	Modificar un caso de uso
CP-019	Mostrar error caso de uso no modificado
CP-020	Mostrar caso de uso modificado satisfactoriamente
CP-021	Eliminar caso de uso
CP-022	Mostrar error caso de uso no eliminado
CP-023	Mostrar caso de uso eliminado satisfactoriamente
CP-024	Registrar una interfaz
CP-025	Mostrar error interfaz no registrada
CP-026	Mostrar interfaz registrada satisfactoriamente
CP-027	Modificación de una interfaz
CP-028	Mostrar error interfaz no modificado
CP-029	Mostrar interfaz modificado satisfactoriamente
CP-030	Eliminar una interfaz
CP-031	Mostrar error interfaz no eliminado
CP-032	Mostrar interfaz eliminado satisfactoriamente
CP-033	Registrar una diagrama de robustez
CP-034	Mostrar error diagrama de robustez no registrado
CP-035	Mostrar diagrama de robustez registrado satisfactoriamente
CP-036	Modificación de un diagrama de robustez

CP-037	Mostrar error diagrama de robustez no modificado
CP-038	Mostrar diagrama de robustez modificado satisfactoriamente
CP-039	Eliminación de un diagrama de robustez sin trazar
CP-040	Mostrar error diagrama de robustez no eliminada
CP-041	Mostrar diagrama de robustez eliminada satisfactoriamente
CP-042	Registrar una clase
CP-043	Mostrar error clase no registrada
CP-044	Mostrar clase registrada satisfactoriamente
CP-045	Modificación de una clase
CP-046	Mostrar error clase no modificada
CP-047	Mostrar clase modificada satisfactoriamente
CP-048	Eliminar una clase
CP-049	Mostrar error clase no eliminado
CP-050	Mostrar clase eliminado satisfactoriamente
CP-051	Registro de un caso de prueba
CP-052	Mostrar error caso de prueba no registrada
CP-053	Mostrar caso de prueba registrada satisfactoriamente
CP-054	Modificación de un caso de prueba
CP-055	Mostrar error caso de prueba no modificada
CP-056	Mostrar caso de prueba modificada satisfactoriamente
CP-057	Eliminar caso de prueba
CP-058	Mostrar error caso de prueba no eliminada
CP-059	Mostrar caso de prueba eliminada satisfactoriamente
CP-060	Listar trazado requisito
CP-061	Listar trazado caso de uso
CP-062	Buscar trazado requisito - caso de uso
CP-063	Mostrar requisito - caso de uso trazado satisfactoriamente
CP-064	Mostrar trazado de requisito - caso de uso eliminado satisfactoriamente
CP-065	Registrar trazado requisito - caso de uso
CP-066	Eliminar trazado requisito - caso de uso
CP-067	Buscar trazado caso de uso - interfaz
CP-068	Mostrar caso de uso - interfaz trazado satisfactoriamente
CP-069	Mostrar trazado de caso de uso - interfaz eliminado satisfactoriamente
CP-070	Registrar trazado caso de uso - interfaz
CP-071	Eliminar trazado caso de uso - interfaz
CP-072	Buscar trazado caso de uso - diagrama de robustez
CP-073	Mostrar caso de uso - diagrama de robustez trazado satisfactoriamente
CP-074	Mostrar trazado de caso de uso - diagrama de robustez eliminado

	satisfactoriamente
CP-075	Registrar trazado caso de uso - diagrama de robustez
CP-076	Eliminar trazado caso de uso - diagrama de robustez
CP-077	Buscar trazado diagrama de robustez - clase
CP-078	Mostrar diagrama de robustez - clase trazado satisfactoriamente
CP-079	Mostrar trazado de diagrama de robustez - clase eliminado satisfactoriamente
CP-080	Registrar trazado diagrama de robustez - clase
CP-081	Eliminar trazado diagrama de robustez - clase
CP-082	Buscar trazado requisito - caso de prueba
CP-083	Mostrar requisito - caso de prueba trazado satisfactoriamente
CP-084	Mostrar trazado de requisito - caso de prueba eliminado satisfactoriamente
CP-085	Registrar trazado requisito - caso de prueba
CP-086	Eliminar trazado requisito - caso de prueba
CP-087	Listar proyecto
CP-088	Buscar proyecto
CP-089	Registrar un proyecto
CP-090	Mostrar error proyecto no registrado
CP-091	Mostrar proyecto registrado satisfactoriamente
CP-092	Modificación de un proyecto
CP-093	Mostrar error proyecto no modificado
CP-094	Mostrar proyecto modificado satisfactoriamente
CP-095	Crear una cuenta
CP-096	Mostrar error en identificador ya existe (ID ocupado)
CP-097	Mostrar error cuenta no creada
CP-098	Mostrar cuenta creada satisfactoriamente
CP-099	Listar cuenta
CP-100	Buscar cuenta
CP-101	Modificar una cuenta
CP-102	Mostrar error cuenta no modificada
CP-103	Mostrar cuenta modificada satisfactoriamente
CP-104	Desactivar una cuenta
CP-105	Mostrar error cuenta no desactivada
CP-106	Mostrar cuenta desactivada satisfactoriamente
CP-107	Reactivar una cuenta
CP-108	Mostrar error cuenta no reactivada
CP-109	Mostrar cuenta reactivada satisfactoriamente
CP-110	Iniciar sesión
CP-111	Mostrar error usuario no existe

CP-112	Mostrar error contraseña incorrecta
CP-113	Mostrar página principal
CP-114	Cerrar sesión
CP-115	Generar reporte
CP-116	Mostrar error reporte no generado
CP-117	Mostrar página de error
CP-118	Cargar reportes portables

Tabla N° 4.2: Casos de prueba.

GLOSARIO DE OBJETOS

1. Usuario
2. Administrador
3. Cuenta
4. Proyecto
5. Requisito
6. Caso de uso
7. Interfaz
8. Diagrama de robustez
9. Clase
10. Caso de prueba
11. Trazado de requisito-caso de uso
12. Trazado de caso de uso-interfaz
13. Trazado caso de uso-diagrama de robustez
14. Trazado diagrama de robustez-clase
15. Trazado requisito-caso de prueba

MODELO DE DOMINIO

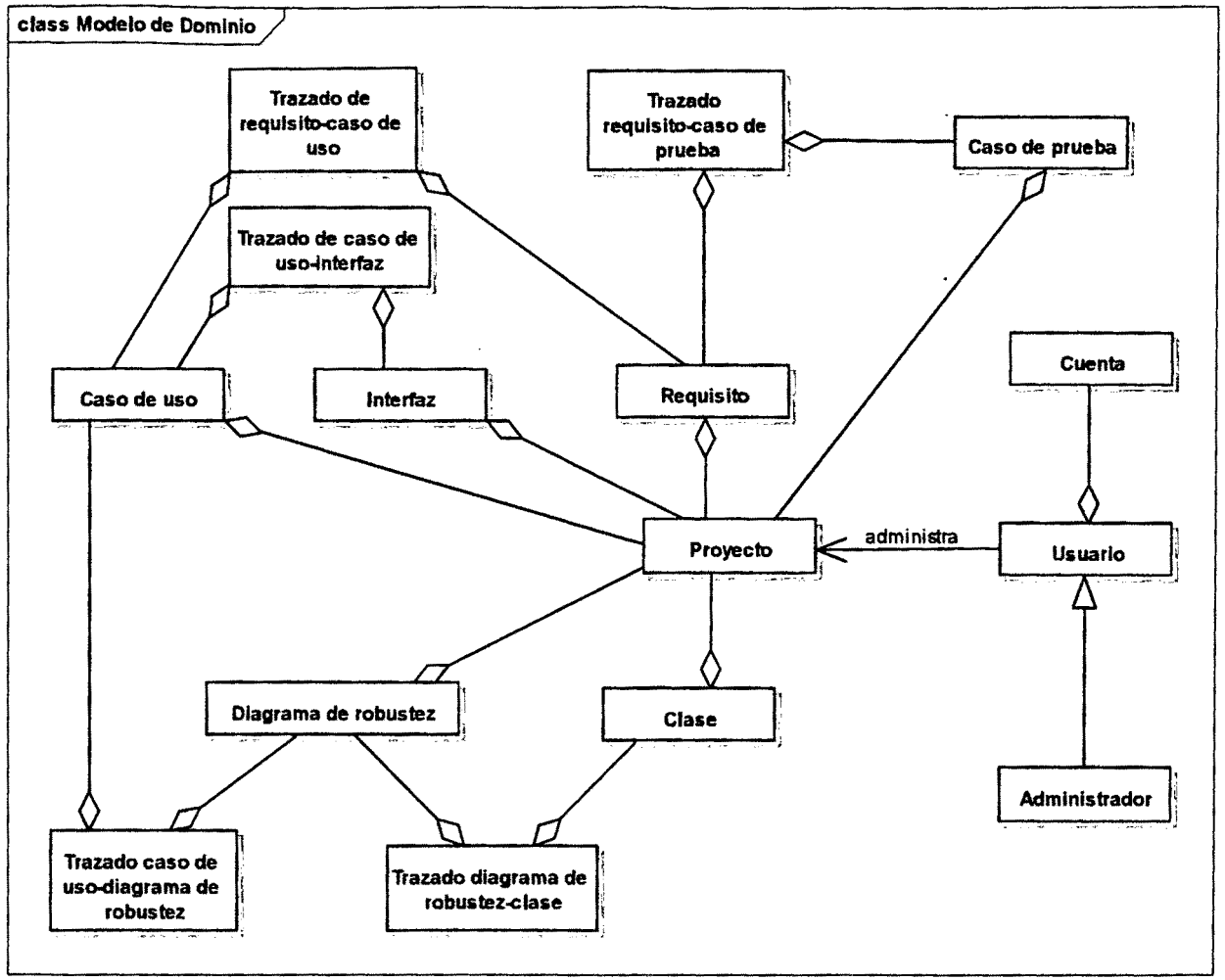


Figura N° 4.4: Modelo de dominio.

PROTOTIPO GUI

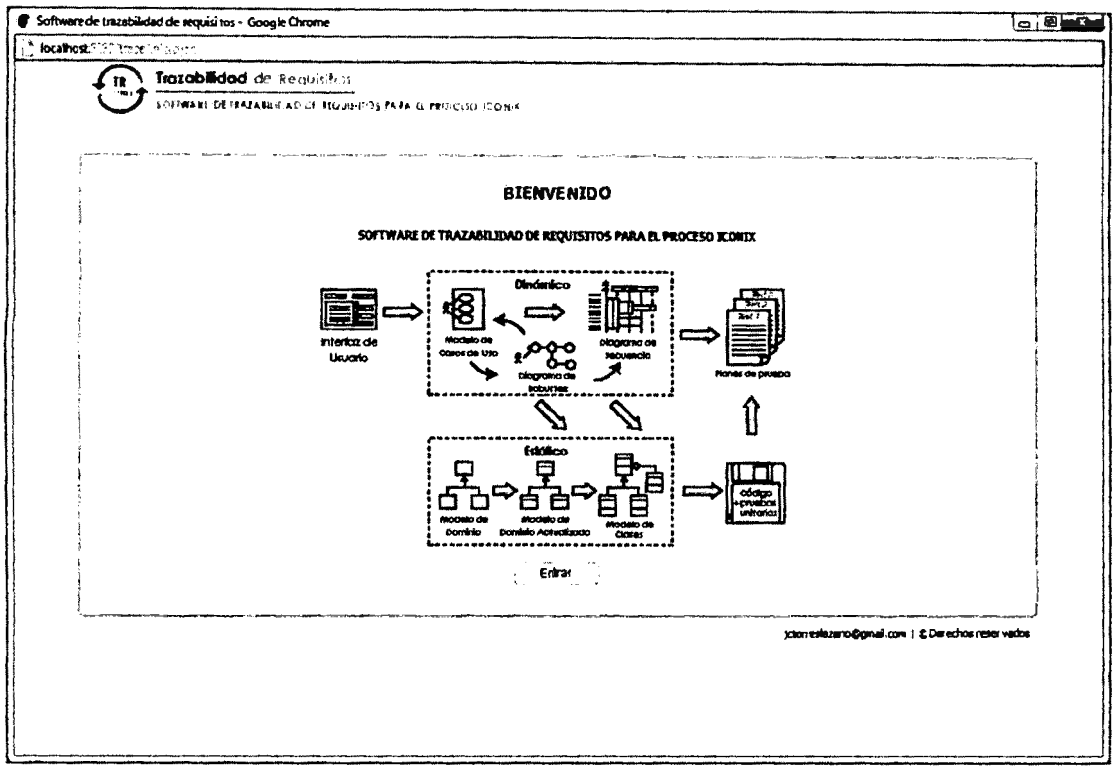


Figura N° 4.5: Interfaz inicial.

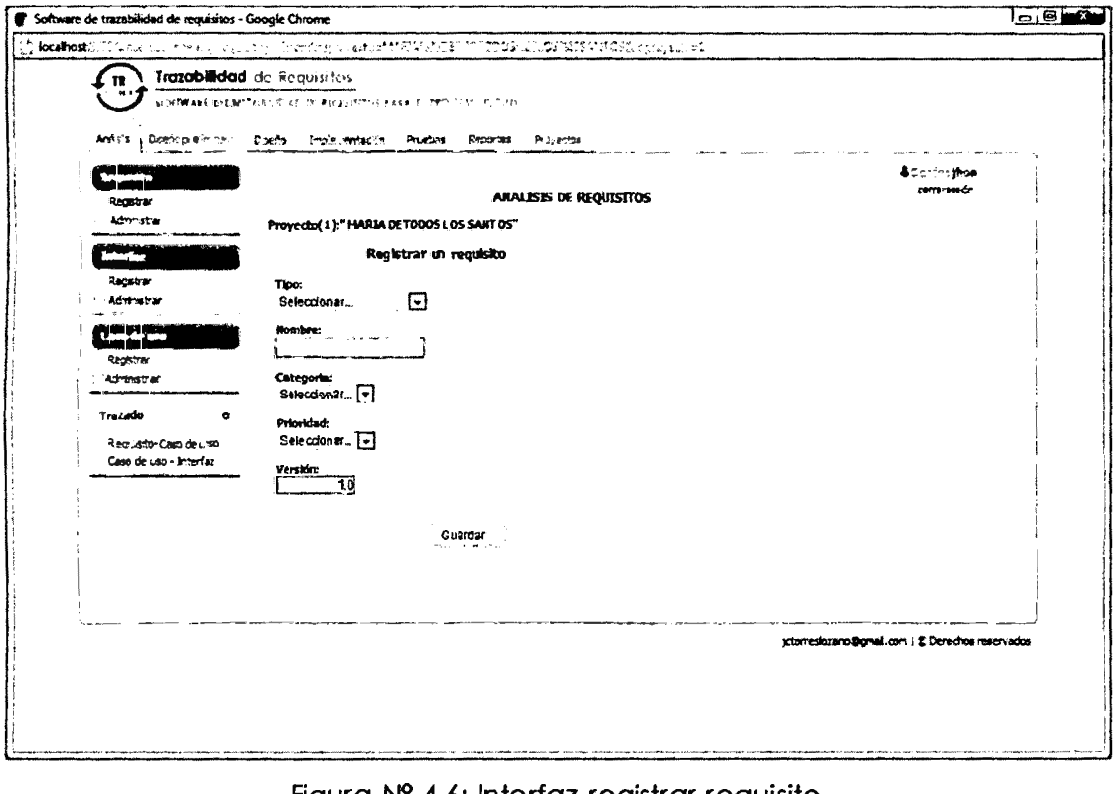


Figura N° 4.6: Interfaz registrar requisito.

LISTADO DE CASOS DE USO

CASOS DE USO	
ID	**Nombre del caso de uso**
CU-001	Registrar un requisito
CU-002	Modificar un requisito
CU-003	Eliminar un requisito
CU-004	Registrar un caso de uso
CU-005	Modificar un caso de uso
CU-006	Eliminar un caso de uso
CU-007	Registrar una interfaz
CU-008	Modificar una interfaz
CU-009	Eliminar una interfaz
CU-010	Registrar un diagrama de robustez
CU-011	Modificar un diagrama de robustez
CU-012	Eliminar un diagrama de robustez
CU-013	Registrar una clase
CU-014	Modificar una clase
CU-015	Eliminar una clase
CU-016	Registrar un caso de prueba
CU-017	Modificar un caso de prueba
CU-018	Eliminar un caso de prueba
CU-019	Trazar un requisito a un caso de uso
CU-020	Trazar un caso de uso a una interfaz
CU-021	Trazar un caso de uso a un diagrama de robustez
CU-022	Trazar un diagrama de robustez a una clase
CU-023	Trazar un requisito a un caso de prueba
CU-024	Registrar un proyecto
CU-025	Modificar un proyecto
CU-026	Crear una cuenta
CU-027	Modificar una cuenta
CU-028	Desactivar una cuenta
CU-029	Reactivar una cuenta
CU-030	Iniciar sesión
CU-031	Cerrar sesión
CU-032	Generar reportes

Tabla N° 4.3: Casos de uso.

EMPAQUETADO DE CASOS DE USO

a. PAQUETE ADMINISTRACIÓN

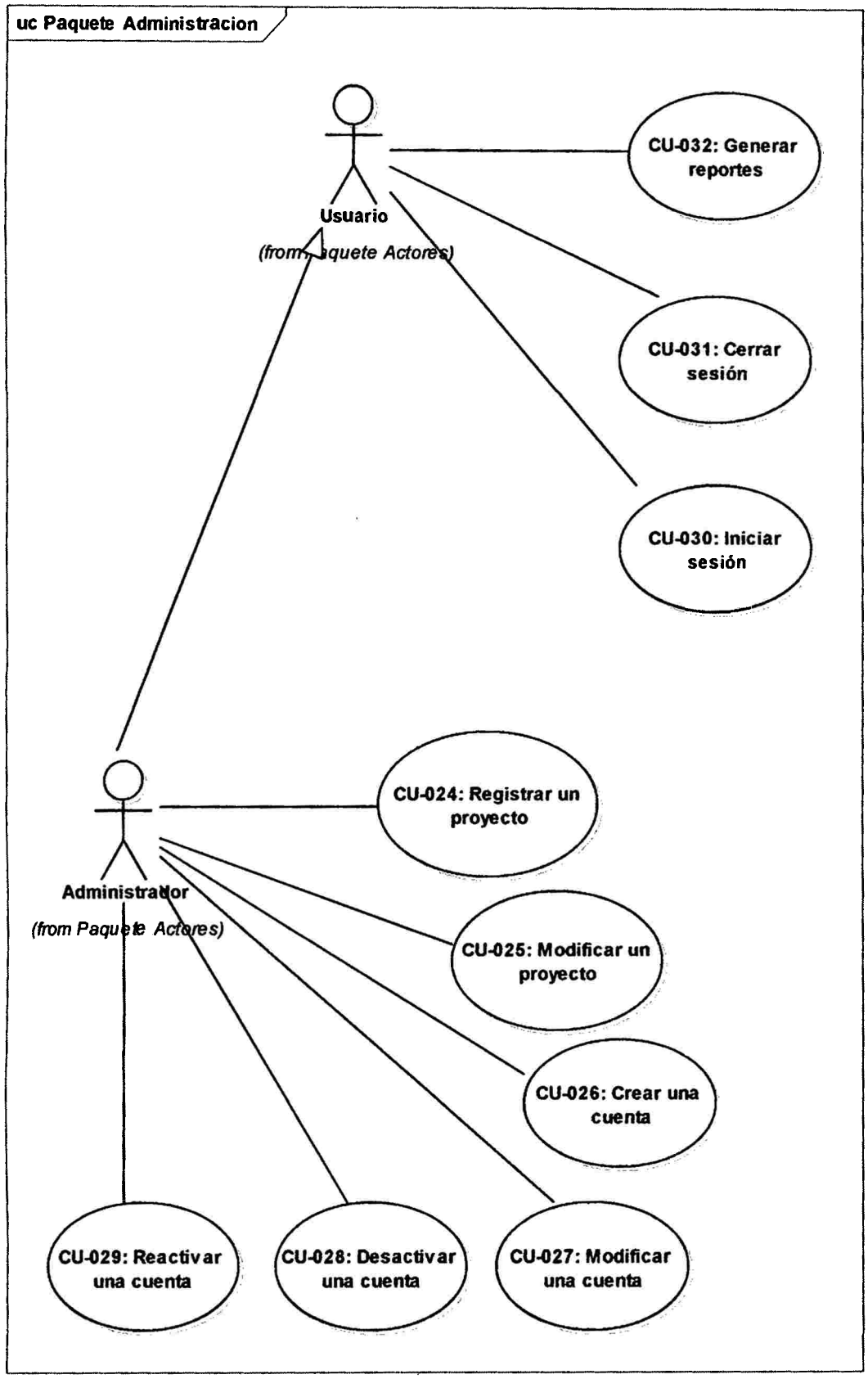


Figura N° 4.7: Paquete administración.

b. PAQUETE ANÁLISIS

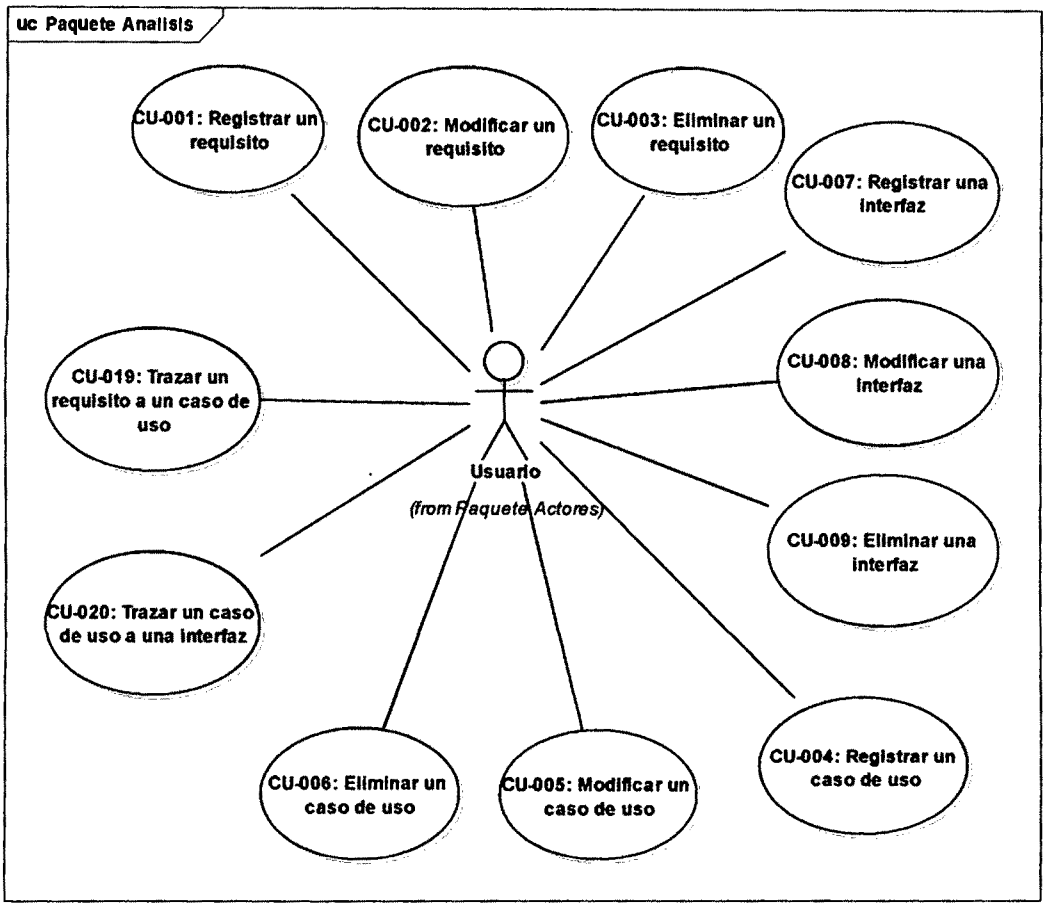


Figura N° 4.8: Paquete análisis.

c. PAQUETE DISEÑO PRELIMINAR

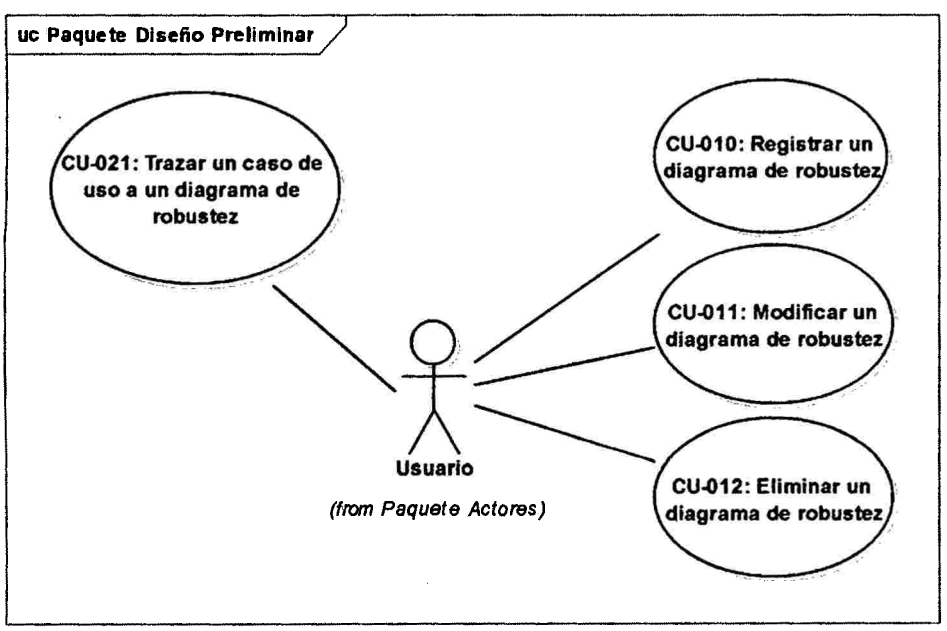


Figura N° 4.9: Paquete diseño preliminar.

d. PAQUETE DISEÑO

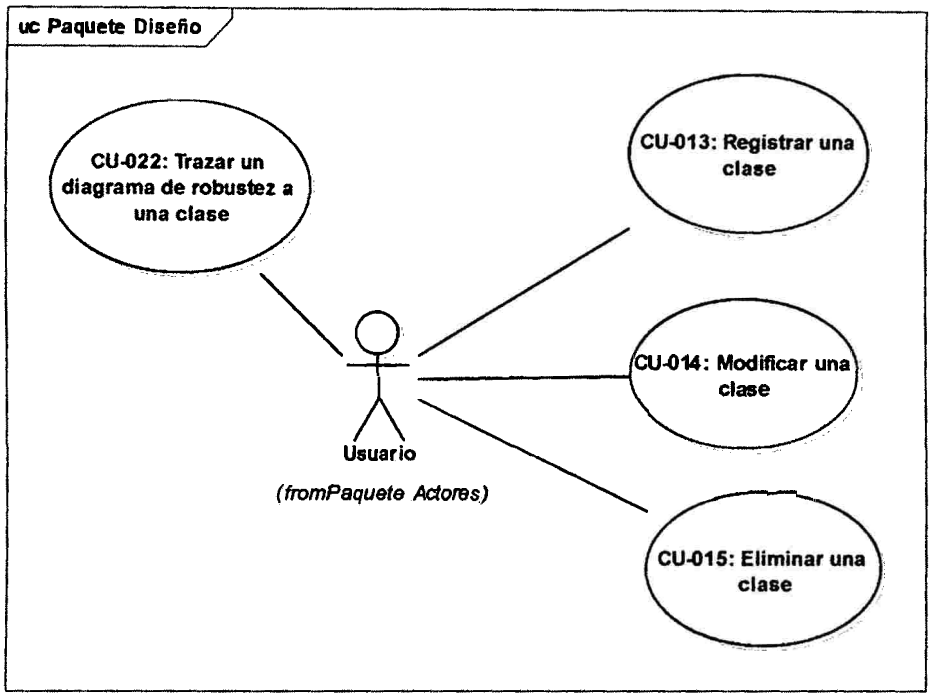


Figura N° 4.10: Paquete diseño.

e. PAQUETE PRUEBAS

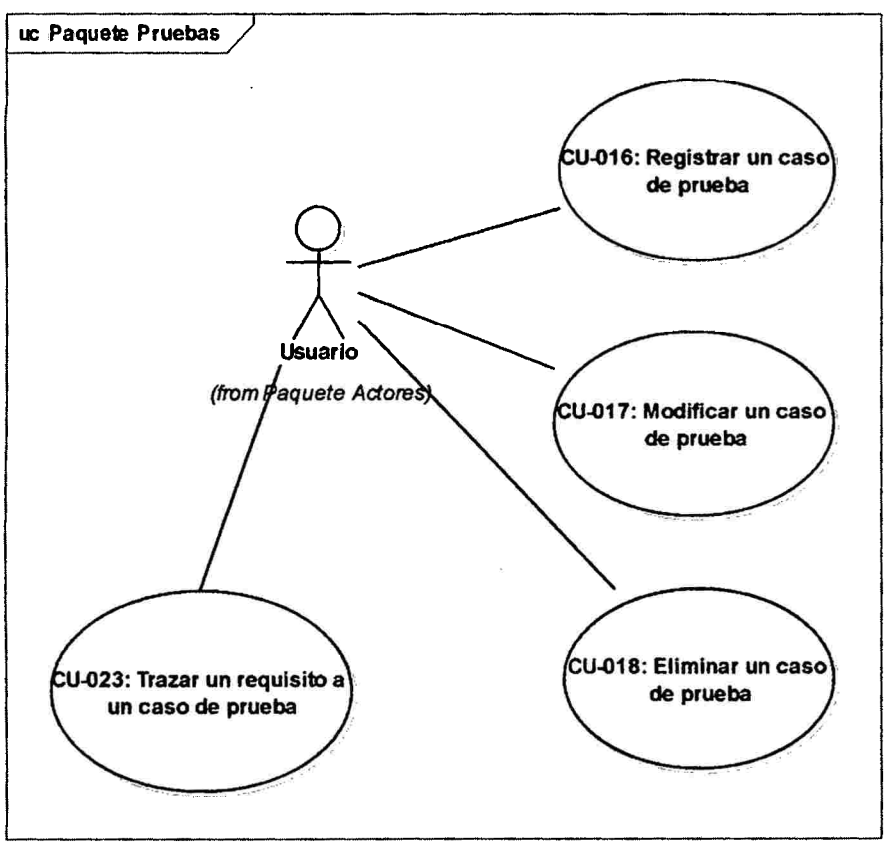


Figura N° 4.11: Paquete pruebas.

PAQUETES DE CASOS DE USO

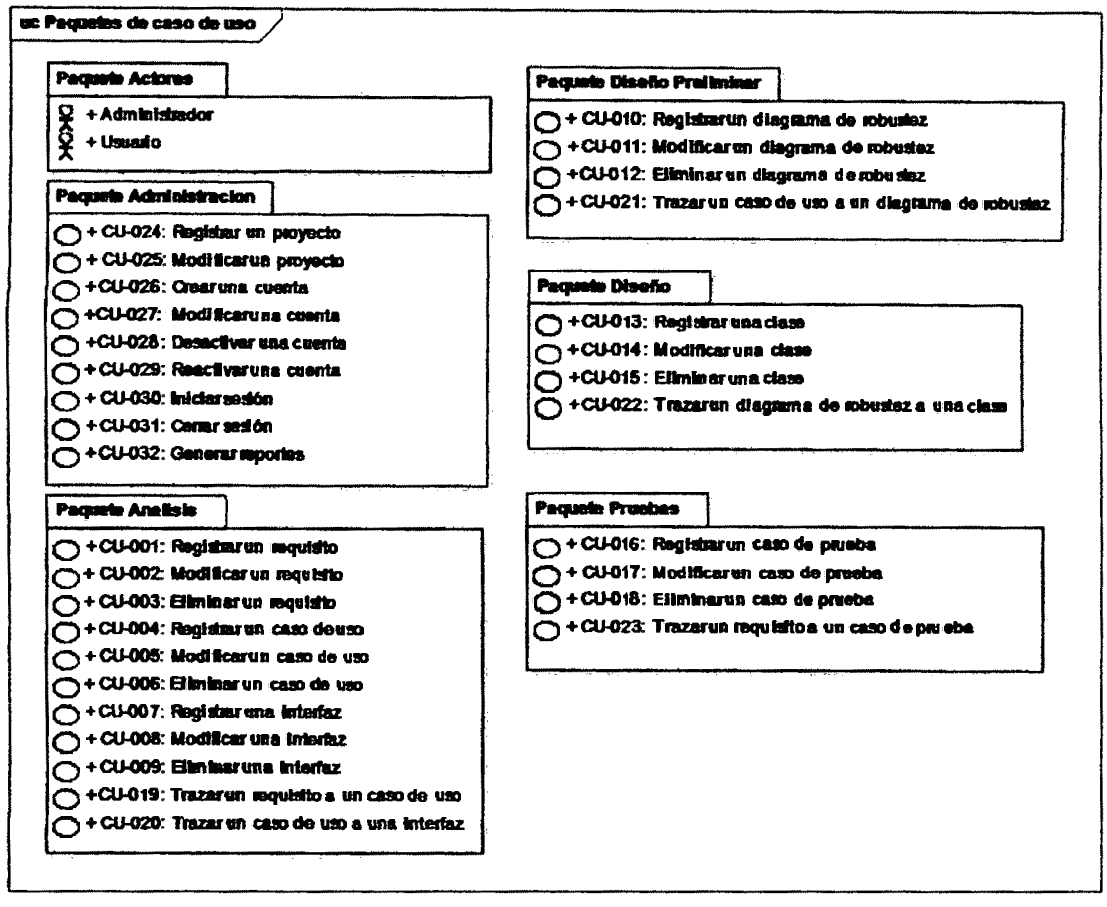


Figura N° 4.12: Paquetes de casos de uso.

RELACIÓN ENTRE REQUISITOS Y CASOS DE USO

REQUISITOS		CASOS DE USO	
ID	**Nombre del requisito**	**ID**	**Nombre del caso de uso**
RF-001	El usuario registra un requisito	CU-001	Registrar un requisito
RF-002	El usuario modifica un requisito	CU-002	Modificar un requisito
RF-003	El usuario elimina un requisito	CU-003	Eliminar un requisito
RF-004	El usuario registra un caso de uso	CU-004	Registrar un caso de uso
RF-005	El usuario modifica un caso de uso	CU-005	Modificar un caso de uso
RF-006	El usuario elimina un caso de uso	CU-006	Eliminar un caso de uso
RF-007	El usuario registra una interfaz	CU-007	Registrar una interfaz
RF-008	El usuario modifica una interfaz	CU-008	Modificar una interfaz

RF-009	El usuario elimina una interfaz	CU-009	Eliminar una interfaz
RF-010	El usuario registra un diagrama de robustez	CU-010	Registrar un diagrama de robustez
RF-011	El usuario modifica un diagrama de robustez	CU-011	Modificar un diagrama de robustez
RF-012	El usuario elimina un diagrama de robustez	CU-012	Eliminar un diagrama de robustez
RF-013	El usuario registra una clase	CU-013	Registrar una clase
RF-014	El usuario modifica una clase	CU-014	Modificar una clase
RF-015	El usuario elimina una clase	CU-015	Eliminar una clase
RF-016	El usuario registra un caso de prueba	CU-016	Registrar un caso de prueba
RF-017	El usuario modifica un caso de prueba	CU-017	Modificar un caso de prueba
RF-018	El usuario elimina un caso de prueba	CU-018	Eliminar un caso de prueba
RF-019	El usuario traza un requisito a un caso de uso	CU-019	Trazar un requisito a un caso de uso
RF-020	El usuario traza un caso de uso a una interfaz	CU-020	Trazar un caso de uso a una interfaz
RF-021	El usuario traza un caso de uso a un diagrama de robustez	CU-021	Trazar un caso de uso a un diagrama de robustez
RF-022	El usuario traza un diagrama de robustez a una clase	CU-022	Trazar un diagrama de robustez a una clase
RF-023	El usuario traza un requisito a un caso de prueba	CU-023	Trazar un requisito a un caso de prueba
RF-024	El administrador registra un proyecto	CU-024	Registrar un proyecto
RF-025	El administrador modifica un proyecto	CU-025	Modificar un proyecto
RF-026	El administrador crea una cuenta	CU-026	Crear una cuenta
RF-027	El administrador modifica la cuenta	CU-027	Modificar una cuenta
RF-028	El administrador desactiva una cuenta	CU-028	Desactivar una cuenta
RF-029	El administrador reactiva una cuenta	CU-029	Reactivar una cuenta
RF-030	El usuario inicia sesión	CU-030	Iniciar sesión
RF-031	El usuario cierra sesión	CU-031	Cerrar sesión
RNF-005	El sistema generará reportes portables	CU-032	Generar reportes

Tabla N° 4.4: Relación entre requisitos y casos de uso.

PRIMER BORRADOR DE CASOS DE USO

CU-001	Registrar un requisito
CURSO BÁSICO: El usuario hace click en el link de registrar requisito, el sistema muestra la página registrar nuevo requisito con los campos correspondientes, el usuario rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda el requisito, el sistema muestra un mensaje de requisito registrado satisfactoriamente en la página resultado de registro.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede registrar el requisito**, el sistema muestra un mensaje de error, que el requisito no ha sido registrado.	

CU-002	Modificar un requisito
CURSO BÁSICO: El usuario hace click en el link de administrar requisitos, el sistema muestra la página lista de requisitos con la lista de requisitos, seleccionamos el requisito a modificar, el sistema muestra el requisito con los campos correspondientes, el usuario modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda el requisito, el sistema muestra un mensaje de requisito modificado satisfactoriamente en la página resultado de modificación.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede modificar el requisito**, el sistema muestra un mensaje de error, que el requisito no ha sido modificado.	

CU-003	Eliminar un requisito
CURSO BÁSICO: El usuario hace click en el link de administrar requisitos, el sistema muestra la página lista de requisitos con la lista de requisitos,	

seleccionamos el requisito a eliminar, el sistema muestra los datos del requisito en la página eliminar requisito, el usuario hace click para confirmar la eliminación, el sistema elimina el requisito y muestra un mensaje de requisito eliminado satisfactoriamente en la página resultado de eliminación.
CURSO ALTERNO: **Cuando se cancela la eliminación**, el sistema regresa a la página de lista de requisitos. **Cuando no se puede eliminar el requisito**, el sistema muestra un mensaje de error, que el requisito no ha sido eliminado.

CU-004	**Registrar un caso de uso**
CURSO BÁSICO: El usuario hace click en el link de registrar caso de uso, el sistema muestra la página registrar nuevo caso de uso con los campos correspondientes, el usuario rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda el caso de uso, el sistema muestra un mensaje de caso de uso registrado satisfactoriamente en la página resultado de registro.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede registrar el caso de uso**, el sistema muestra un mensaje de error, que el caso de uso no ha sido registrado.	

CU-005	**Modificar un caso de uso**
CURSO BÁSICO: El usuario hace click en el link de administrar casos de uso, el sistema muestra la página lista de casos de uso con la lista de casos de uso, seleccionamos el caso de uso a modificar, el sistema muestra el caso de uso con los campos correspondientes, el usuario modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda el caso de uso, el sistema muestra un mensaje de caso de uso modificado satisfactoriamente en la página resultado de modificación.	

CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede modificar el caso de uso**, el sistema muestra un mensaje de error, que el caso de uso no ha sido modificado.

CU-006	**Eliminar un caso de uso**
CURSO BÁSICO: El usuario hace click en el link de administrar casos de uso, el sistema muestra la página lista de caso de uso con la lista de casos de uso, seleccionamos el caso de uso a eliminar, el sistema muestra los datos del caso de uso en la página eliminar caso de uso, el usuario hace click para confirmar la eliminación, el sistema elimina el caso de uso y muestra un mensaje de caso de uso eliminado satisfactoriamente en la página resultado de eliminación.	
CURSO ALTERNO: **Cuando se cancela la eliminación**, el sistema regresa a la página de lista de requisitos. **Cuando no se puede eliminar el caso de uso**, el sistema muestra un mensaje de error, que el caso de uso no ha sido eliminado.	

CU-007	**Registrar una interfaz**
CURSO BÁSICO: El usuario hace click en el link de registrar interfaz, el sistema muestra la página registrar nuevo interfaz con los campos correspondientes, el usuario rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda la interfaz, el sistema muestra un mensaje de interfaz registrada satisfactoriamente en la página resultado de registro.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede registrar la interfaz**, el sistema muestra un mensaje de error, que la interfaz no ha sido registrada.	

CU-008	Modificar una interfaz

CURSO BÁSICO: El usuario hace click en el link de administrar interfaces, el sistema muestra la página lista de interfaces con la lista de interfaces, seleccionamos la interfaz a modificar, el sistema muestra la interfaz con los campos correspondientes, el usuario modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda la interfaz, el sistema muestra un mensaje de interfaz modificado satisfactoriamente en la página resultado de modificación.

CURSO ALTERNO:

Cuando existe un campo nulo, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos.

Cuando no se puede modificar la interfaz, el sistema muestra un mensaje de error, que la interfaz no ha sido modificada.

CU-009	Eliminar una interfaz

CURSO BÁSICO: El usuario hace click en el link de administrar interfaces, el sistema muestra la página lista de interfaces con la lista de interfaces, seleccionamos la interfaz a eliminar, el sistema muestra los datos de la interfaz en la página eliminar interfaz, el usuario hace click para confirmar la eliminación, el sistema elimina la interfaz y muestra un mensaje de interfaz eliminado satisfactoriamente en la página resultado de eliminación.

CURSO ALTERNO:

Cuando se cancela la eliminación, el sistema regresa a la página de lista de interfaces.

Cuando no se puede eliminar la interfaz, el sistema muestra un mensaje de error, que la interfaz no ha sido eliminada.

CU-010	Registrar un diagrama de robustez

CURSO BÁSICO: El usuario hace click en el link de registrar diagrama de robustez, el sistema muestra la página registrar nuevo diagrama de robustez

con los campos correspondientes, el usuario rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda el diagrama de robustez, el sistema muestra un mensaje de diagrama de robustez registrado satisfactoriamente en la página resultado de registro.
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede registrar el diagrama de robustez**, el sistema muestra un mensaje de error, que el diagrama de robustez no ha sido registrado.

CU-011	**Modificar un diagrama de robustez**
CURSO BÁSICO: El usuario hace click en el link de administrar diagramas de robustez, el sistema muestra la página lista de diagramas de robustez con la lista de diagramas de robustez, seleccionamos el diagrama de robustez a modificar, el sistema muestra el diagrama de robustez con los campos correspondientes, el usuario modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda el diagrama de robustez, el sistema muestra un mensaje de diagrama de robustez modificado satisfactoriamente en la página resultado de modificación.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede modificar el diagrama de robustez**, el sistema muestra un mensaje de error, que el diagrama de robustez no ha sido modificado.	

CU-012	**Eliminar un diagrama de robustez**
CURSO BÁSICO: El usuario hace click en el link de administrar diagramas de robustez, el sistema muestra la página lista de diagramas de robustez con la lista de diagramas de robustez, seleccionamos el diagrama de robustez a eliminar, el sistema muestra los datos del diagrama de robustez en la página eliminar diagrama de robustez, el usuario hace click para confirmar la	

eliminación, el sistema elimina el diagrama de robustez y muestra un mensaje de diagrama de robustez eliminado satisfactoriamente en la página resultado de eliminación.
CURSO ALTERNO: **Cuando se cancela la eliminación**, el sistema regresa a la página de lista de diagramas de robustez. **Cuando no se puede eliminar el diagrama de robustez**, el sistema muestra un mensaje de error, que el diagrama de robustez no ha sido eliminado.

CU-013	Registrar una clase
CURSO BÁSICO: El usuario hace click en el link de registrar clase, el sistema muestra la página registrar nuevo clase con los campos correspondientes, el usuario rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda la clase, el sistema muestra un mensaje de clase registrado satisfactoriamente en la página resultado de registro.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede registrar la clase**, el sistema muestra un mensaje de error, que la clase no ha sido registrada.	

CU-014	Modificar una clase
CURSO BÁSICO: El usuario hace click en el link de administrar clases, el sistema muestra la página lista de clases con la lista de clases, seleccionamos la clase a modificar, el sistema muestra la clase con los campos correspondientes, el usuario modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda la clase, el sistema muestra un mensaje de clase modificado satisfactoriamente en la página resultado de modificación.	
CURSO ALTERNO:	

Cuando existe un campo nulo, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede modificar la clase**, el sistema muestra un mensaje de error, que la clase no ha sido modificada.

CU-015	Eliminar una clase
CURSO BÁSICO: El usuario hace click en el link de administrar clases, el sistema muestra la página lista de clases con la lista de clases, seleccionamos la clase a eliminar, el sistema muestra los datos de la clase en la página eliminar clase, el usuario hace click para confirmar la eliminación, el sistema elimina la clase y muestra un mensaje de clase eliminado satisfactoriamente en la página resultado de eliminación.	
CURSO ALTERNO: **Cuando se cancela la eliminación**, el sistema regresa a la página de lista de clases. **Cuando no se puede eliminar la clase**, el sistema muestra un mensaje de error, que la clase no ha sido eliminada.	

CU-016	Registrar un caso de prueba
CURSO BÁSICO: El usuario hace click en el link de registrar caso de prueba, el sistema muestra la página registrar nuevo caso de prueba con los campos correspondientes, el usuario rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda el caso de prueba, el sistema muestra un mensaje de caso de prueba registrado satisfactoriamente en la página resultado de registro.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede registrar el caso de prueba**, el sistema muestra un mensaje de error, que el caso de prueba no ha sido registrado.	

CU-017	Modificar un caso de prueba
CURSO BÁSICO: El usuario hace click en el link de administrar casos de prueba, el sistema muestra la página lista de caso de pruebas con la lista de casos de prueba, seleccionamos el caso de prueba a modificar, el sistema muestra el caso de prueba con los campos correspondientes, el usuario modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda el caso de prueba, el sistema muestra un mensaje de caso de prueba modificado satisfactoriamente en la página resultado de modificación.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede modificar el caso de prueba**, el sistema muestra un mensaje de error, que el caso de prueba no ha sido modificado.	

CU-018	Eliminar un caso de prueba
CURSO BÁSICO: El usuario hace click en el link de administrar casos de prueba, el sistema muestra la página lista de casos de prueba con la lista de casos de prueba, seleccionamos el caso de prueba a eliminar, el sistema muestra los datos del caso de prueba en la página eliminar caso de prueba, el usuario hace click para confirmar la eliminación, el sistema elimina el caso de prueba y muestra un mensaje de caso de prueba eliminado satisfactoriamente en la página resultado de eliminación.	
CURSO ALTERNO: **Cuando se cancela la eliminación,** el sistema regresa a la página de lista de casos de prueba. **Cuando no se puede eliminar el caso de prueba**, el sistema muestra un mensaje de error, que el caso de prueba no ha sido eliminado.	

CU-019	Trazar un requisito a un caso de uso
CURSO BÁSICO: El usuario hace click en el link de administrar trazado de	

requisito-a-caso de uso, el sistema muestra página lista de trazado de requisito con la lista de requisitos, el usuario selecciona la acción(trazar, eliminar), el sistema muestra los detalles del requisito y la lista de casos de uso, si el usuario hace click en trazar a un caso de uso, el sistema verifica la reglas de consistencia y guarda el trazado, y muestra un mensaje de requisito trazado satisfactoriamente; si el usuario hace click en eliminar, el sistema elimina el trazado y muestra un mensaje de trazado eliminado satisfactoriamente.
CURSO ALTERNO: **Cuando un requisito funcional traza a más de un caso de uso**, el sistema muestra un mensaje que el requisito funcional ya fue trazado a un caso de uso (error de consistencia).

CU-020	**Trazar un caso de uso a una interfaz**
CURSO BÁSICO: El usuario hace click en el link de administrar trazado de caso de uso a interfaz, el sistema muestra página lista de trazado de caso de uso con la lista de casos de uso, el usuario selecciona la acción(trazar, eliminar), el sistema muestra los detalles del caso de uso y la lista de interfaces, si el usuario hace click en trazar a una interfaz, el sistema verifica la reglas de consistencia y guarda el trazado, y muestra un mensaje de caso de uso trazado satisfactoriamente; si el usuario hace click en eliminar, el sistema elimina el trazado y muestra un mensaje de trazado eliminado satisfactoriamente.	
CURSO ALTERNO: **Cuando no se puede trazar el caso de uso**, el sistema muestra un mensaje de error, que el caso de uso no ha sido trazado.	

CU-021	**Trazar un caso de uso a un diagrama de robustez**
CURSO BÁSICO: El usuario hace click en el link de administrar trazado de caso de uso a diagrama de robustez, el sistema muestra página lista de trazado de caso de uso con la lista de casos de uso, el usuario selecciona la	

acción(trazar, eliminar), el sistema muestra los detalles del caso de uso y la lista de diagramas de robustez, si el usuario hace click en trazar a un diagrama de robustez, el sistema verifica la reglas de consistencia y guarda el trazado, y muestra un mensaje de caso de uso trazado satisfactoriamente; si el usuario hace click en eliminar, el sistema elimina el trazado y muestra un mensaje de trazado eliminado satisfactoriamente.
CURSO ALTERNO: **Cuando un caso de uso traza a más de un diagrama de robustez,** el sistema muestra un mensaje que el caso de uso ya fue trazado a un diagrama de robustez (error de consistencia).

CU-022	**Trazar un diagrama de robustez a una clase**
CURSO BÁSICO: El usuario hace click en el link de administrar trazado de diagrama de robustez a clase, el sistema muestra página lista de trazado de diagrama de robustez con la lista de diagramas de robustez, el usuario selecciona la acción(trazar, eliminar), el sistema muestra los detalles del diagrama de robustez y la lista de clases, si el usuario hace click en trazar a una clase, el sistema verifica la reglas de consistencia y guarda el trazado, y muestra un mensaje de diagrama de robustez trazado satisfactoriamente; si el usuario hace click en eliminar, el sistema elimina el trazado y muestra un mensaje de trazado eliminado satisfactoriamente.	
CURSO ALTERNO: **Cuando no se puede trazar el diagrama de robustez,** el sistema muestra un mensaje de error, que el diagrama de robustez no ha sido trazado.	

CU-023	**Trazar un requisito a un caso de prueba**
CURSO BÁSICO: El usuario hace click en el link de administrar trazado de requisito-a-caso de prueba, el sistema muestra página lista de trazado de requisito con la lista de requisitos, el usuario selecciona la acción(trazar, eliminar), el sistema muestra los detalles del requisito y la lista de casos de prueba, si el usuario hace click en trazar a un caso de prueba, el sistema	

verifica la reglas de consistencia y guarda el trazado, y muestra un mensaje de requisito trazado satisfactoriamente; si el usuario hace click en eliminar, el sistema elimina el trazado y muestra un mensaje de trazado eliminado satisfactoriamente.
CURSO ALTERNO: **Cuando no se puede trazar el requisito**, el sistema muestra un mensaje de error, que el requisito no ha sido trazado.

CU-024	Registrar un proyecto
CURSO BÁSICO: El administrador hace click en el link de registrar proyecto, el sistema muestra la página registrar nuevo proyecto con los campos correspondientes, el administrador rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda el proyecto, el sistema muestra un mensaje de proyecto registrado satisfactoriamente en la página resultado de registro.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando no se puede registrar el proyecto**, el sistema muestra un mensaje de error, que el proyecto no ha sido registrado.	

CU-025	Modificar un proyecto
CURSO BÁSICO: El administrador hace click en el link de administrar proyectos, el sistema muestra la página lista de proyectos con la lista de proyectos, seleccionamos el proyecto a modificar, el sistema muestra el proyecto con los campos correspondientes, el administrador modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda el proyecto, el sistema muestra un mensaje de proyecto modificado satisfactoriamente en la página resultado de modificación.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le	

pide que vuelva a ingresar los datos. **Cuando no se puede modificar el proyecto**, el sistema muestra un mensaje de error, que el proyecto no ha sido modificado.

CU-026	Crear una cuenta
CURSO BÁSICO: El administrador hace click en el link crear una cuenta, el sistema muestra la página crear nueva cuenta con los campos correspondientes, el administrador rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda la cuenta, el sistema muestra un mensaje de la cuenta se ha creado satisfactoriamente en la página resultado de cuenta.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando el identificador de usuario ya existe**, el sistema muestra un mensaje de error, que el identificador de usuario ya existe. **Cuando no se puede crear la cuenta**, el sistema muestra un mensaje de error, que la cuenta no ha sido creada.	

CU-027	Modificar una cuenta
CURSO BÁSICO: El administrador hace click en el link de administrar cuentas, el sistema muestra la página lista de cuentas con la lista de cuentas, seleccionamos la cuenta a modificar, el sistema muestra la cuenta con los campos correspondientes, el administrador modifica los campos y hace click en el botón guardar, el sistema valida los campos y guarda la cuenta, el sistema muestra un mensaje de que la cuenta se ha modificado satisfactoriamente en la página resultado de cuenta.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos.	

Cuando no se puede modificar la cuenta, el sistema muestra un mensaje de error, que la cuenta no ha sido modificada.

CU-028	Desactivar una cuenta
CURSO BÁSICO: El administrador hace click en el link de administrar cuentas, el sistema muestra la página lista de cuentas con la lista de cuentas, seleccionamos la cuenta a desactivar, el sistema muestra la cuenta con los campos correspondientes, el administrador confirma la desactivación de la cuenta y hace click en el botón guardar, el sistema guarda los cambios, el sistema muestra un mensaje de que la cuenta se ha desactivado satisfactoriamente en la página resultado de cuenta.	
CURSO ALTERNO: **Cuando no se puede desactivar la cuenta**, el sistema muestra un mensaje de error, que la cuenta no ha sido desactivada.	

CU-029	Reactivar una cuenta
CURSO BÁSICO: El administrador hace click en el link de administrar cuentas, el sistema muestra la página lista de cuentas con la lista de cuentas, seleccionamos la cuenta a reactivar, el sistema muestra la cuenta con los campos correspondientes, el administrador confirma la reactivación de la cuenta y hace click en el botón guardar, el sistema guarda los cambios, el sistema muestra un mensaje de que la cuenta se ha reactivado satisfactoriamente en la página resultado de cuenta.	
CURSO ALTERNO: **Cuando no se puede reactivar la cuenta**, el sistema muestra un mensaje de error, que la cuenta no ha sido reactivada.	

CU-030	Iniciar sesión
CURSO BÁSICO: El usuario ingresa al sistema y hace click en entrar, el sistema muestra la página de autenticación, el usuario ingresa su identificador de usuario y contraseña, el sistema valida los campos y verifica los datos, el	

sistema muestra la página principal para el usuario.
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error y le pide que vuelva a ingresar los datos. **Cuando el identificador de usuario no existe**, el sistema muestra un mensaje de error, que la cuenta no existe. **Cuando la contraseña es incorrecta**, el sistema muestra un mensaje de error, que la contraseña es incorrecta y cuenta como un intento de inicio de sesión. **Cuando una contraseña incorrecta es ingresada mas de tres veces**, el sistema muestra un mensaje de error, que la cuenta de usuario ha sido bloqueada temporalmente y le muestra un tiempo de espera antes de volver a intentar. **Cuando la cuenta de usuario esta desactivada**, el sistema muestra un mensaje de error, que la cuenta no existe. **Cuando la cuenta de usuario esta desactivada temporalmente**, el sistema muestra un mensaje de error, que la cuenta de usuario está bloqueada temporalmente.

CU-031	**Cerrar sesión**
CURSO BÁSICO: El usuario hace click en cerrar sesión, el sistema cierra la sesión y muestra la página inicial del sistema.	
CURSO ALTERNO: **Cuando no se puede cerrar la sesión**, el sistema muestra un mensaje de error, y un mensaje en la página de error del sistema.	

CU-032	**Generar reportes**
CURSO BÁSICO: El usuario hace click en el link de reportes, el sistema muestra la lista de reportes del proyecto (la lista de artefactos trazables y las matrices de trazabilidad), el usuario hace click en el reporte deseado, el sistema carga el reporte y lo muestra en un formato portable.	

CURSO ALTERNO:

Cuando no se puede generar el reporte, el sistema muestra un mensaje de error, que el reporte no se ha generado.

HITO 1. REVISIÓN DE REQUISITOS

A partir de este hito presento solamente los artefactos para los casos de uso; registrar caso de uso y trazar un requisito a un caso de uso.

REVISIÓN DEL MODELO DE DOMINIO

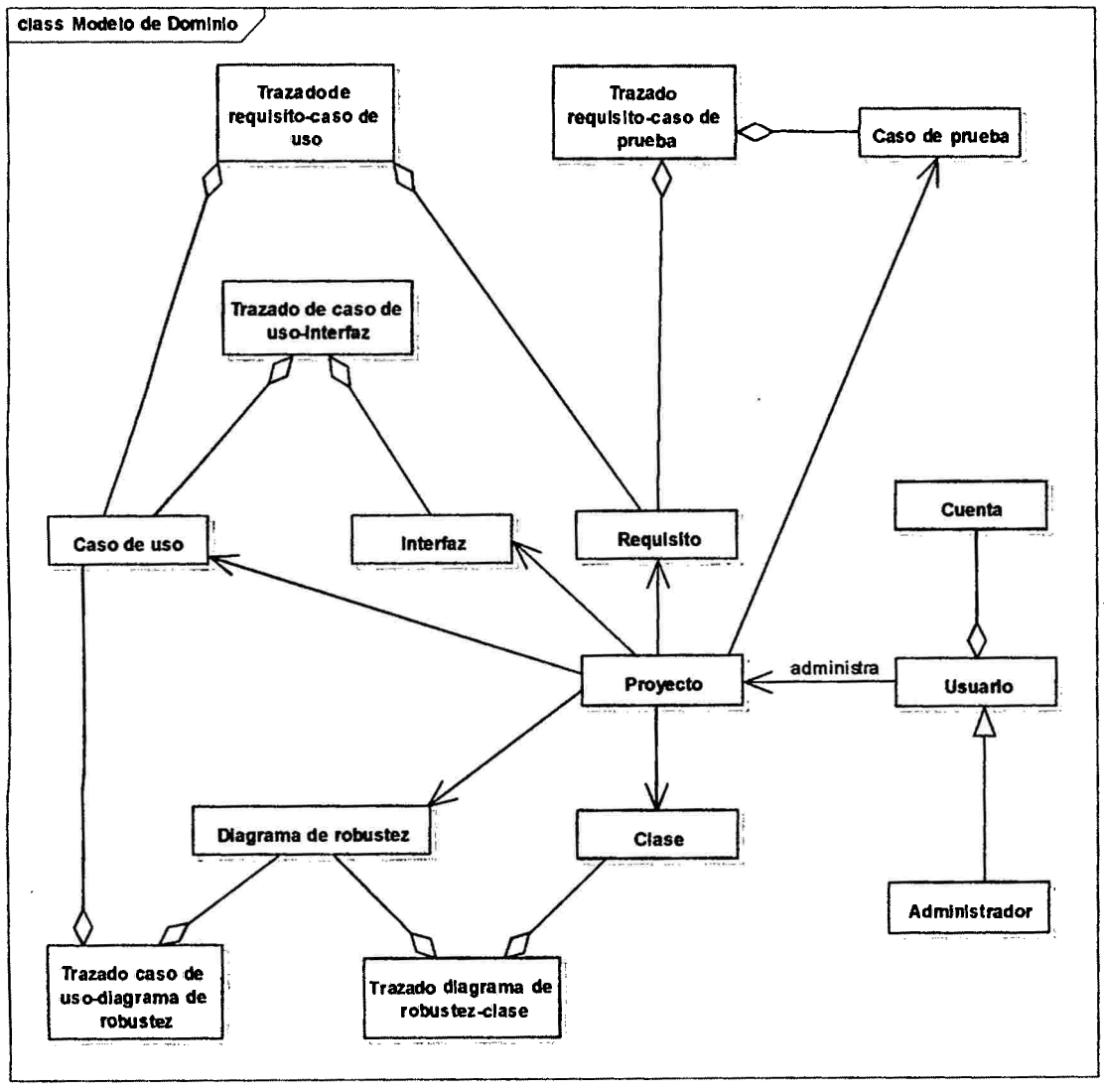


Figura N° 4.13: Modelo de dominio revisado

REVISIÓN DE LOS PROTOTIPOS GUI

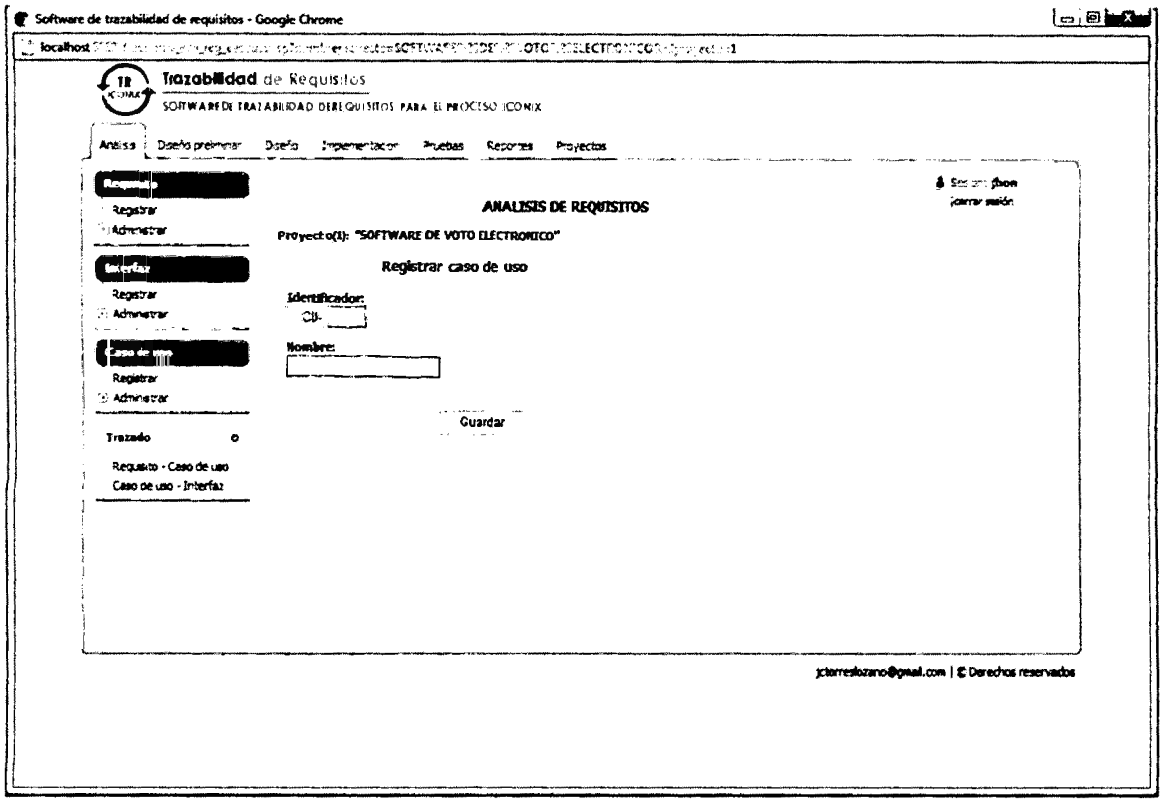


Figura N° 4.14: Interfaz de registro de un caso de uso.

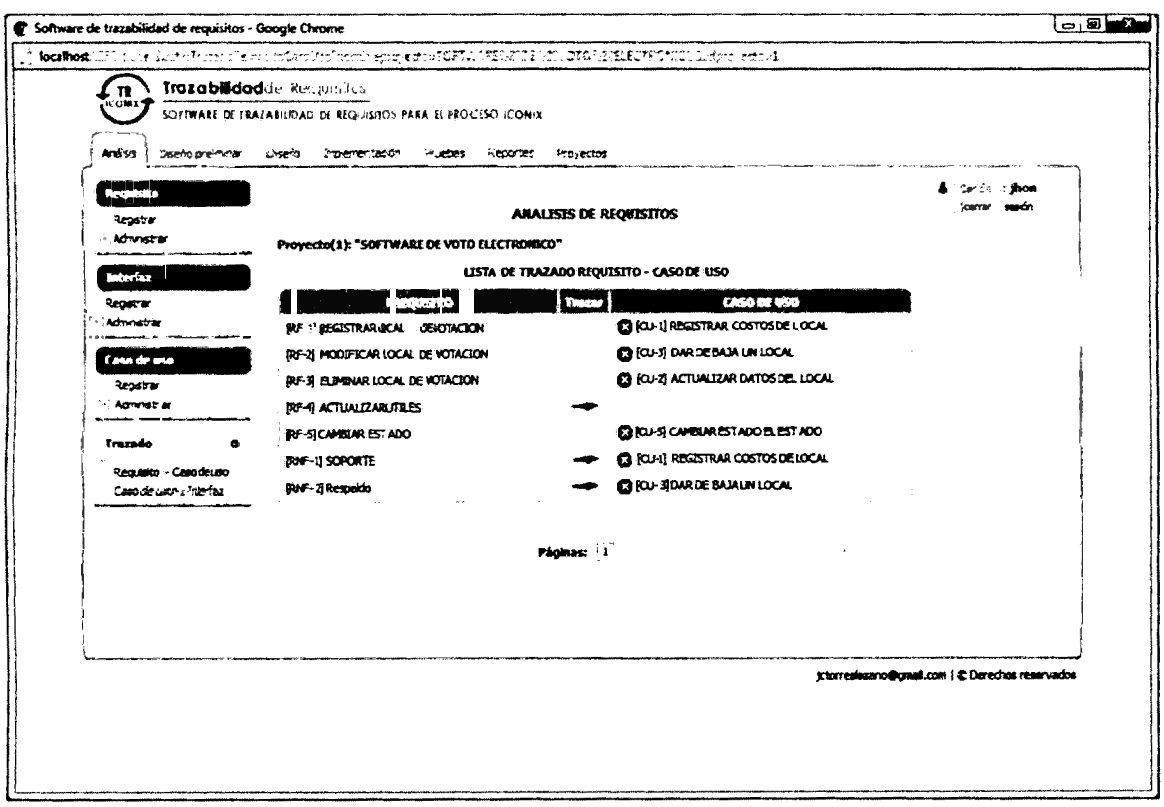


Figura N° 4.15: Interfaz de trazado de un requisito a un caso de uso.

REVISIÓN DE LOS CASOS DE USO

CU-004	Registrar un caso de uso
CURSO BÁSICO: El usuario hace click en el link de registrar caso de uso, el sistema muestra la página registrar nuevo caso de uso con los campos correspondientes, el usuario rellena los campos y hace click en el botón guardar, el sistema valida los campos y guarda el caso de uso, el sistema muestra un mensaje de caso de uso registrado satisfactoriamente en la página resultado de registro.	
CURSO ALTERNO: **Cuando existe un campo nulo**, el sistema muestra un mensaje de error que es un campo necesario. **Cuando ingresa un carácter no valido para la numeración**, el sistema muestra un mensaje de error que se requiere un número. **Cuando no se puede registrar el caso de uso**, el sistema muestra un mensaje de error, que el caso de uso no ha sido registrado.	

CU-019	Trazar un requisito a un caso de uso
CURSO BÁSICO: El usuario hace click en el link de administrar trazado de requisito-a-caso de uso, el sistema muestra página lista de trazado de requisito con la lista de requisitos, el usuario selecciona la acción(trazar, eliminar), el sistema muestra los detalles del requisito y la lista de casos de uso, si el usuario hace click en trazar a un caso de uso, el sistema verifica la reglas de consistencia y guarda el trazado, y muestra un mensaje de requisito trazado satisfactoriamente; si el usuario hace click en eliminar, el sistema elimina el trazado y muestra un mensaje de trazado eliminado satisfactoriamente.	
CURSO ALTERNO: **Cuando un requisito funcional traza a más de un caso de uso**, el sistema muestra un mensaje que el requisito funcional ya fue trazado a un caso de	

uso (error de consistencia).

Cuando el trazado no es registrado, el sistema muestra un mensaje de error el trazado requisito – caso de uso no fue registrado.

Cuando el trazado no puede ser eliminado, el sistema muestra un mensaje de error el trazado requisito – caso de uso no fue eliminado.

B. ARQUITECTURA TÉCNICA

DIAGRAMA DE COMPONENTES

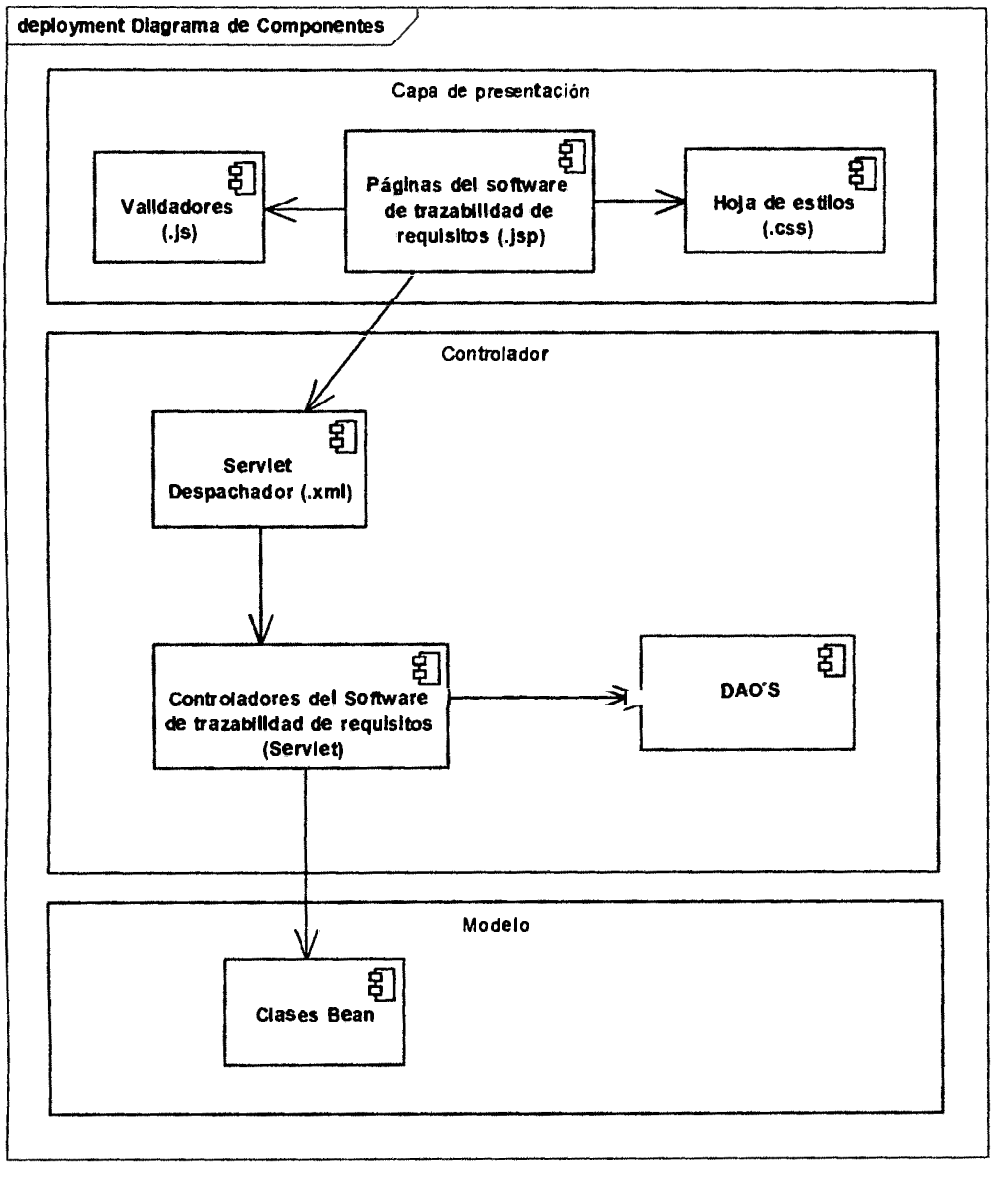


Figura N° 4.16: Diagrama de componentes.

DIAGRAMA DE DESPLIEGUE

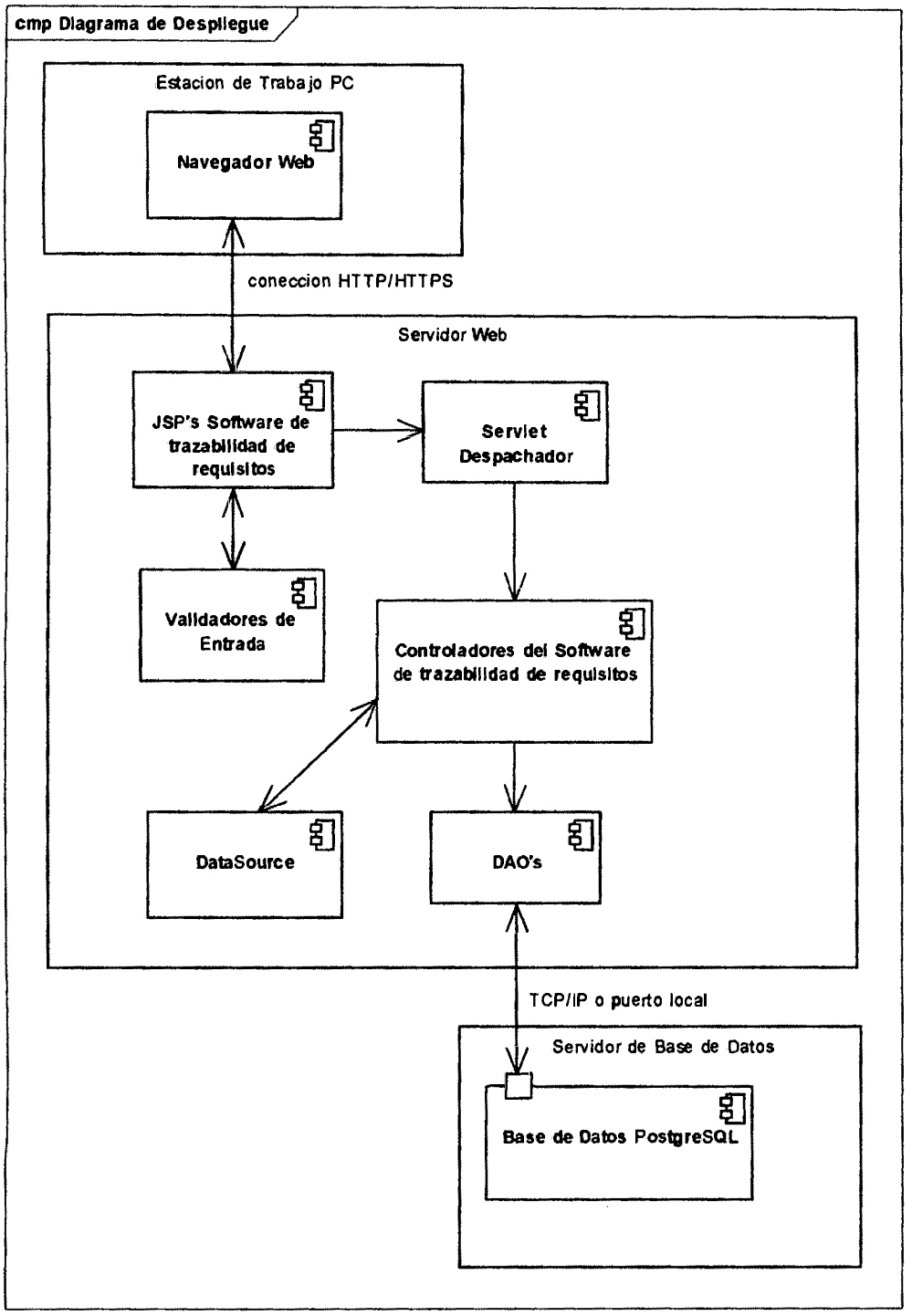


Figura Nº 4.17: Diagrama de despliegue.

C. DISEÑO PRELIMINAR

DIAGRAMA DE ROBUSTEZ

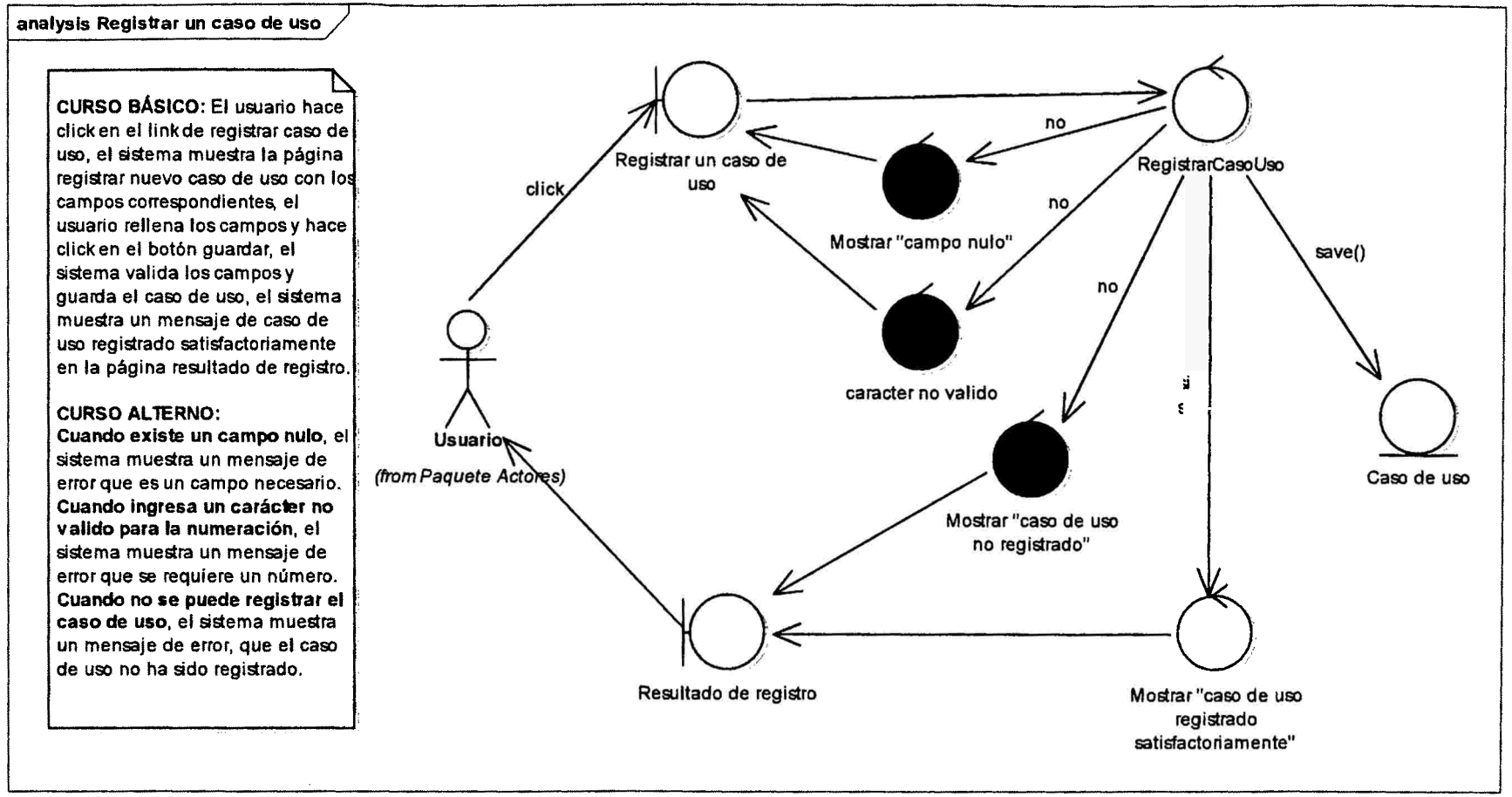


Figura N° 4.18: Diagrama de robustez registrar caso de uso.

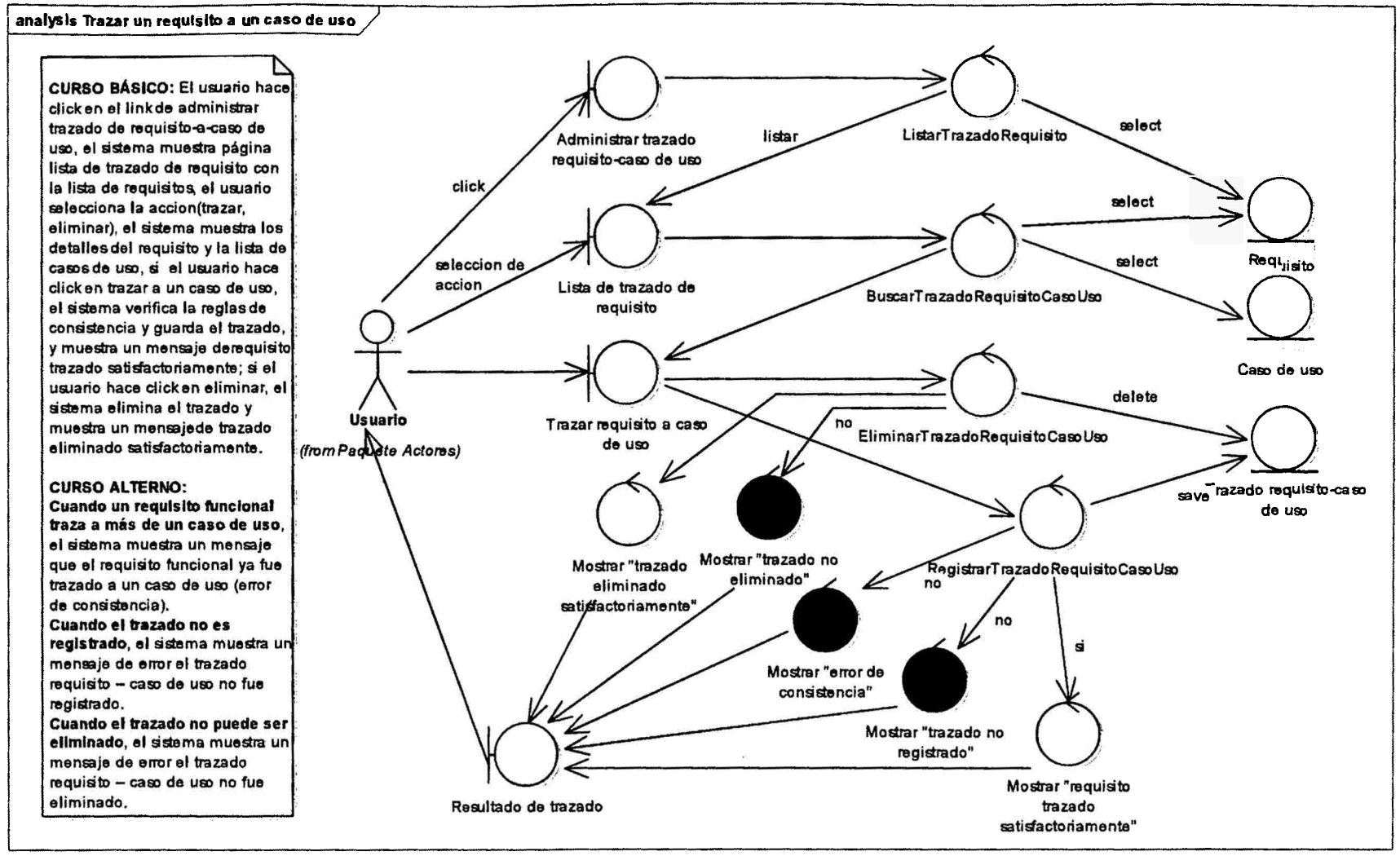


Figura Nº 4.19: Diagrama de robustez trazado requisito - caso de uso.

MODELO DE DOMINIO ACTUALIZADO

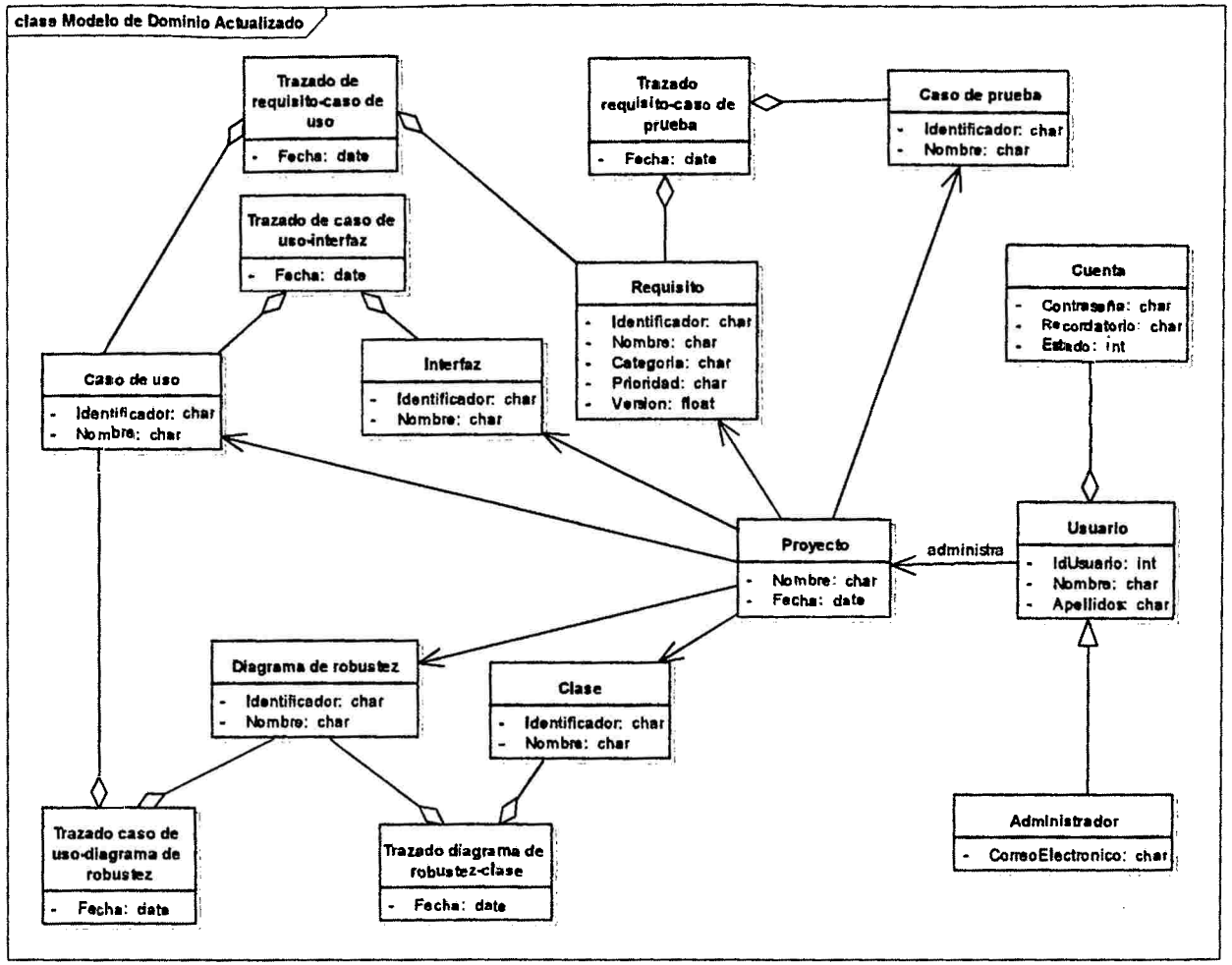


Figura N° 4.20: Modelo de dominio actualizado.

HITO 2. REVISIÓN DE DISEÑO PRELIMINAR

REVISION DEL DIAGRAMA DE ROBUSTEZ

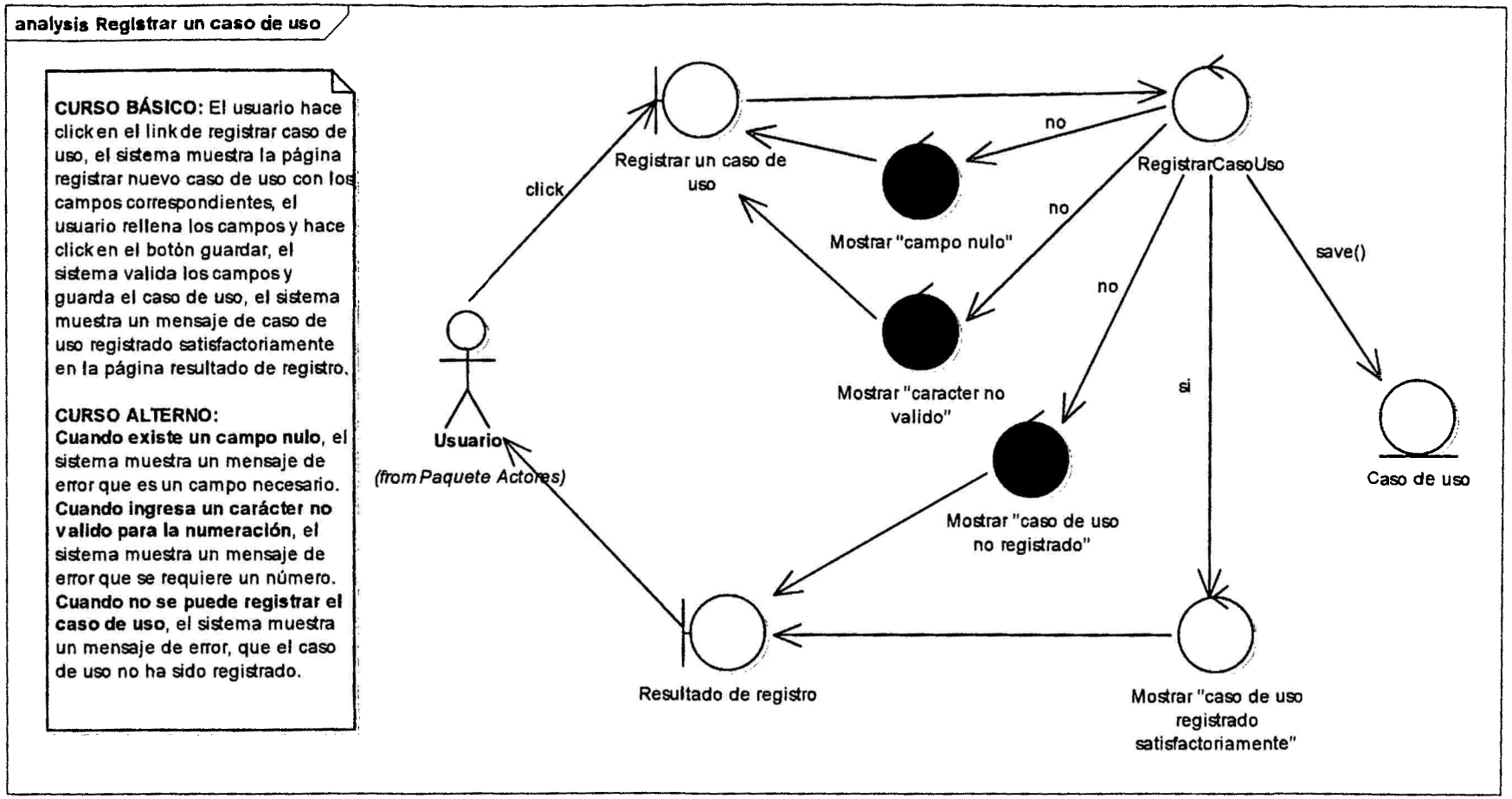


Figura N° 4.21: Diagrama de robustez registrar caso de uso revisado.

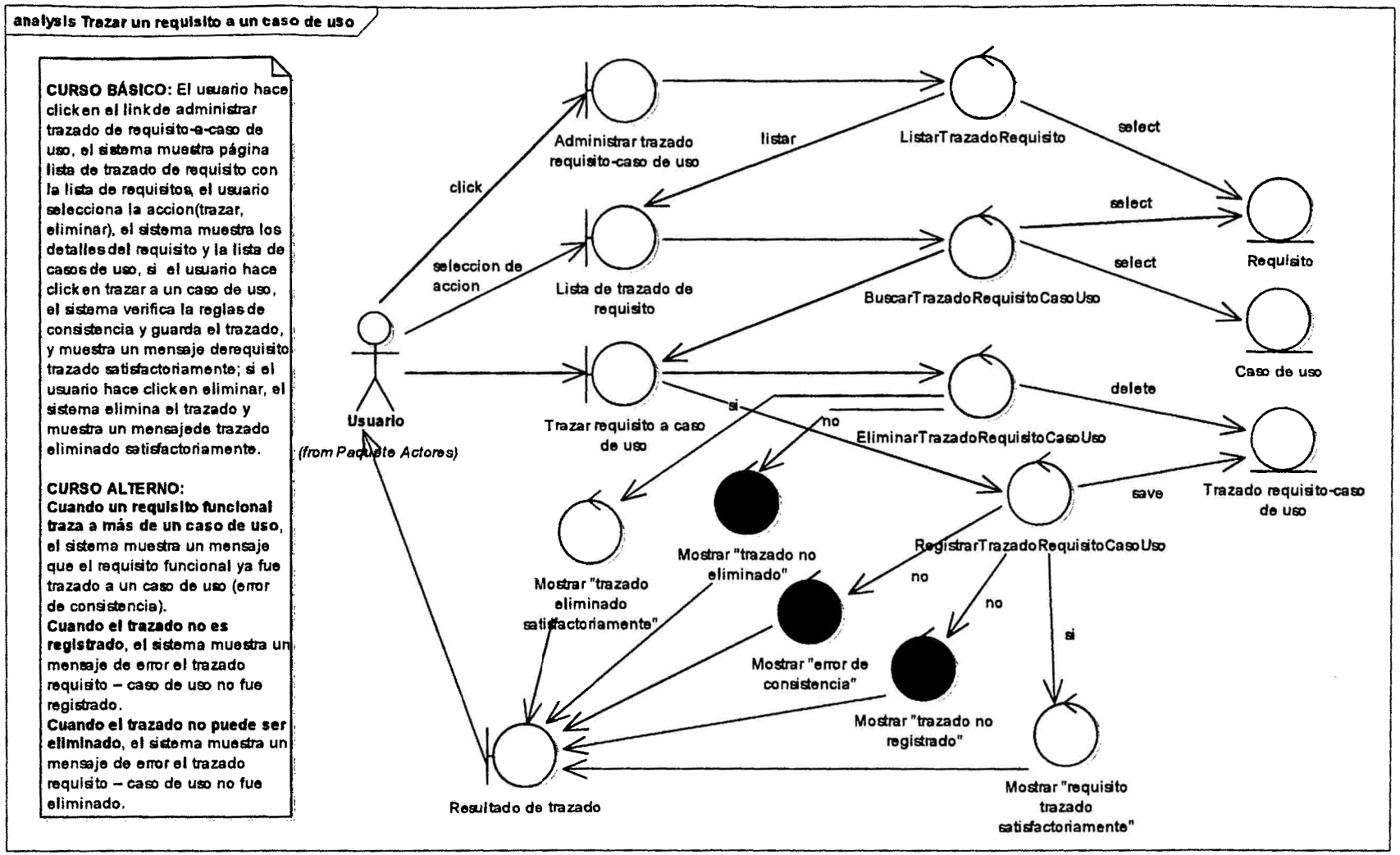


Figura N° 4.22: Diograma de robustez trazado requisito - caso de uso revisado.

REVISIÓN DEL MODELO DE DOMINIO ACTUALIZADO

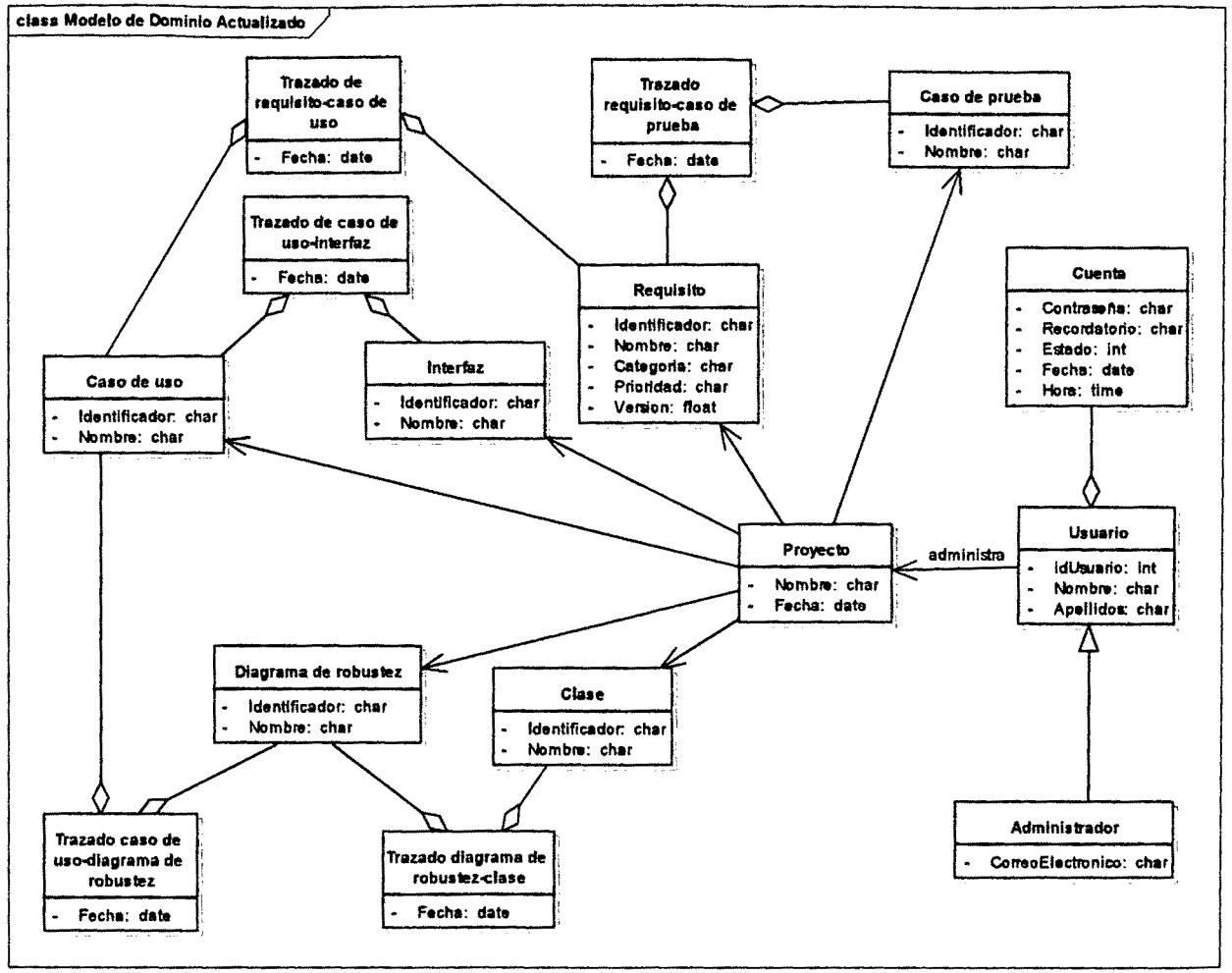


Figura N° 4.23: Modelo de dominio actualizado revisado.

D. DISEÑO

PARTE DEL MODELO DE DOMINIO ACTUALIZADO

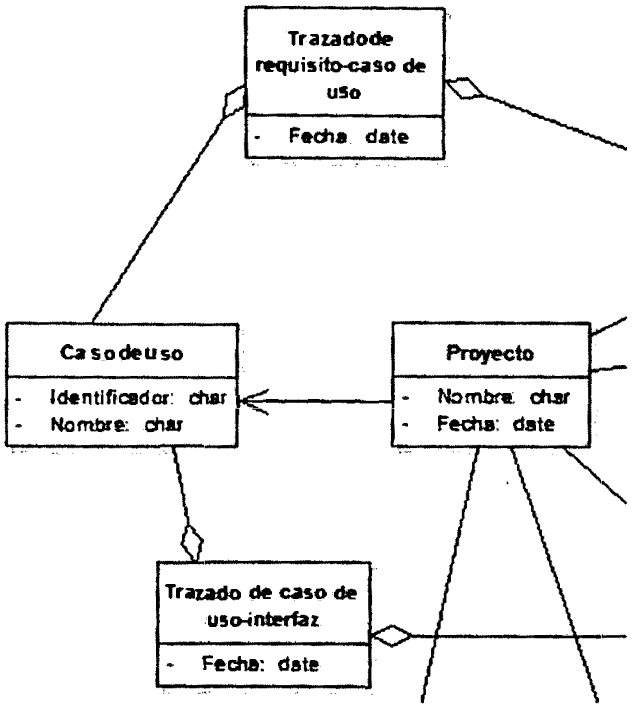


Figura Nº 4.24: Parte del modelo de dominio actualizado para objeto caso de uso.

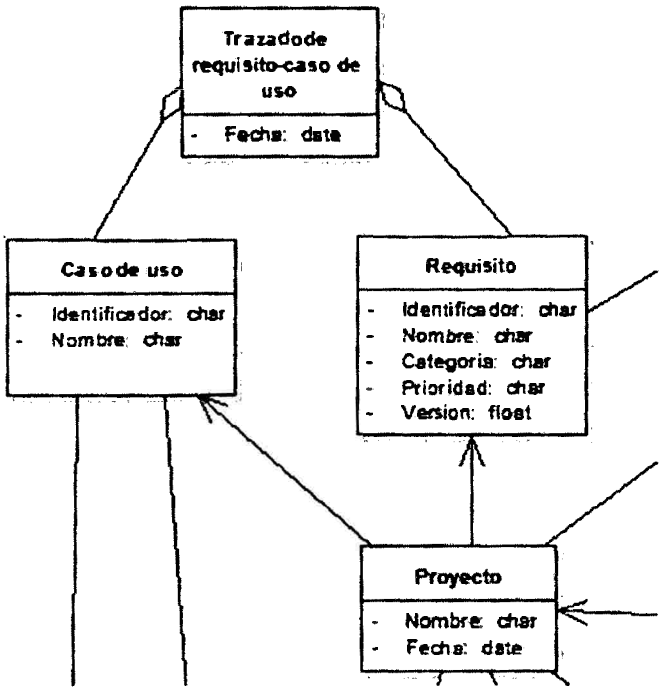


Figura Nº 4.25: Parte del modelo de dominio actualizado para objeto trazado requisito - caso de uso.

DIAGRAMA DE SECUENCIA

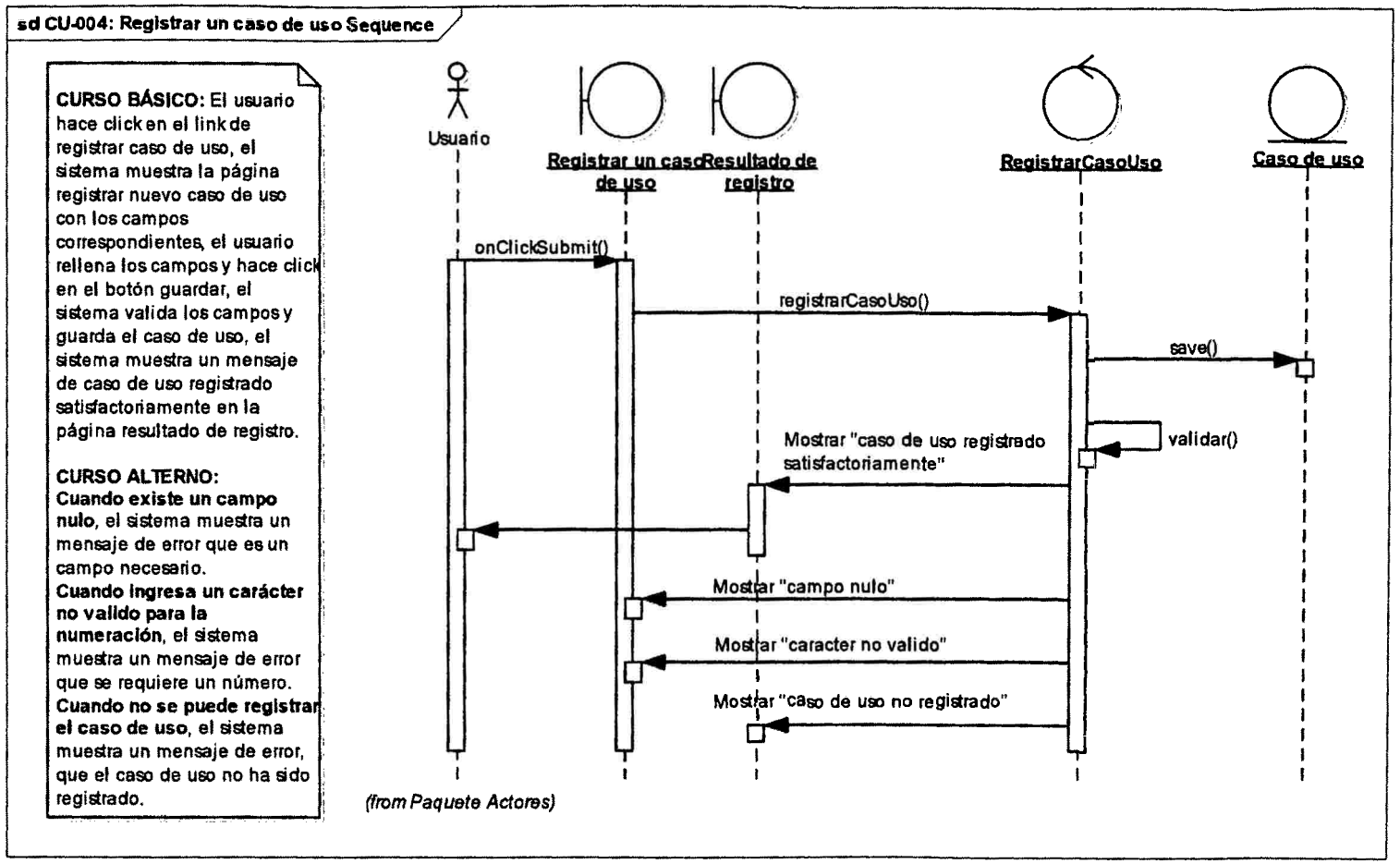


Figura N° 4.26: Diagrama de secuencia registrar caso de uso.

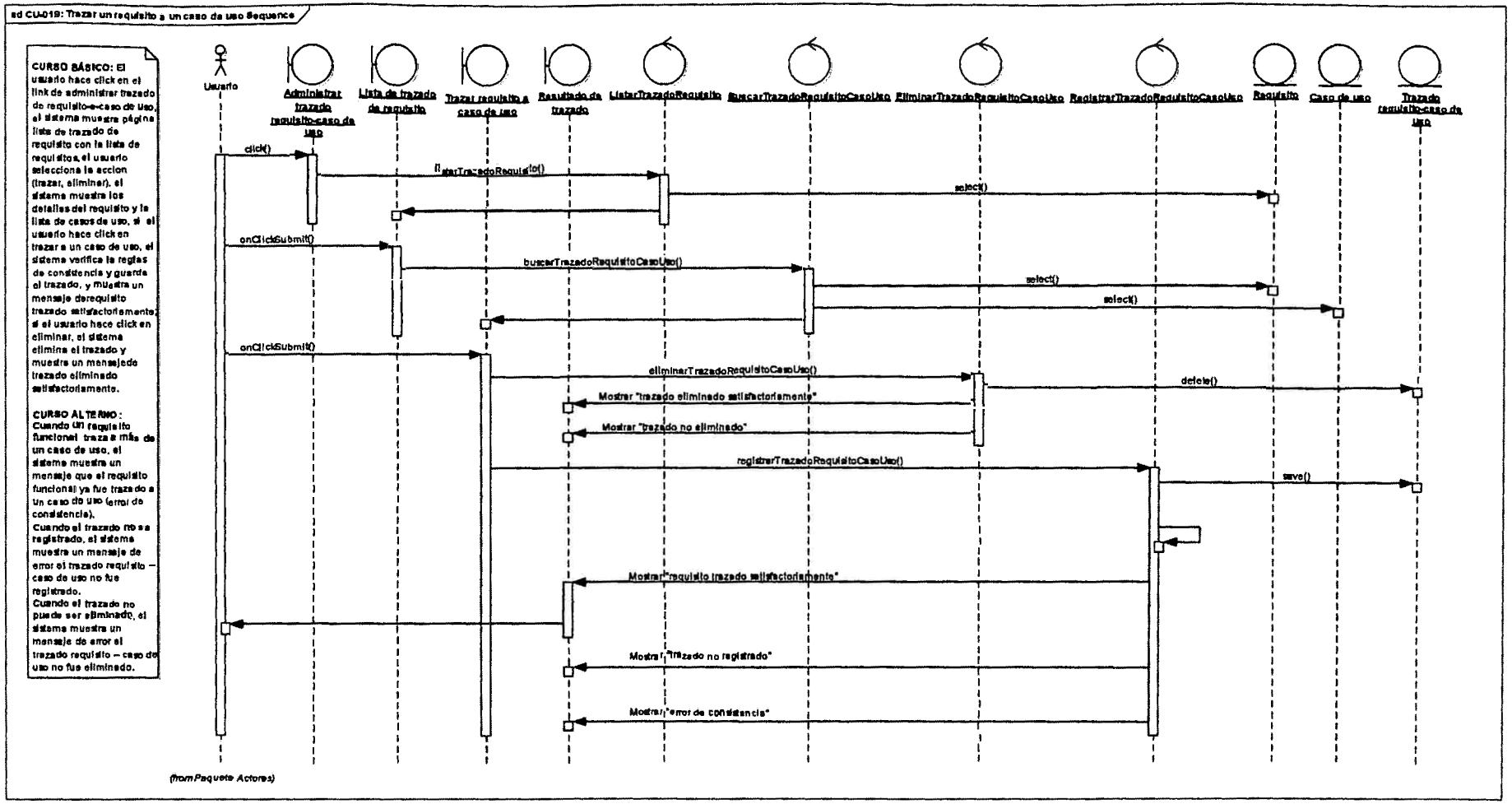


Figura N° 4.27: Diagrama de secuencia trazar requisito – caso de uso.

MODELO DE CLASES

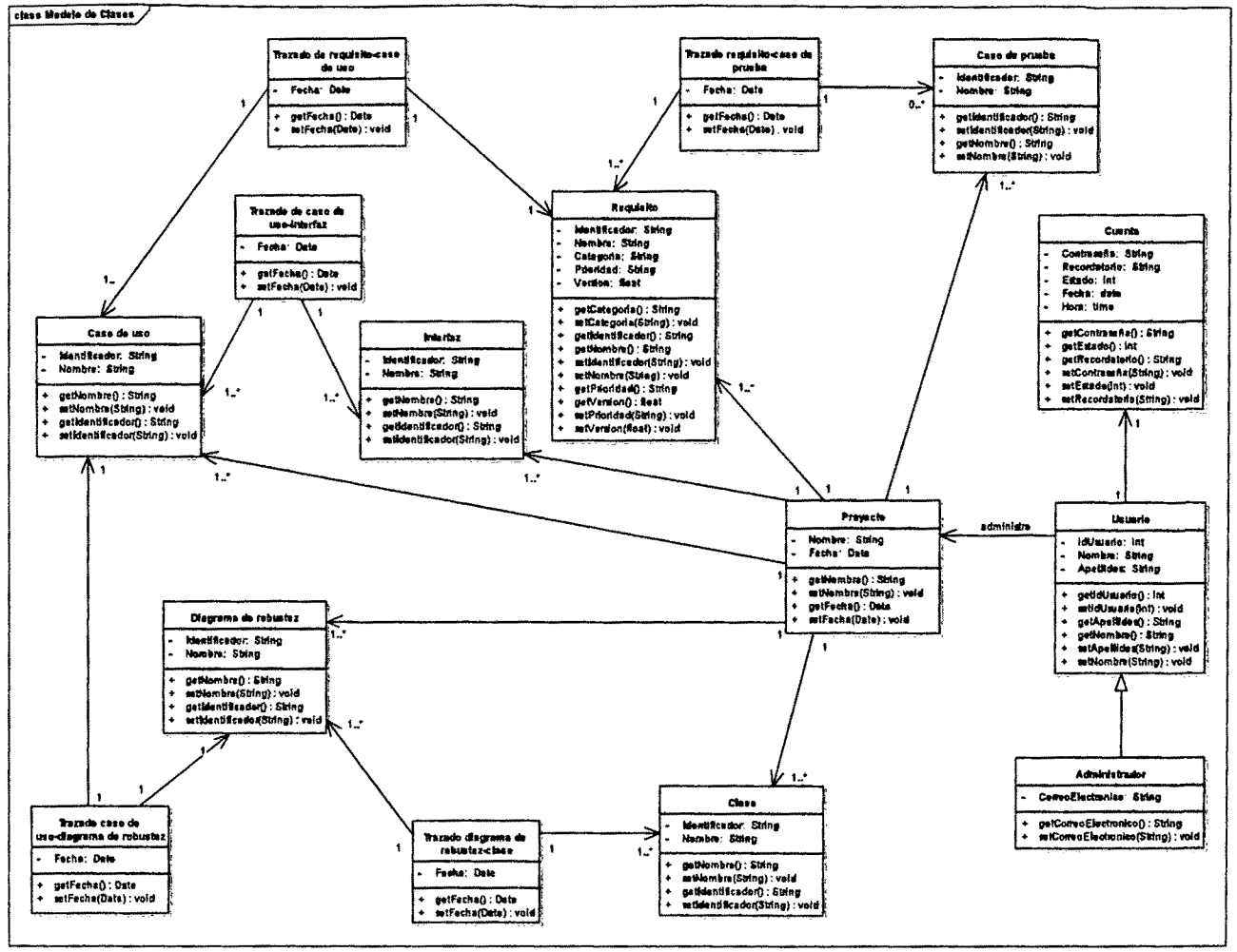


Figura N° 4.28: Modelo de clases.

LISTA DE CONTROLADORES

Controladores de la clase caso de uso
Vista onClickSubmit()
Control registrarCasoUso() validar() *save()*
Modelo getNombre() getIdentificador() setNombre() setIdentificador()

Tabla N° 4.5: Lista de controladores de la clase caso de uso.

Controladores de la clase trazado requisito - caso de uso
Vista onClickSubmit()
Control listarTrazadoRequisito() buscarTrazadoRequisitoCasoUso() eliminarTrazadoRequisitoCasoUso() registrarTrazadoRequisitoCasoUso() validar() select() delete() save()
Modelo getFecha() setFecha ()

Tabla N° 4.6: Lista de controladores de la clase trazado requisito - caso de uso.

HITO 3. REVISIÓN CRÍTICA DE DISEÑO

REVISIÓN DEL DIAGRAMA DE SECUENCIA

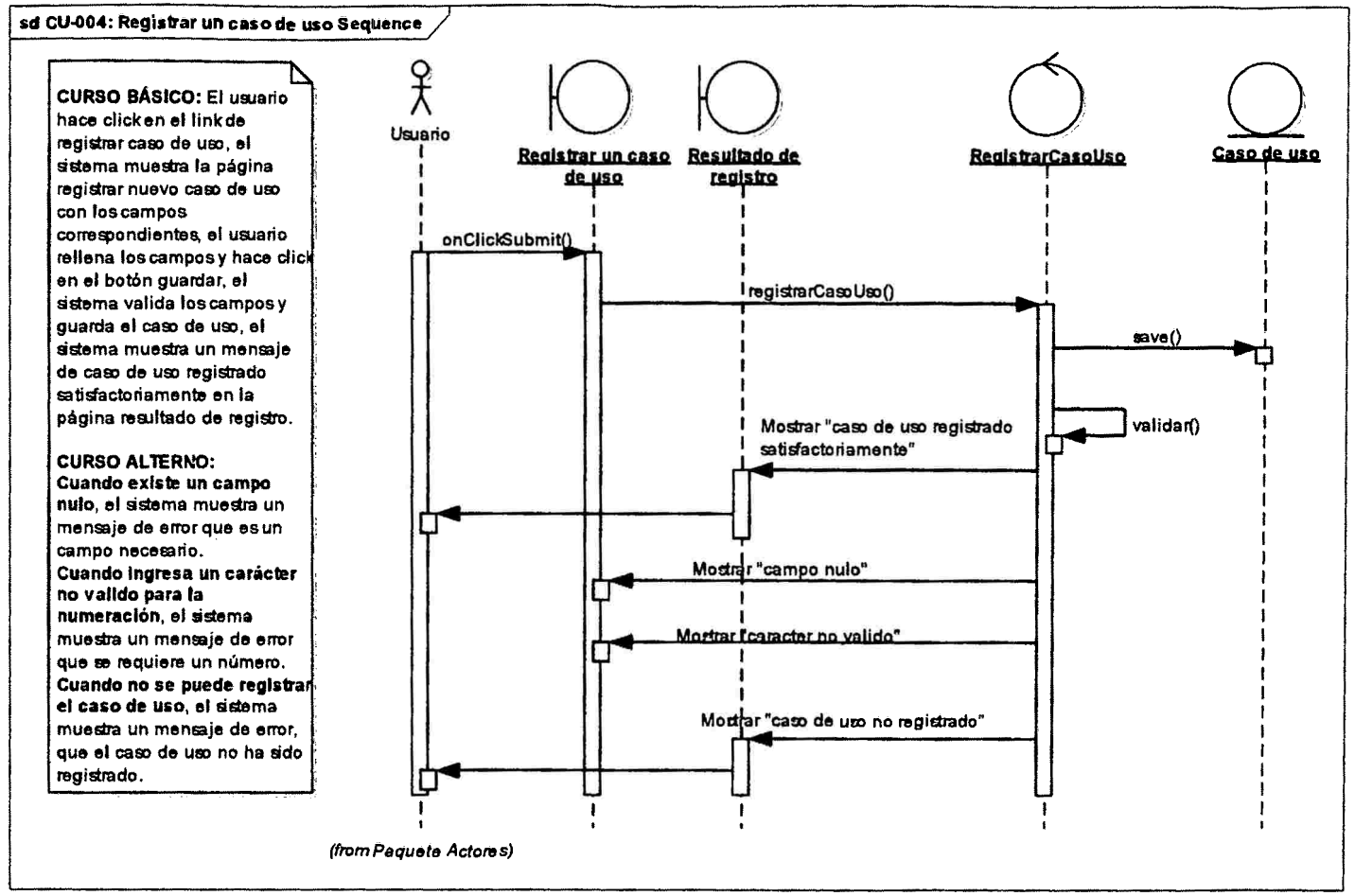


Figura N° 4.29: Diagrama de secuencia registrar caso de uso revisado.

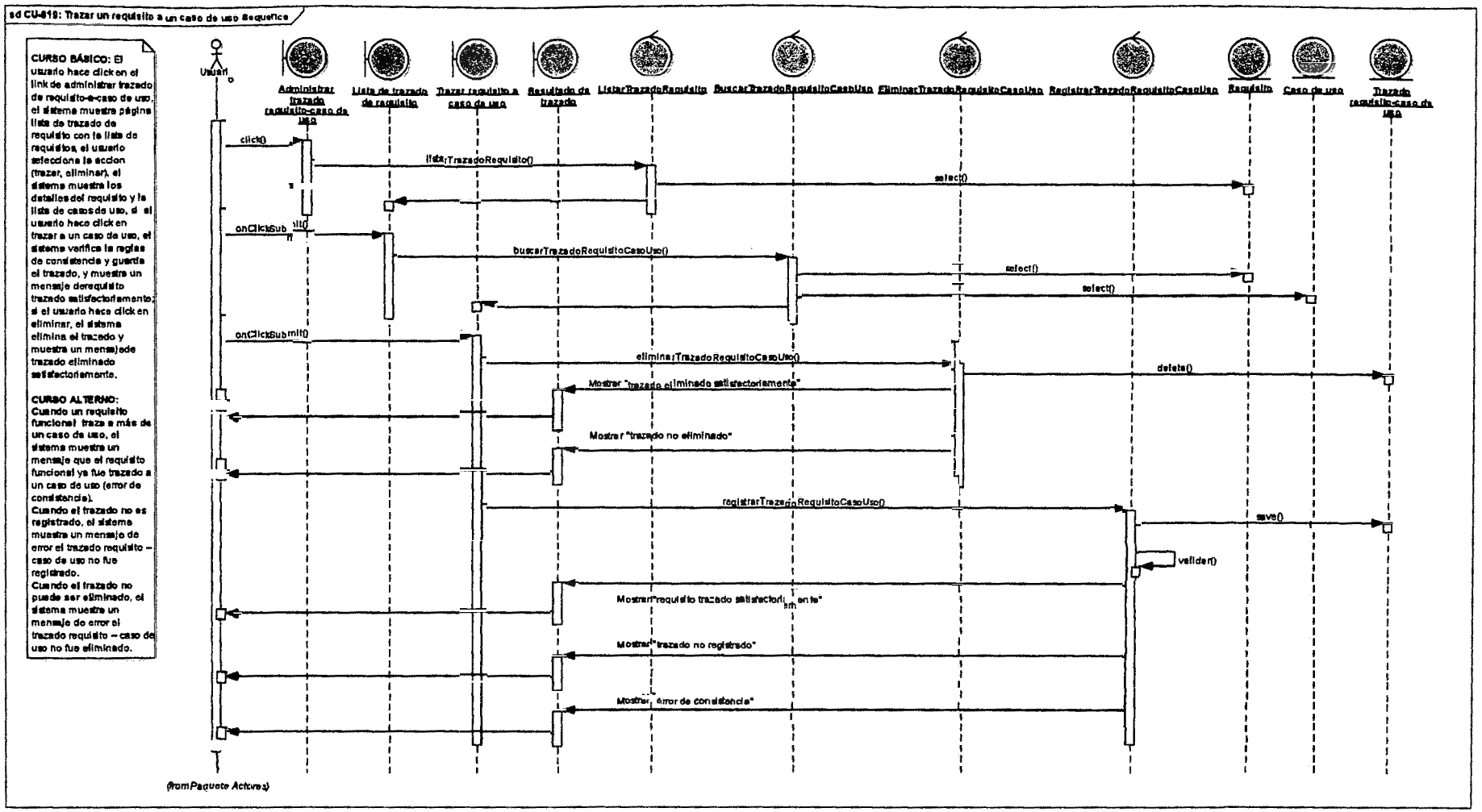


Figura N° 4.30: Diagrama de secuencia trazar requisito – caso de uso revisado.

REVISIÓN DEL MODELO DE CLASES

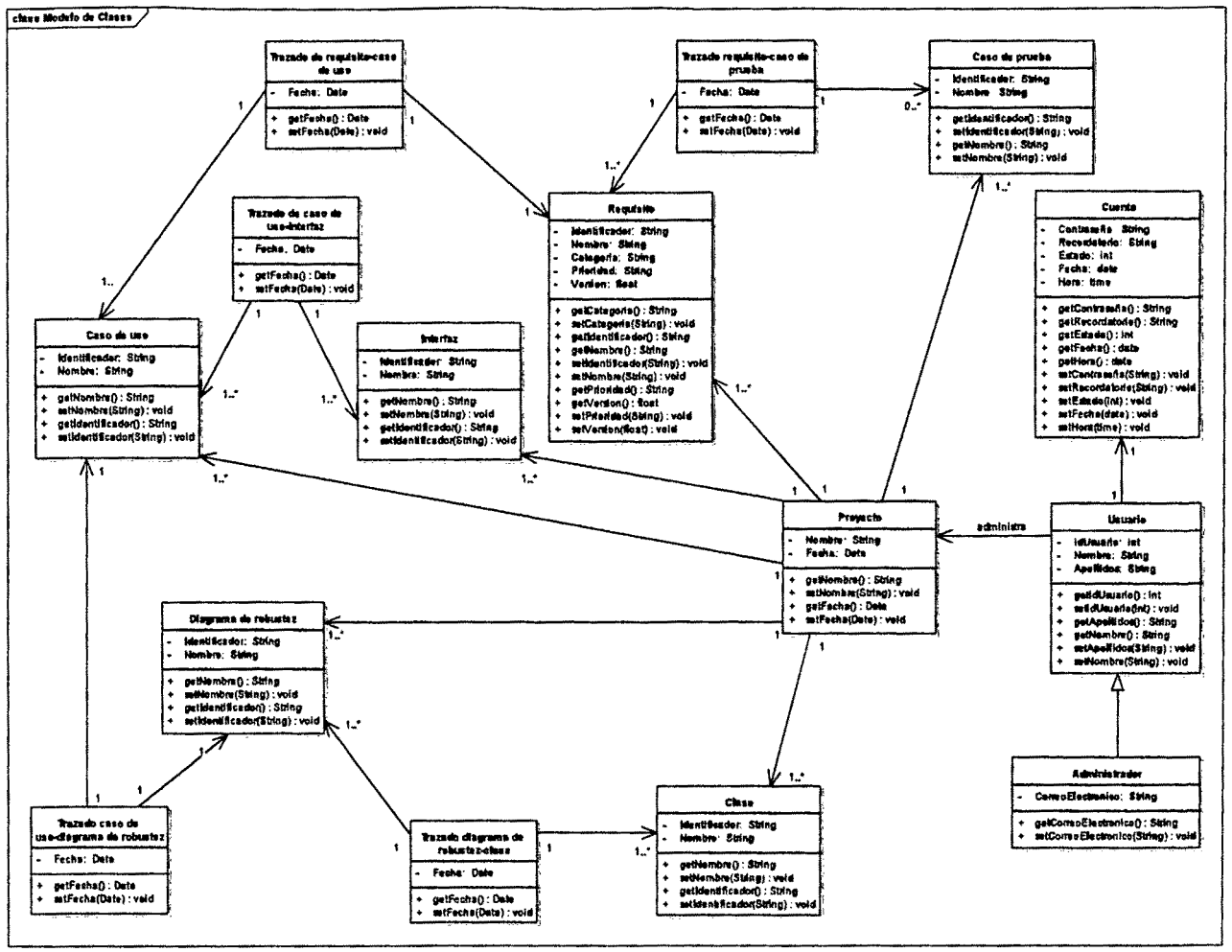


Figura Nº 4.31: Modelo de clases revisado.

REVISIÓN DEL MODELO DE DOMINIO ACTUALIZADO

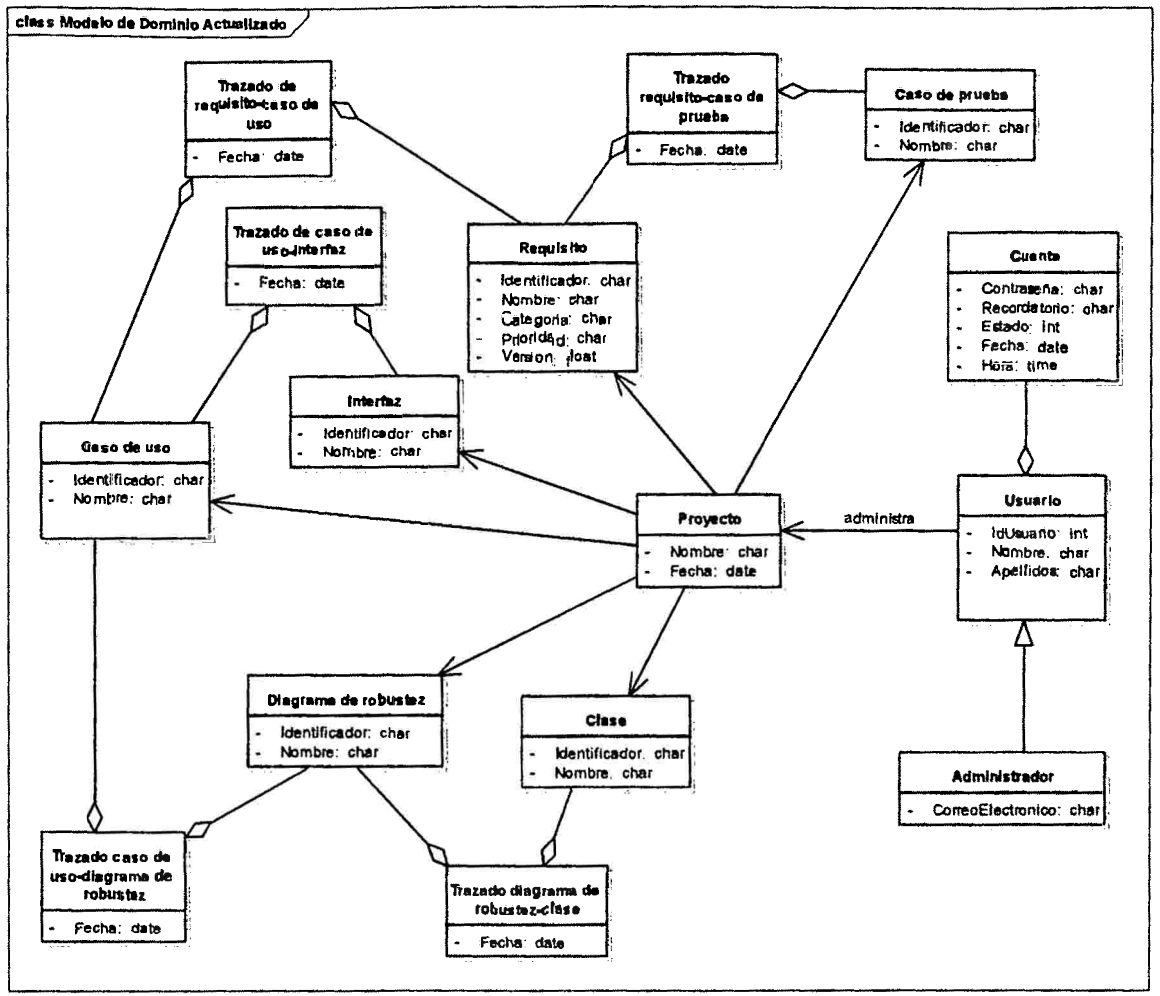


Figura N° 4.32: Modelo de dominio actualizado revisado.

REVISIÓN DE LA LISTA DE CONTROLADORES

Controladores de la clase caso de uso
Vista onClickSubmit()
Control registrarCasoUso(BCasouso bcasouso) validar() save(BCasouso bcasouso)
Modelo getNombre() getIdentificador() setNombre(String nombre) setIdentificador(String identificador)

Tabla N° 4.7: Lista de controladores de la clase caso de uso revisado.

Controladores de la clase trazado requisito - caso de uso
Vista onClickSubmit()
Control listarTrazadoRequisito(BProyecto bproyecto) buscarTrazadoRequisitoCasoUso(BTrazadoRequisitoCasouso btrcu) eliminarTrazadoRequisitoCasoUso(BCasouso bcasouso) registrarTrazadoRequisitoCasoUso(BCasouso bcasouso) validar() select() delete(BCasouso bcasouso) save(BCasouso bcasouso)
Modelo getFecha() setFecha (Date fecha)

Tabla N° 4.8: Lista de controladores de la clase trazado requisito - caso de uso revisado.

E. IMPLEMENTACIÓN

DEFINICIÓN DE LAS CLASES

La siguientes tablas (véase el código en el Anexo F – Código fuente) muestran clases para los casos de uso del sistema. A continuación se muestran las clases para los siguientes casos de uso:

a. CU-004. Registrar caso de uso

b. CU-019. Trazar un requisito a un caso de uso

CLASE CASO DE USO

Clase que representa un caso de uso, la caracterización de la clase puede observarse en las tablas siguientes y en la figura F.2.

Atributo	Tipo	Descripción
idCasouso	int	Atributo empleado para almacenar el id de caso de uso.
Identificador	String	Atributo empleado para almacenar el identificador del caso de uso.
Nombre	String	Atributo empleado para almacenar el nombre del caso de uso.
idProyecto	int	Atributo que permite indicar el id del proyecto al que pertenece el caso de uso.
Existe	int	Atributo que permite indicar si existe el caso de uso.
Resultado	int	Atributo que permite almacenar el resultado de una operación con los casos de uso.

Tabla N° 4.9: Atributos de la Clase BCasoUso.

Operación
public int **getExiste**()
public void **setExiste**(int Existe)
public String **getIdentificador**()
public void **setIdentificador**(String Identificador)
public String **getNombre**()
public void **setNombre**(String Nombre)
public int **getResultado**()
public void **setResultado**(int Resultado)
public int **getIdCasouso**()
public void **setIdCasouso**(int idCasouso)
public int **getIdProyecto**()
public void **setIdProyecto**(int idProyecto)

Tabla N° 4.10: Operaciones de la Clase BCasoUso

CLASE TRAZADO REQUISITO - CASO DE USO

Clase que representa el trazado requisito - caso de uso, la caracterización de la clase puede observarse en las tablas siguientes y en la figura F.3.

Atributo	Tipo	Descripción
idTrazadorequisitocasouso	int	Atributo empleado para almacenar el id del trazado de requisito - caso de uso.
fecha	String	Atributo empleado para almacenar la fecha del trazado de requisito - caso de uso.
idRequisito	int	Atributo que permite indicar el id del requisito que traza.
idCasouso	int	Atributo que permite indicar el id del caso de uso al que se traza el requisito.
Existe	int	Atributo que permite indicar si existe el trazado de requisito - caso de uso.
Resultado	int	Atributo que permite almacenar el resultado de una operación con la clase trazado de requisito - casos de uso.

Tabla N° 4.11: Atributos de la Clase BTrazadoRequisitoCasoUso.

Operación
public int **getExiste**()
public void **setExiste**(int Existe)
public int **getResultado**()
public void **setResultado**(int Resultado)
public String **getFecha**()
public void **setFecha**(String fecha)
public int **getIdCasouso**()
public void **setIdCasouso**(int idCasouso)
public int **getIdRequisito**()
public void **setIdRequisito**(int idRequisito)
public int **getIdTrazadorequisitocasouso**()
public void **setIdTrazadorequisitocasouso**(int idTrazadorequisitocasouso)

Tabla N° 4.12: Operaciones de la Clase BTrazadoRequisitoCasoUso.

SERVLET REGISTRAR CASO DE USO

Clase que representa el controlador para registrar un caso de uso (SRegistrarCasoUso), esta clase hereda de la clase HttpServlet (extends HttpServlet).

SERVLET REGISTRAR TRAZADO REQUISITO - CASO DE USO

Clase que representa el controlador para registrar un caso de uso (SRegistrarTrazadoRequisitoCasoUso), esta clase hereda de la clase HttpServlet (extends HttpServlet).

Operación
protected void **processRequest**(HttpServletRequest request, HttpServletResponse response)
@Override protected void **doGet**(HttpServletRequest request, HttpServletResponse response)
@Override protected void **doPost**(HttpServletRequest request, HttpServletResponse response)
@Override public String **getServletInfo**()

Tabla N° 4.13: Operaciones de las Clases heredadas de HttpServlet.

DATA ACCESS OBJECT CASO DE USO

Atributo	Tipo	Descripción
connection	Connection	Atributo empleado para almacenar un objeto conexión.

Tabla N° 4.14: Atributos de la Clase DaoCasoUso.

Operación
public **DaoCasoUso**(Connection connection)
public void **guardarCasoUso**(BCasoUso bCasoUso)

Tabla N° 4.15: Operaciones de la Clase DaoCasoUso.

DATA ACCESS OBJECT TRAZADO REQUISITO - CASO DE USO

Atributo	Tipo	Descripción
connection	Connection	Atributo empleado para almacenar un objeto conexión.

Tabla N° 4.16: Atributos de la Clase DaoTrazadoRequisitoCasoUso.

Operación
public **DaoTrazadoRequisitoCasoUso** (Connection connection)
public void **guardarTrazadoRequisitoCasoUso**(BTrazadoRequisitoCasoUso btrcu)
public void **listarTrazadoCasoUso**(AListaArray aListaArray, int idproyecto)

Tabla N° 4.17: Operaciones de la Clase DaoTrazadoRequisitoCasoUso.

DISEÑO DE LAS INTERFACES

Las siguientes figuras muestran las interfaces diseñadas para el sistema.

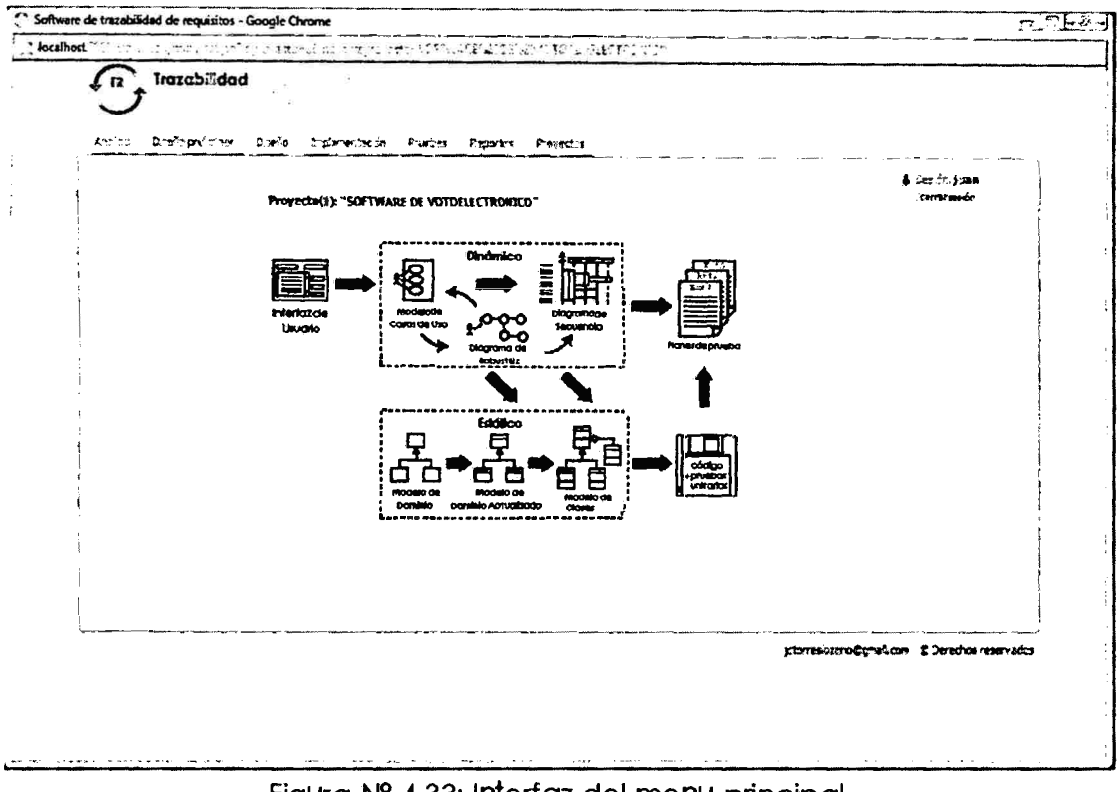


Figura N° 4.33: Interfaz del menu principal.

INTERFAZ DE REGISTRO DE UN CASO DE USO

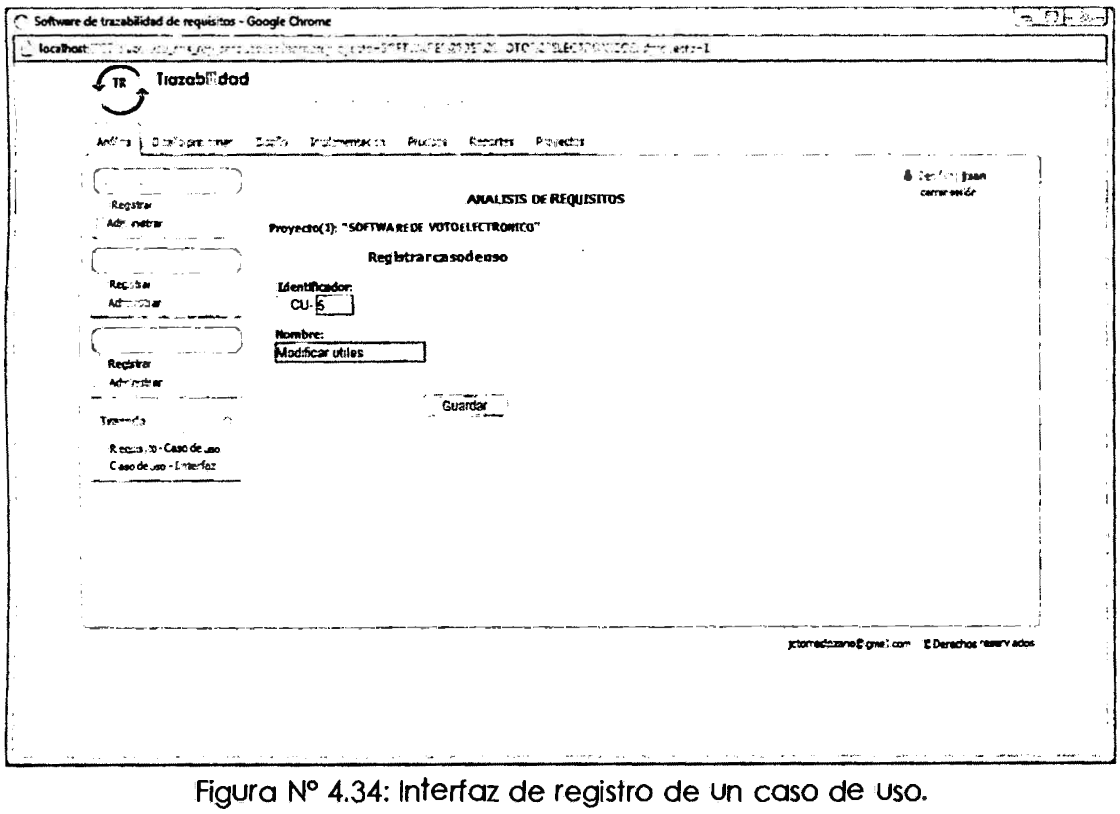


Figura N° 4.34: Interfaz de registro de un caso de uso.

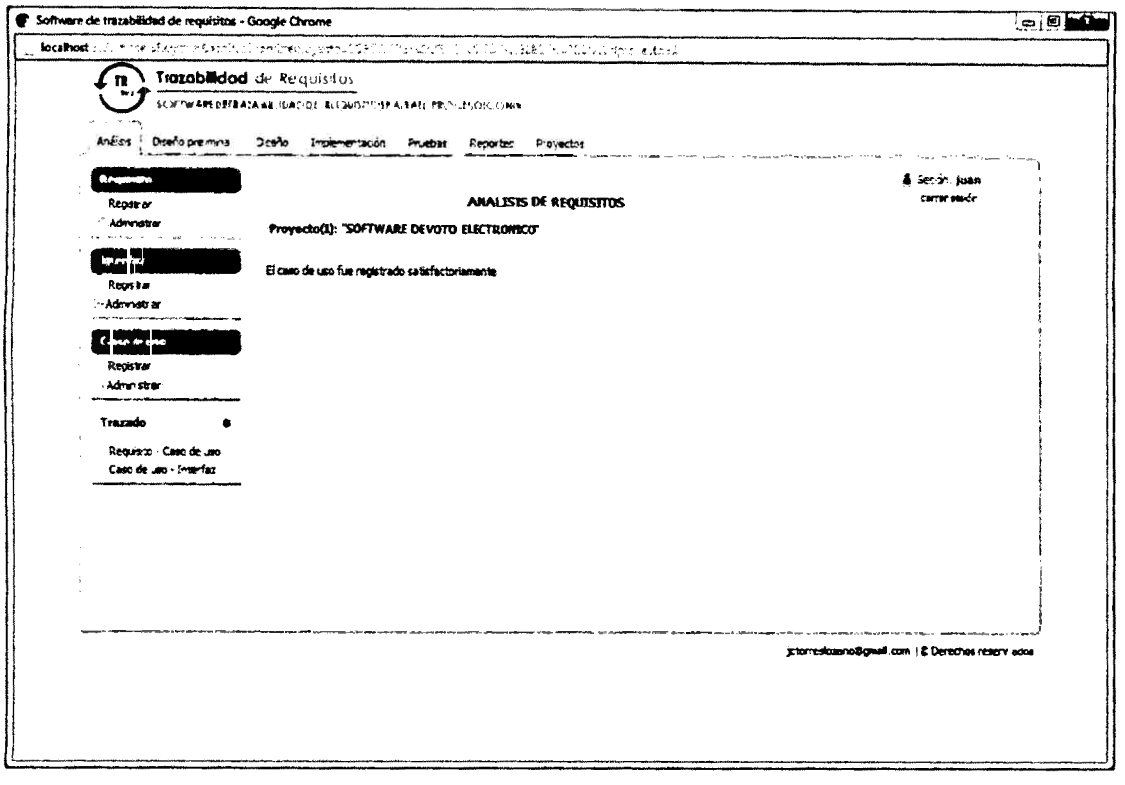


Figura N° 4.35: Interfaz de resultado de para el registro de un caso de uso.

INTERFAZ DE TRAZADO DE UN REQUISITO A UN CASO DE USO

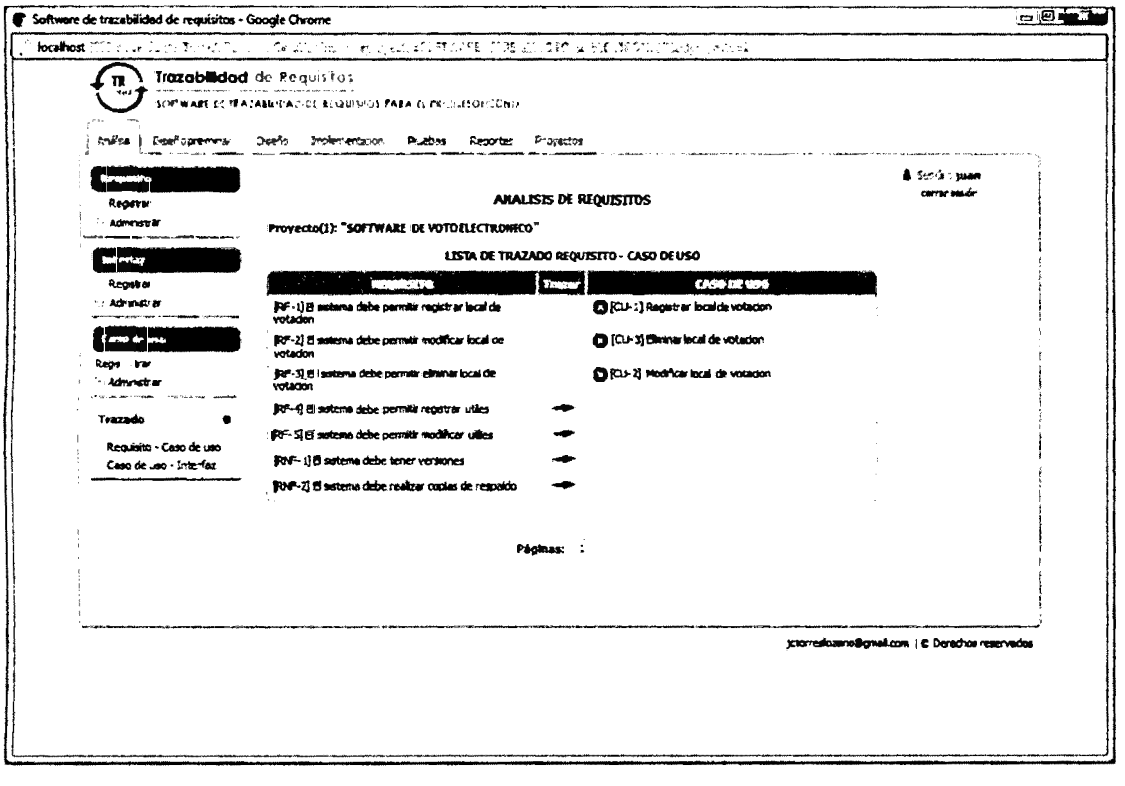


Figura N° 4.36: Interfaz de la lista de trazado de requisito - caso de uso.

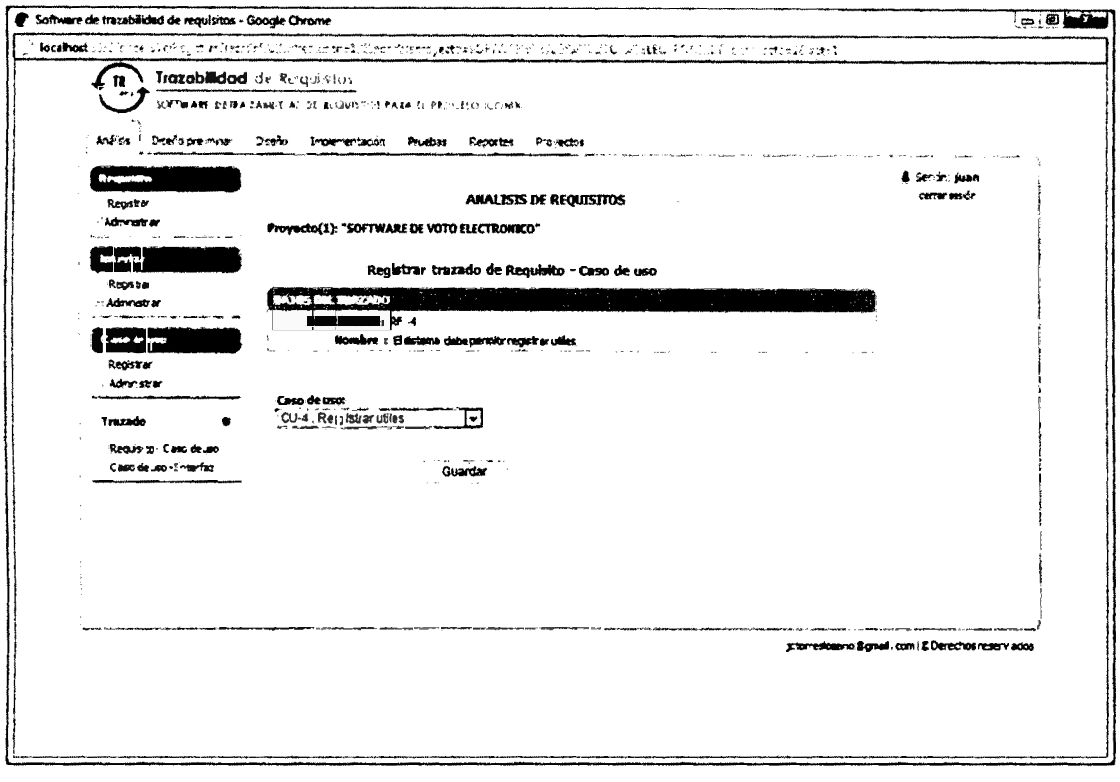


Figura N° 4.37: Interfaz de registro de trazado de un requisito a un caso de uso.

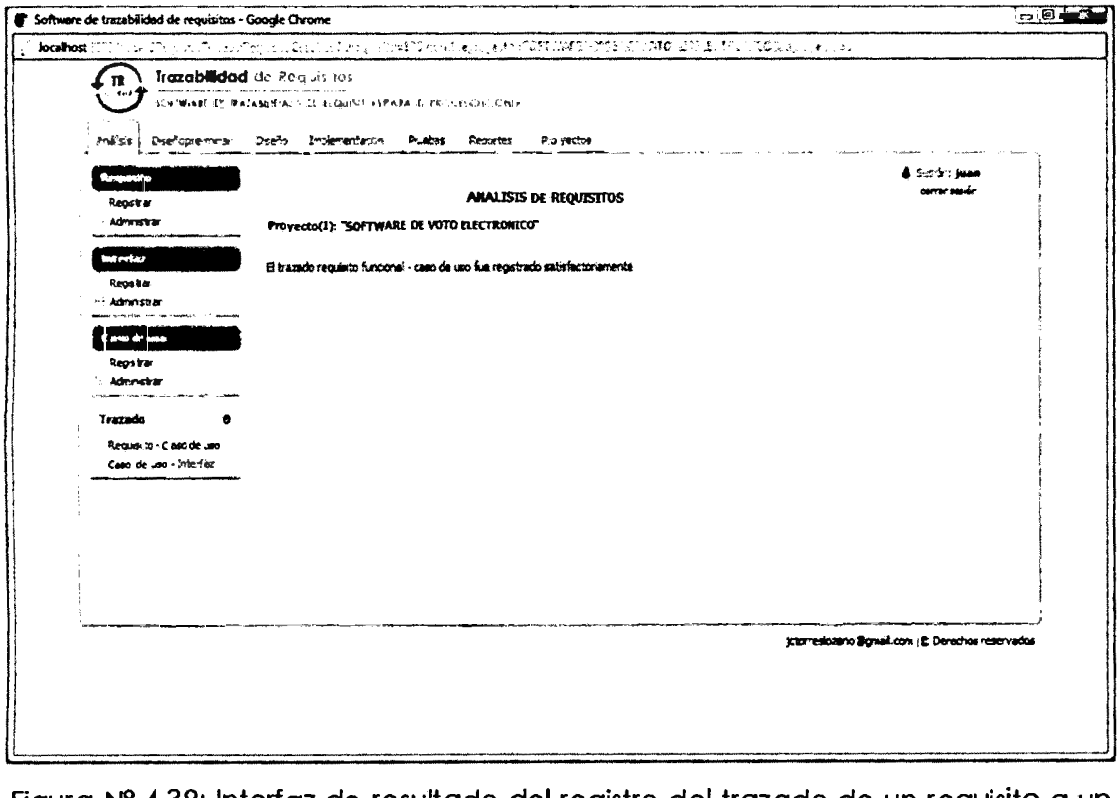


Figura N° 4.38: Interfaz de resultado del registro del trazado de un requisito a un caso de uso.

REPORTE DE PRUEBAS UNITARIAS

Las pruebas unitarias se realizaron a los controladores del modelo MVC, en este caso a las clases Servlet; habiendo sido testeados satisfactoriamente.

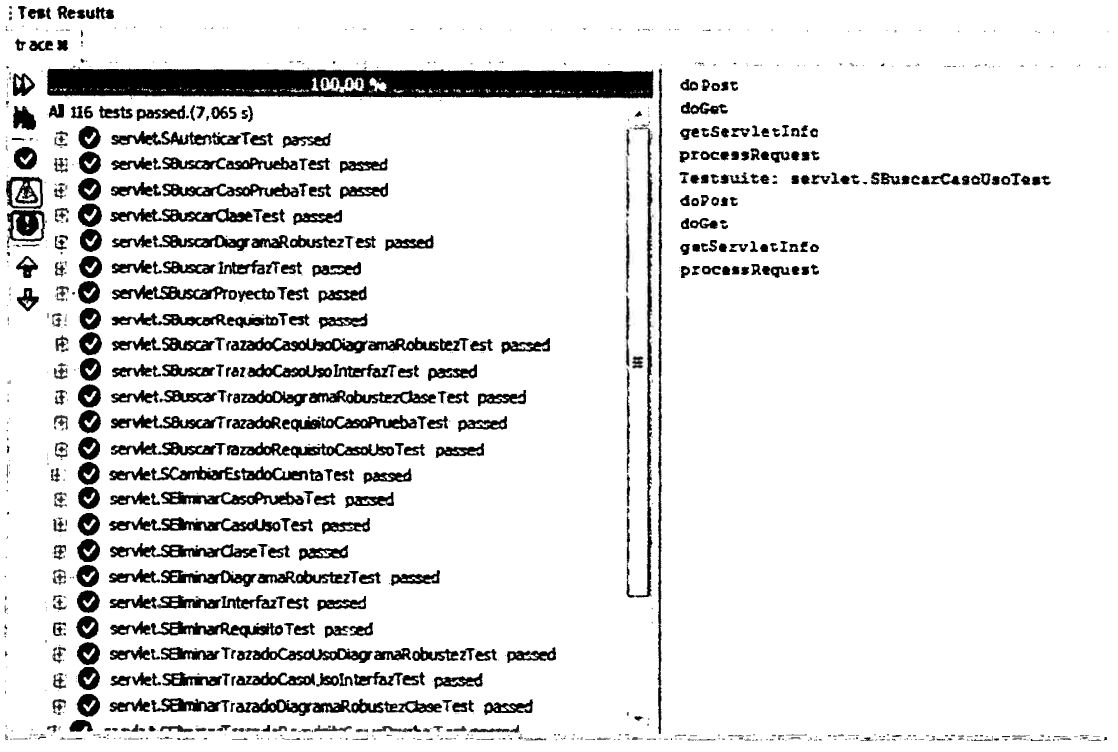


Figura N° 4.39: Resultado de pruebas unitarias

REPORTE DE PRUEBAS DE ACEPTACIÓN

Para realizar las pruebas de aceptación basadas en casos de uso tal como lo recomienda ICONIX, se probó todos los cursos básicos y alternos del caso de uso descritos en los casos de prueba. Se hace la prueba de aceptación para los casos de uso con los casos de prueba correspondientes.

PRUEBA DE ACEPTACIÓN PARA REGISTRAR UN CASO DE USO

Casos de prueba		**Criterio**	**Paso**	**Fallo**
CP-015	Registrar un caso de uso	Pasó	1	0
CP-016	Mostrar error caso de uso no registrado	Pasó	1	0
CP-017	Mostrar caso de uso registrado satisfactoriamente	Pasó	1	0
CP-001	Mostrar error de campo nulo	Pasó	1	0
			4	0

PRUEBA DE ACEPTACIÓN PARA TRAZAR REQUISITO - CASO DE USO

Casos de prueba		**Criterio**	**Paso**	**Fallo**
CP-002	Mostrar error de consistencia	Pasó	1	0
CP-060	Listar trazado requisito	Pasó	1	0
CP-062	Buscar trazado requisito - caso de uso	Pasó	1	0
CP-063	Mostrar requisito - caso de uso trazado satisfactoriamente	Pasó	1	0
CP-065	Registrar trazado requisito - caso de uso	Pasó	1	0
			5	0

HITO 4. REVISIÓN DE LA IMPLEMENTACIÓN (ENTREGABLE)

REVISIÓN DEL MODELO DE DOMINIO ACTUALIZADO

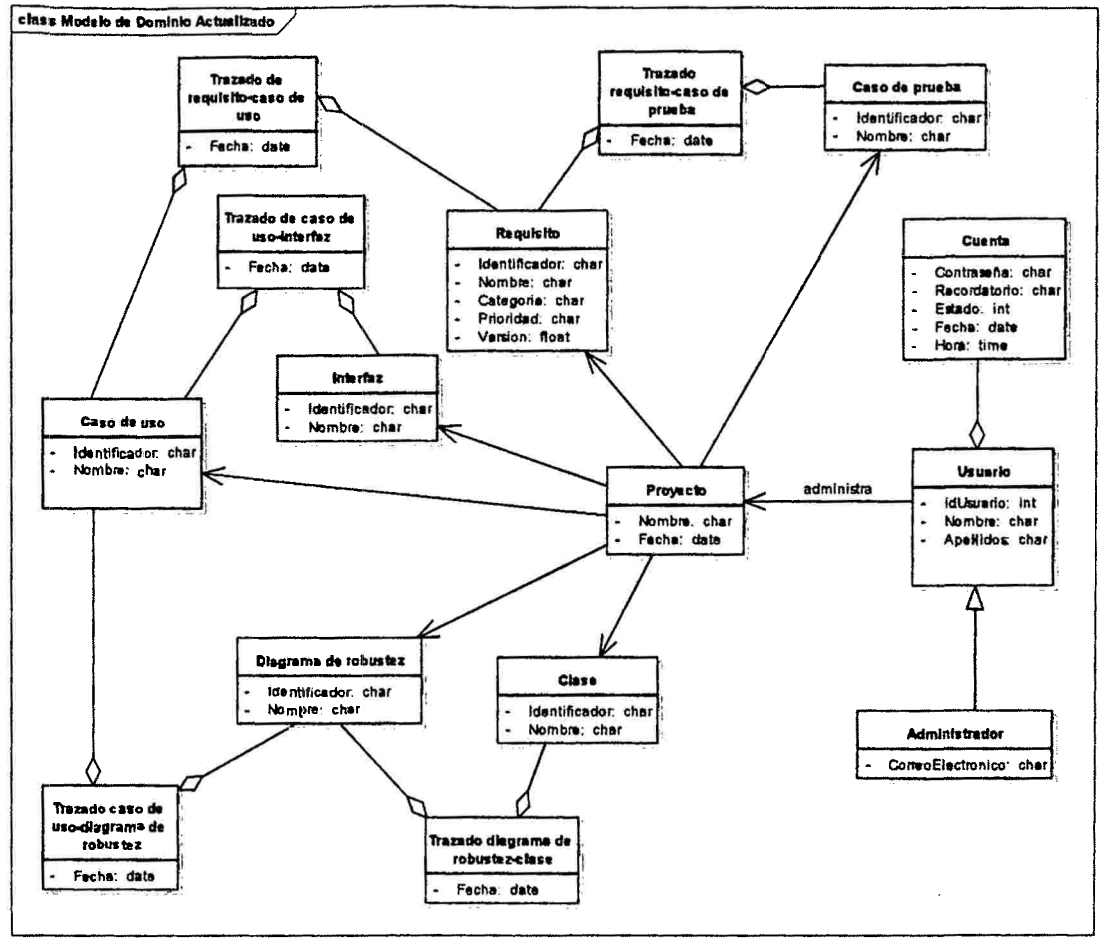


Figura N° 4.40: Modelo de dominio actualizado(Entregable).

CAPITULO V
CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

a. Para la fase de análisis se identificó como artefactos trazables; los requisitos, casos de uso e interfaces que se muestra en la tabla N° 4.1, los requisitos para la trazabilidad se muestran en la figura N° 4.2 y los casos de uso en la Figura N° 4.8, que luego se diseñó, desarrolló y se probó en las siguientes fases, y que ayudan a refinar los requisitos y el modelo de dominio.

b. Para la fase de diseño preliminar se identificó como artefacto trazable los diagramas de robustez que se muestra en la tabla N° 4.1, los requisitos para la trazabilidad se muestran en la figura N° 4.2 y los casos de uso en la Figura N° 4.9, que luego se diseñó, desarrolló y se probó en las siguientes fases, y que ayudan a mejorar los diagramas de robustez, actualizar el modelo de dominio y el borrador de casos de uso.

c. Para la fase de diseño se identificó como artefacto trazable las clases que se muestra en la tabla N° 4.1, los requisitos para la trazabilidad se muestran en la figura N° 4.2 y los casos de uso en la Figura N° 4.10, que luego se diseñó, desarrolló y se probó en las siguientes fases, y que ayudan a actualizar el modelo de dominio, realizar los diagramas de secuencia, limpiar el modelo de clases e identificar los controladores.

d. Para la fase de implementación no se ha encontrado artefactos trazables que se muestra en la tabla N° 4.1, por lo cual para mejorar el modelo de dominio, escribir el código y las pruebas unitarias se utilizan la fase de diseño.

e. Para la fase de pruebas se identificó como artefacto trazable los casos de prueba que se muestra en la tabla N° 4.1, los requisitos para la trazabilidad se muestran en la figura N° 4.2 y los casos de uso en la Figura N° 4.11, que luego se diseñó, desarrolló y se probó en las siguientes fases, y que ayudan a escribir las pruebas y direccionar los requisitos a los casos de prueba.

f. Finalmente se desarrolló un prototipo del software de trazabilidad de requisitos para el proceso ICONIX, esta herramienta utiliza técnicas de trazabilidad apoya la gestión de la transformación y la evolución de los requisitos en el desarrollo de software con el proceso ICONIX.

5.2. RECOMENDACIONES

a. Para futuras líneas de investigación se recomienda incorporar un analizador de completitud semántico con selección automática del sentido del verbo de acuerdo con el contexto o dominio del requisito.

b. Para el software de trazabilidad requisito desarrollado se recomienda la ampliación del sistema para ser una ayuda y tutor en las fases del modelado con el proceso ICONIX.

c. Se recomienda la adaptación de la herramienta para un entorno de desarrollo de software que utilice como metodología en proceso ICONIX.

d. La trazabilidad de requisitos es una práctica que puede ser utilizada en otras metodologías de desarrollo.

BIBLIOGRAFÍA

1. Abián, M. Á. (2006). *Orientación a objetos: conceptos ,terminología y lenguajes.* http://www.javahispano.org.
2. Abián, M. Á. (Diciembre de 2003). Promesas incumplidas: sobre los métodos ágiles. Obtenido de javahispano: http://www.javahispano.org
3. Alonso, F., Martínez, L., & Segovia, F. J. (2005). Introducción a la ingenieria del software. Zaragoza: Delta publicaciones.
4. Anaya, V., & Letelier, P. (2002). Trazabilidad de requisitos adaptada a las necesidades del proyecto: un caso de estudio usando alternativamente RUP y XP. Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación, Valencia.
5. Aprende UML. (12 de Julio de 2012). Recuperado el 9 de Septiembre de 2012, de Aprende UML: www.aprendeuml.com.
6. Aumaille, B. (2002). J2EE desarrollo de aplicaciones web. Barcelona: Editions ENI.
7. Bernal Torres, C. A. (2006). Metodología de la investigación. Mexico: Pearson.
8. Blanco, M. L. (2007). Metodologías de desarrollo de software aplicadas a SOA. Buenos Aires: Universidad Nacional de Luján.
9. Camps, Casillas, Costal, Gibert, Martín, & Pérez. (2005). Base de datos.
10. Carrasco Díaz, S. (2006). Metodología de la investigación científica. Lima: Editorial San Marcos.
11. De Pablos, C., López, J. J., Romero, S. M.-R., & Medina, S. (2004). Informática y comunicaciones en la emprèsa. Madrid: Esic editorial.
12. Garcia, E., & Roque, C. (2007). Principios básicos de informática. Madrid: Dykinson.
13. Gomez, J. P. (2006). Ingeniería de Requerimientos. Bogotá: Universidad Tecnológica de Pereira.
14. Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2006). Metodología de la Investigación. México: McGraw Hill.

15. INTECO. (2008). Guía práctica de gestión de requisitos. Instituto Nacional de Tecnologías de la Comunicación , Madrid.
16. León, G. (1996). Ingeniería de sistemas de software (Primera ed.). Madrid: Isdefe.
17. Londoño, Anaya, & Tabares. (2008). Análisis de la ingeniería de requisitos orientada por aspectos según la industria del software. Revista EIA , 43-52.
18. Maniasi, S. D. (2005). Identificación de riesgos de proyectos de software en base a taxonomías. Córdoba.
19. Navarro, J., Orozco, R., & Jaramillo, A. F. (2005). Analizador de completitud de requisitos escritos en español restringido. Revista Avances en Sistemas e Informática, 1 (2), 19-25.
20. Porras, E. E. (2011). La metodología ágil y formal ICONIX para el desarrollo de software: Teoría y Práctica. Ayacucho.
21. Pressman, R. S. (2005). Ingeniería del Software. McGraw-Hill.
22. Rodríguez, J. J., Santamaría, L., Rabasa, A., & Martínez, O. (2003). Introducción a la programación: teoría y práctica. Alicante: Club Universitario.
23. Rosenberg, D., & Stephens, M. (2007). Casos de uso guiados por modelado de objetos con UML. Nueva York: Apress.
24. Schenone, M. H. (2004). Diseño de una metodología ágil de desarrollo de software. Buenos Aires: Fiuba.
25. Sommerville, I. (2005). Ingeniería del Software. Madrid: Pearson.
26. Tabares, M. S. (2009). Un patrón de trazabilidad para controlar la evolución de los intereses en un espacio multidimensional. Medellín.
27. Tabares, M.S., Arango, F., & Anaya, R. (2006). Una revisión de modelos y semánticas para la trazabilidad de requisitos. Revista EIA (6), 33-42.
28. Tabares, M. S., Barrera, A. F., Arroyave, J. D., & Pineda, J. D. (2007). Un método para la trazabilidad de requisitos en el proceso unificado de desarrollo. Revista EIA (8), 69-82.
29. Zapata, C. M., Villegas, S.M., & Arango, F. (2006). Reglas de consistencia entre modelos de requisitos de un método. REVISTA Universidad EAFIT, 42 (141), 40-59.

ANEXOS

ANEXO A. FICHA DE ANÁLISIS DOCUMENTAL DE LA EVOLUCIÓN DE LOS ARTEFACTOS DEL PROCESO ICONIX

ARTEFACTO	FASES DEL PROCESO ICONIX				
	Análisis	**Diseño preliminar**	**Diseño**	**Implementación**	**Pruebas**
Requisitos funcionales y no funcionales	Crea				
Casos de prueba	Crea				Modifica
Glosario de objetos	Crea				
Modelo de dominio	Crea Revisa	Modifica Revisa	Revisa	Revisa	
Prototipo GUI	Crea Revisa				
Lista de casos de uso	Crea				
Diagrama de casos de uso	Crea	Modifica Revisa			
Paquete de casos de uso	Crea				
Relación entre requisitos funcionales y casos de uso	Crea				
Primer borrador de casos de uso	Crea Revisa	Modifica Revisa			
Diagrama de robustez		Crea Revisa			
Parte de modelo de dominio actualizado			Crea		
Diagrama de secuencia			Crea Revisa		
Diagrama de clases			Crea Revisa		
Lista de controladores			Crea Revisa		

Código fuente				Crea Revisa	
Prueba de sistema				Crea	
Prueba de aceptación				Crea	
Plan de prueba					Crea

ANEXO B. FICHA DE ANÁLISIS DOCUMENTAL DE LA TRAZABILIDAD PARA EL PROCESO ICONIX

FASE	TAREA	ARTEFACTO	ARTEFACTO TRAZABLE (SI/NO)	NIVEL DE TRAZABILIDAD
ANÁLISIS	4. Identificar requisitos	Requisitos funcionales y no funcionales	SI	ALTA
		Casos de prueba	NO	
	5. Identificar objetos del mundo real y dibujar el modelo de dominio	Glosario de objetos	NO	
		Modelo de dominio	NO	
	6. Realizar prototipo de interfaz grafica	Prototipo GUI	SI	BAJA
	7. Descubrir casos de uso	Lista de casos de uso	SI	ALTA
	8. Dibujar y empaquetar los casos de uso	Diagrama de casos de uso	NO	
		Paquete de casos de uso	NO	
	9. Asigne requisitos funcionales a los casos de uso	Relación entre requisitos funcionales y casos de uso	NO	
	10. Escribir el primer barrador de casos de uso	Primer barrador de casos de uso	NO	
	1. Revisor el modelo de dominio	Modelo de dominio	NO	
	2. Revisar el prototipo GUI	Prototipo GUI	NO	
	3. Revisar modelo de casos de uso	Casos de uso revisado	NO	
DISEÑO PRELIMINAR	5. Reescriba el primer borradar para cada caso de uso	Caso de usa desambiguado	NO	
	6. Identificar el primer corte de	Diagrama de robustez	SI	ALTA

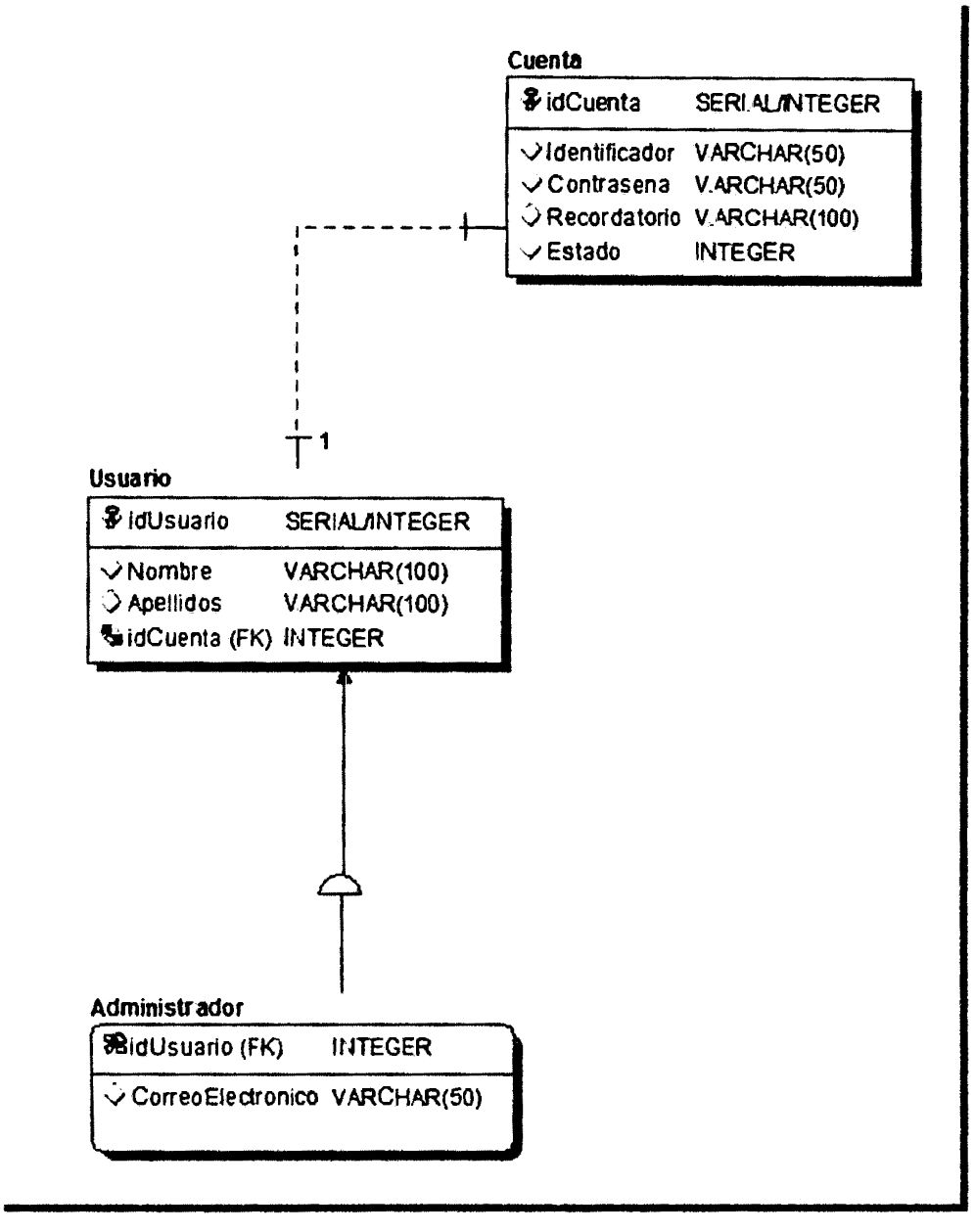
	objetos que contemplan escenarios para cada caso de uso			
	7. Actualizar el modelo de dominio	Modelo de dominio actualizado	NO	
	8. Actualizar el diagrama de clases de análisis	Modelo de dominio actualizado	NO	
	11. Revisar descripción de caso de uso	Caso de uso	NO	
	12. Revisar diagrama de robustez	Diag ama de robustez	NO	
	13. Revisar modelo de dominio actualizado	Modelo de dominio actualizado	NO	
DISEÑO	5. Dividir el modelo de dominio actualizado para cada caso de uso	Parte el modelo de dominio actualizado	NO	
	6. Dibujar un diagrama de secuencia para cada caso de uso	Diagrama de secuencia	NO	
	7. Actualizar el diagrama de clases de un caso de uso	Diagrama de clases	SI	ALTA
	8. Extraer controladores para pruebas unitarias	Lista de controladores	NO	
	5. Revisar el diagrama de secuencia	Diagrama de secuencia	NO	
	6. Revisar el diagrama de clases	Diagrama de clases	NO	
	7. Revisar modelo de dominio actualizado	Modelo de dominio actualizado	NO	
	8. Revisar lista de pruebas unitarias	Lista de controladores	NO	
IMPLEMENTACIÓN	1. Generar las clases de dominio	Código fuente	NO	
	2. Codificar y probar	Código fuente	NO	

	3. Realizar las pruebas del sistema y aceptación del usuario	Prueba de sistema	NO	
		Prueba de aceptación	NO	
	3. Revisar el código	Código fuente	NO	
	4. Actualizar el modelo de dominio	Modelo de dominio actualizado	NO	
PRUEBAS	1. Describir la estrategia de la prueba	Plan de prueba	NO	
	2. Diseñar los casos de prueba	Casos de prueba	SI	MEDIA

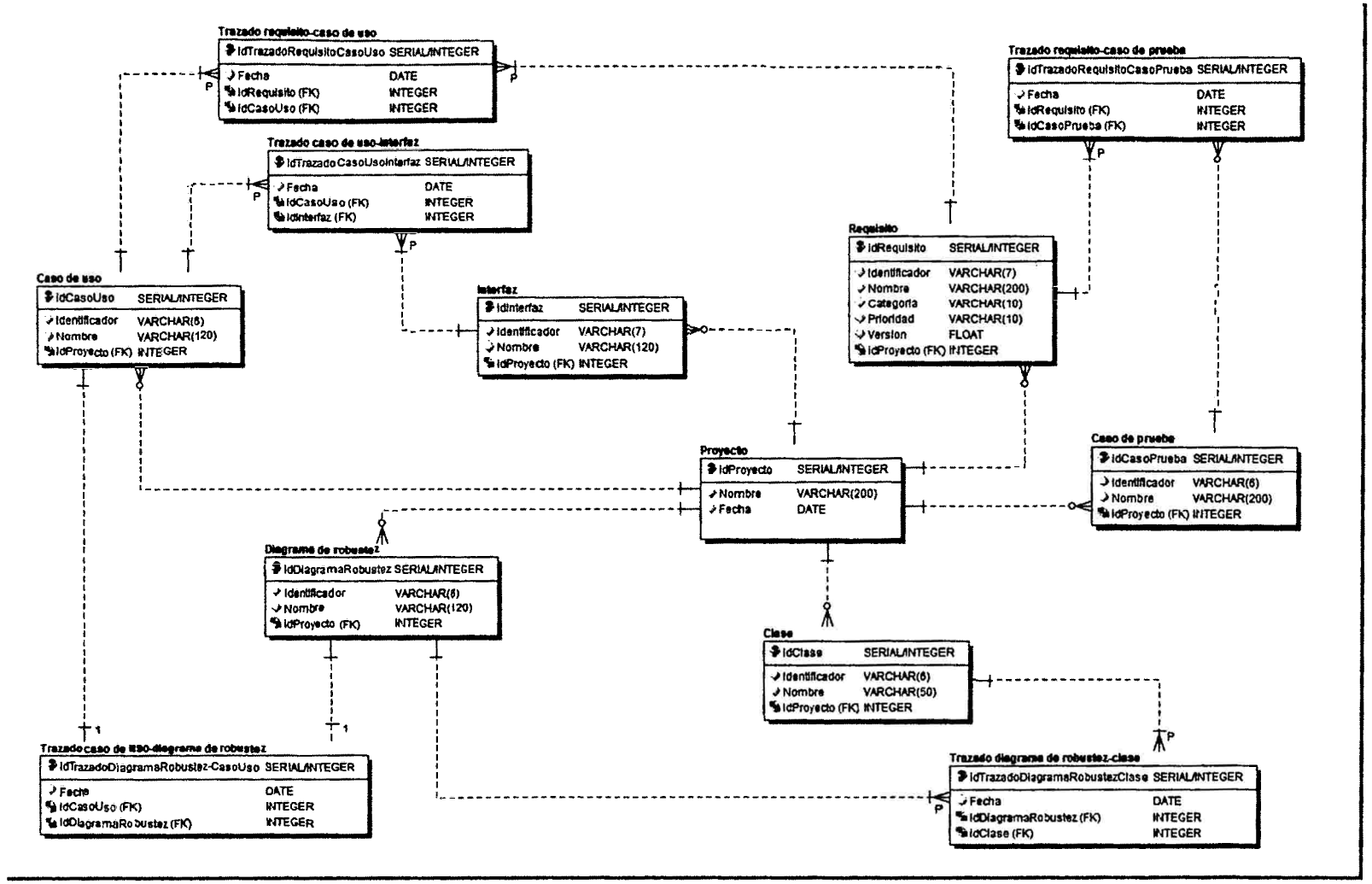
ANEXO C. FICHA DE ANÁLISIS DOCUMENTAL DE LA CONSISTENCIA DEL TRAZADO

ARTEFACTO	RELACIÓN	REGLAS DE CONSISTENCIA
Requisitos	<<trace>>	1°. Un requisito funcional es trazado a un caso. 2°. Un requisito no funcional es trazado a uno o más casos de uso. 3°. Un requisito no funcional no es aplicable a ningún caso de uso. 4°. Un requisito funcional es trazado a uno o más casos de prueba. 5°. Un requisito no funcional es trazado a uno o más casos de prueba. 6°. Un requisito no funcional no es aplicable a ningún caso de prueba.
Caso de uso	<<realize>>	1°. Un caso de uso realiza un requisito funcional. 2°. Un caso de uso realiza un requisito funcional y un requisito no funcional.
	<<trace>>	3°. Un caso de uso es trazado a una o más interfaces. 4°. Un caso de uso es trazado a un diagrama de robustez.
Interfaz	<<realize>>	1°. Una interfaz realiza uno o dos casos de uso.
Diagrama de robustez	<<realize>>	1°. Un diagrama de robustez realiza un caso de uso.
	<<trace>>	2°. Un diagrama de robustez traza a una o más clases.
Clases	<<realize>>	1°. Una clase es realizada en uno o más diagramas de robustez.
Caso de prueba	<<realize>>	1°. Un caso de prueba realiza uno o más requisitos funcionales. 2°. Un caso de prueba realiza uno o más requisitos no funcionales 3°. Un caso de prueba realiza un requisito funcional y un requisito no funcional.

ANEXO D. MODELO DE BASE DE DATOS PARA LA ADMINISTRACIÓN DE USUARIOS



ANEXO E. MODELO DE BASE DE DATOS PARA EL SOFTWARE



ANEXO F. CÓDIGO FUENTE

```
1
2  body {
3          margin:0;
4          padding:0;
5          background:#BBD9EE;
6          color:#000;
7          font-family:tahoma,arial,sans-serif;
8          font-size:11px;
9          }
10 form {
11         margin:0;
12         padding:0
13         }
14 img {border:none;}
15 a {color:#060606;text-decoration: none}
16 a:hover {text-decoration: underline}
17 input {vertical-align:middle}
18 .floatleft {float:left !important}
19 .floatright {float:right !important}
20 .clear {clear:both !important}
21 .bold {font-weight:bold !important}
22 .normal {font-weight:normal !important}
23 .block {display:block !important}
24 input.text,
25 select,
26 textarea {
27         font-family:arial,sans-serif;
28         color:#333;
29         font-size:12px;
30         vertical-align:middle;
31         }
32 input.text {
33         padding:1px 0 0 4px;
34         height:14px;
35         font-weight:normal;
36         }
37
38 #main {
39         width:992px;
40         margin:0 auto;
41         }
42
43 #header {
44         position:relative;
45         width:992px;
```

Figura Nº F.1: Hoja de estilos(CSS).

```
1   package bean;
2
3   public class BCasoUso {
4
5       private int idCasouso;
6       private String Identificador;
7       private String Nombre;
8       private int idProyecto;
9       private int Existe = 0;
10      private int Resultado;
11
12      public int getExiste() {
13          return Existe;
14      }
15
16      public void setExiste(int Existe) {
17          this.Existe = Existe;
18      }
19
20      public String getIdentificador() {
21          return Identificador;
22      }
23
24      public void setIdentificador(String Identificador) {
25          this.Identificador = Identificador;
26      }
27
28      public String getNombre() {
29          return Nombre;
30      }
31
32      public void setNombre(String Nombre) {
33          this.Nombre = Nombre;
34      }
35
36      public int getResultado() {
37          return Resultado;
38      }
39
40      public void setResultado(int Resultado) {
41          this.Resultado = Resultado;
42      }
43
44      public int getIdCasouso() {
45          return idCasouso;
46      }
```

Figura N° F.2: Java Bean Caso de uso.

```
1   package bean;
2
3   public class BTrazadoRequisitoCasoUso {
4
5       private int idTrazadoRequisitoCasoUso;
6       private String fecha;
7       private int idRequisito;
8       private int idCasouso;
9       private int Existe= 0;
10      private int Resultado;
11
12      public int getExiste() {
13          return Existe;
14      }
15
16      public void setExiste(int Existe) {
17          this.Existe = Existe;
18      }
19
20      public int getResultado() {
21          return Resultado;
22      }
23
24      public void setResultado(int Resultado) {
25          this.Resultado = Resultado;
26      }
27
28      public String getFecha() {
29          return fecha;
30      }
31
32      public void setFecha(String fecha) {
33          this.fecha = fecha;
34      }
35
36      public int getidCasouso() {
37          return idCasouso;
38      }
39
40      public void setIdCasouso(int idCasouso) {
41          this.idCasouso = idCasouso;
42      }
43
44      public int getIdRequisito() {
45          return idRequisito;
46      }
```

Figura N° F.3: Java Bean Trazado Requisito - Caso de uso.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/j
3      <context-param>
4          <param-name>driver</param-name>
5          <param-value>org.postgresql.Driver</param-value>
6      </context-param>
7      <context-param>
8          <param-name>url</param-name>
9          <param-value>jdbc:postgresql://localhost:5432/bdtr
10     </context-param>
11     <context-param>
12         <param-name>username</param-name>
13         <param-value>postgres</param-value>
14     </context-param>
15     <context-param>
16         <param-name>pwname</param-name>
17         <param-value>adminadmin</param-value>
18     </context-param>
19     <servlet>
20         <servlet-name>SAutenticar</servlet-name>
21         <servlet-class>servlet.SAutenticar</servlet-class>
22     </servlet>
23     <servlet>
24         <servlet-name>SRegistrarProyecto</servlet-name>
25         <servlet-class>servlet.SRegistrarProyecto</servlet
26     </servlet>
27     <servlet>
28         <servlet-name>SListarProyecto</servlet-name>
29         <servlet-class>servlet.SListarProyecto</servlet-cl
30     </servlet>
31     <servlet>
32         <servlet-name>SBuscarProyecto</servlet-name>
33         <servlet-class>servlet.SBuscarProyecto</servlet-
34     </servlet>
35     <servlet>
36         <servlet-name>SModificarProyecto</servlet-name>
37         <servlet-class>servlet.SModificarProyecto</servl
38     </servlet>
39     <servlet>
40         <servlet-name>SGuardarCuenta</servlet-name>
41         <servlet-class>servlet.SGuardarCuenta</servlet-c
42     </servlet>
43     <servlet>
44         <servlet-name>SListarCuenta</servlet-name>
45         <servlet-class>servlet.SListarCuenta</servlet-cl
46     </servlet>
```

Figura N° F.4: Archivo de configuracion xml.

```
1   package datasource;
2   import java.sql.Connection;
3   import java.sql.DriverManager;
4   import java.sql.SQLException;
5   import javax.servlet.ServletContext;
6
7   public class DsConexion {
8       private String driver;
9       private String url;
10      private String username;
11      private String pwname;
12
13      public DsConexion(ServletContext ctx) {
14          this.driver = ctx.getInitParameter("driver");
15          this.url = ctx.getInitParameter("url");
16          this.username = ctx.getInitParameter("username");
17          this.pwname = ctx.getInitParameter("pwname");
18      }
19      public Connection getConnection() {
20          try {
21              Class.forName(driver).newInstance();
22              Connection connection =
23                      DriverManager.getConnection(this.url,
24                      this.username, this.pwname);
25              return connection;
26          } catch (SQLException sqle) {
27              System.out.println("Error: "
28                      + "DsConexion.getConnection"
29                      + sqle.getMessage());
30              return null;
31          } catch (Exception e) {
32              system.out.println("Error: "
33                      + "DsConexion.getConnection"
34                      + e.getMessage());
35              return null;
36          }
37      }
38      public void closeConnection(Connection conn) {
39          try {
40              if (conn != null) {
41                  conn.close();
42                  conn = null;
43              }
44          } catch (SQLException sqle) {
45              System.out.println("Error: "
46                      + "DsConexion.closeConnection"
47                      + sqle.getMessage());
48          }
49      }
50  }
```

Figura Nº F.5: Archivo Data Source Connection.

ANEXO G. DISEÑO DE REPORTES

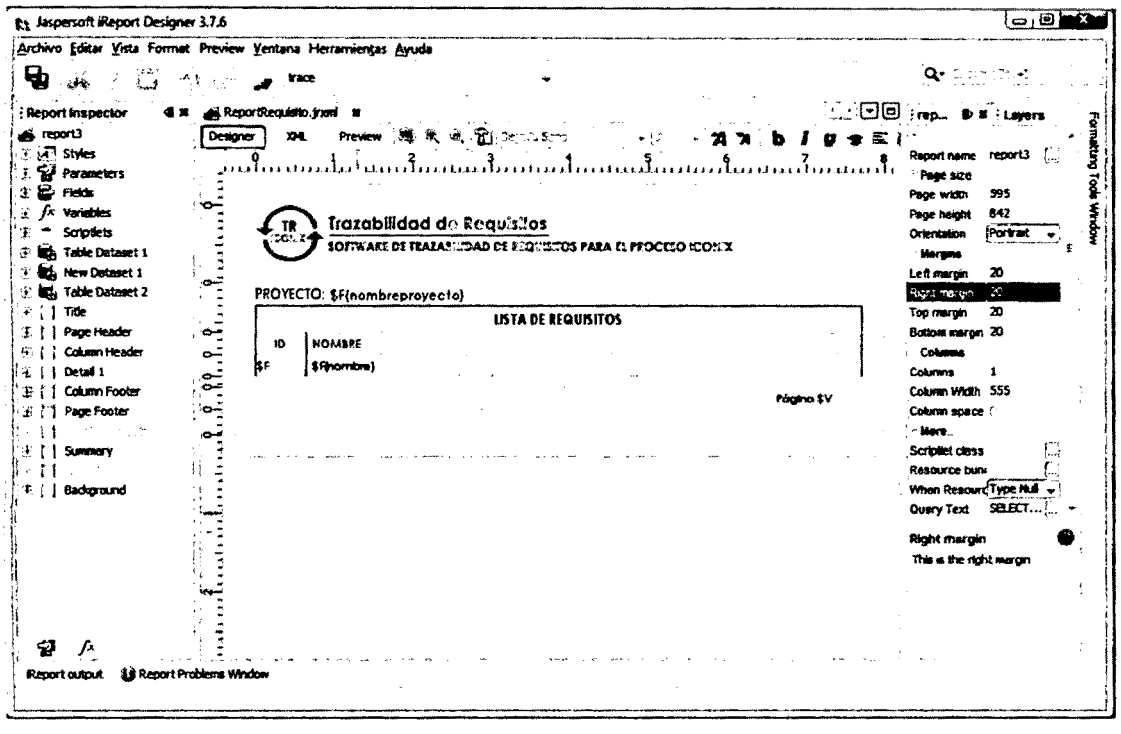


Figura N° G.1: Diseño del reporte para la lista de requisitos.

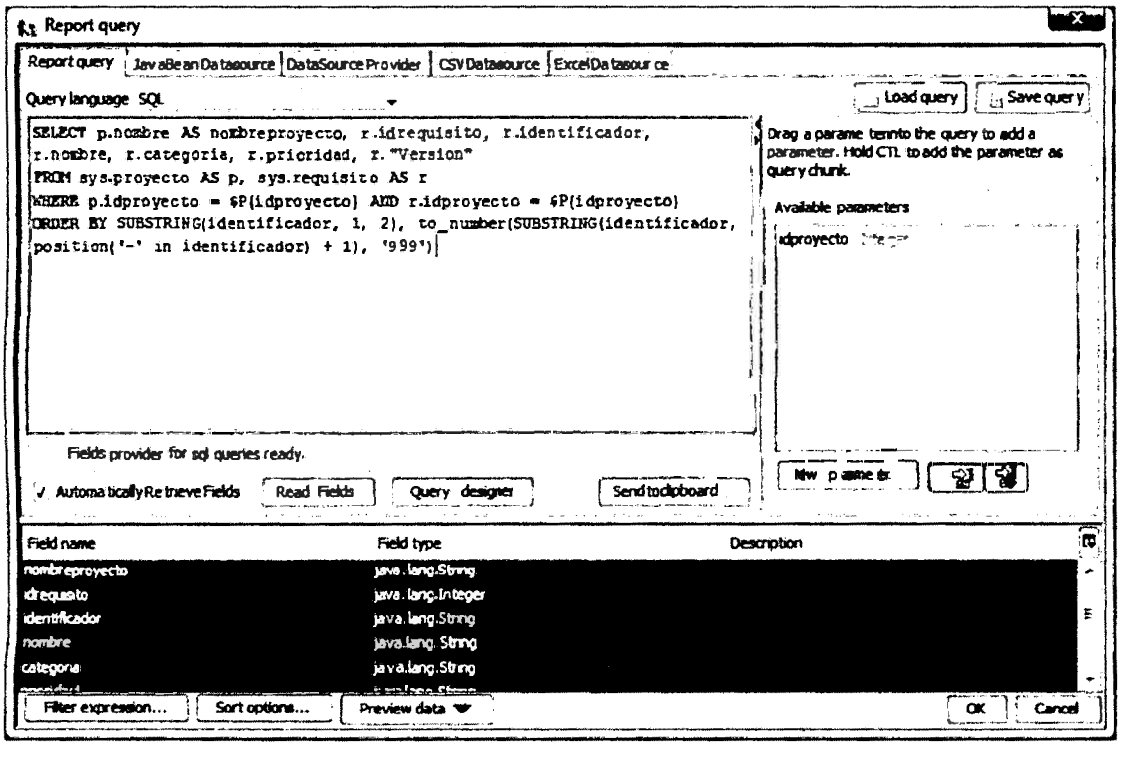


Figura N° G.2: Consulta del reporte para la lista de requisitos.

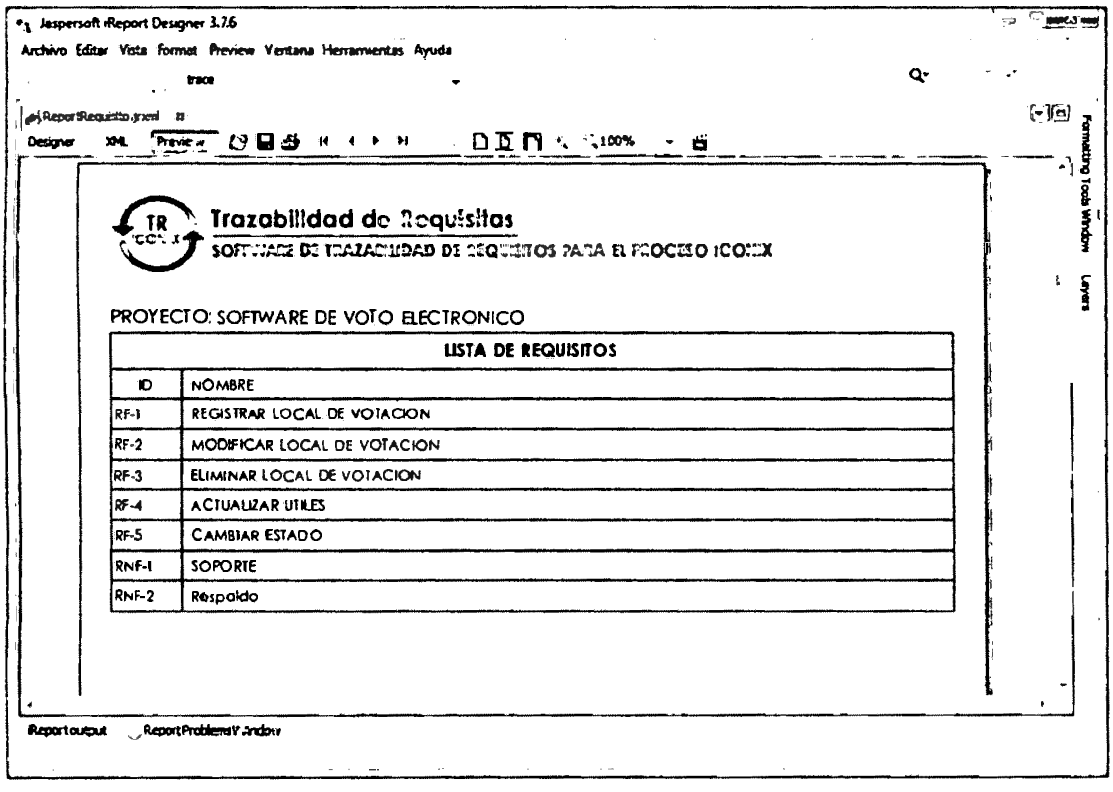


Figura N° G.3: Vista previa del reporte para la lista de requisitos.

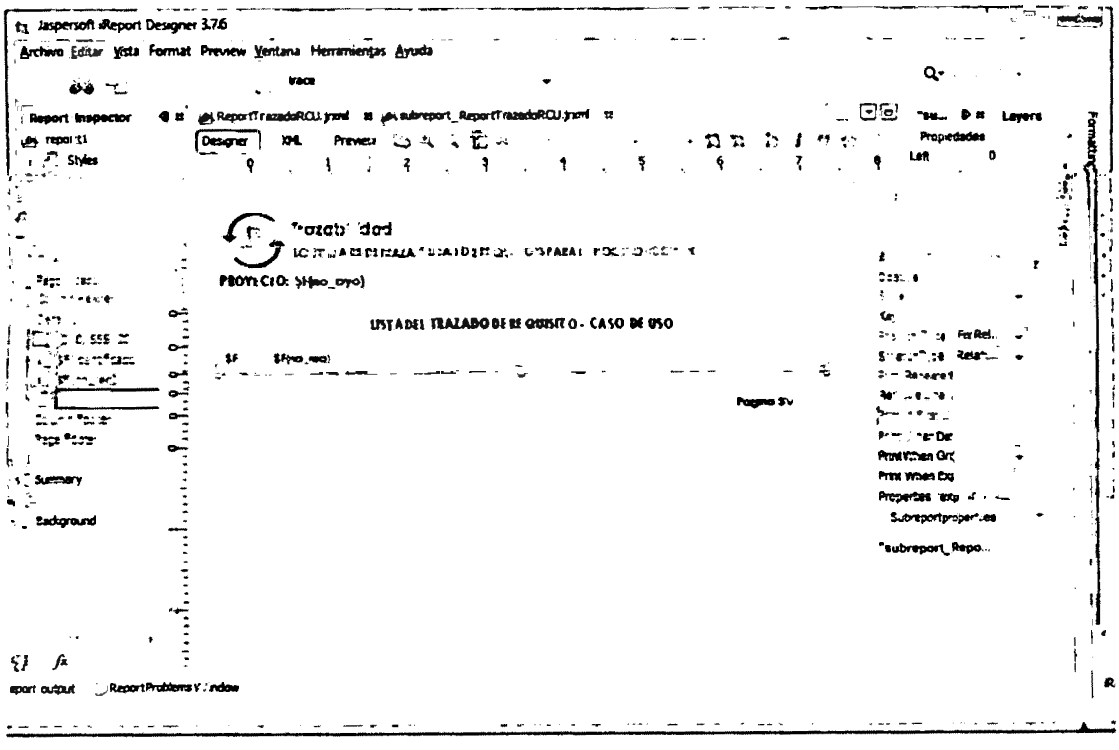


Figura N° G.4: Diseño de reporte para la lista de trazado requisito – caso de uso.

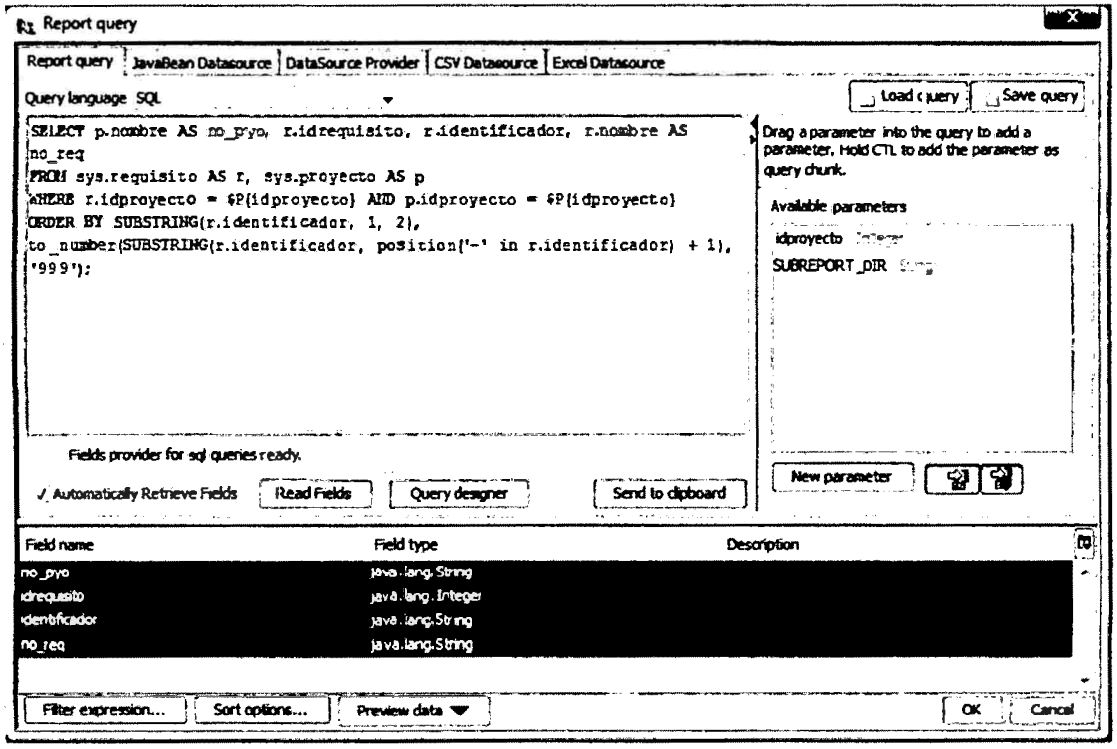


Figura N° G.5: Consulta del reporte para la lista de trazado requisito – caso de uso.

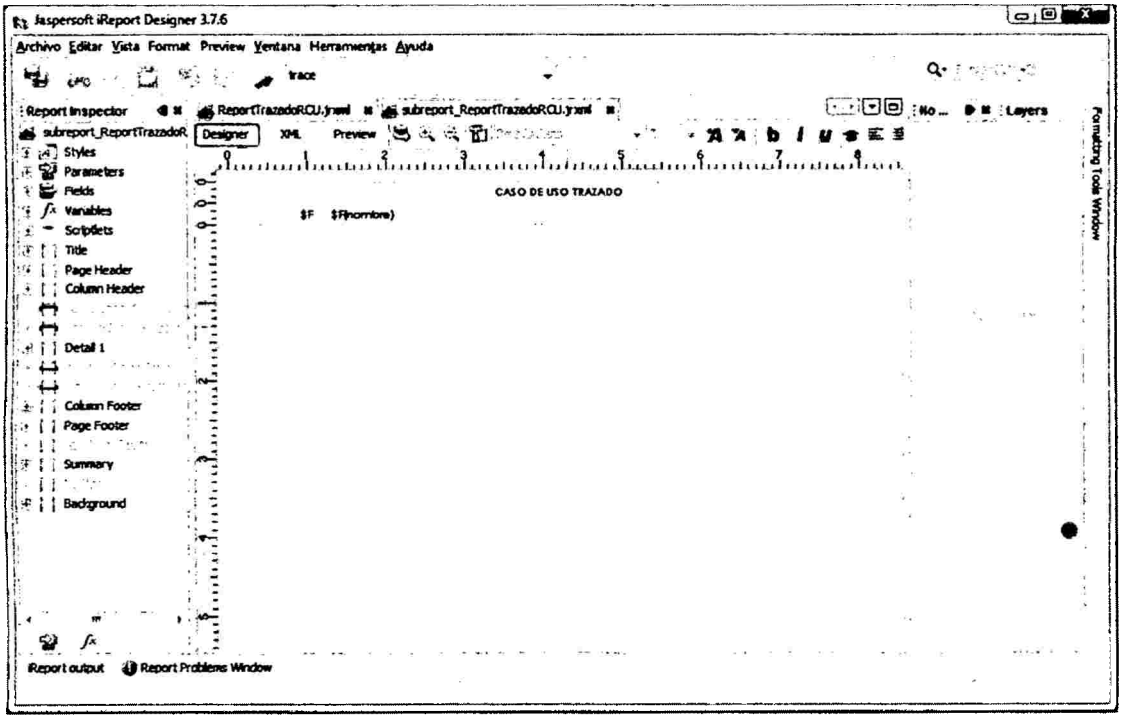


Figura N° G.6: Subreport para la lista de trazado requisito – caso de uso.

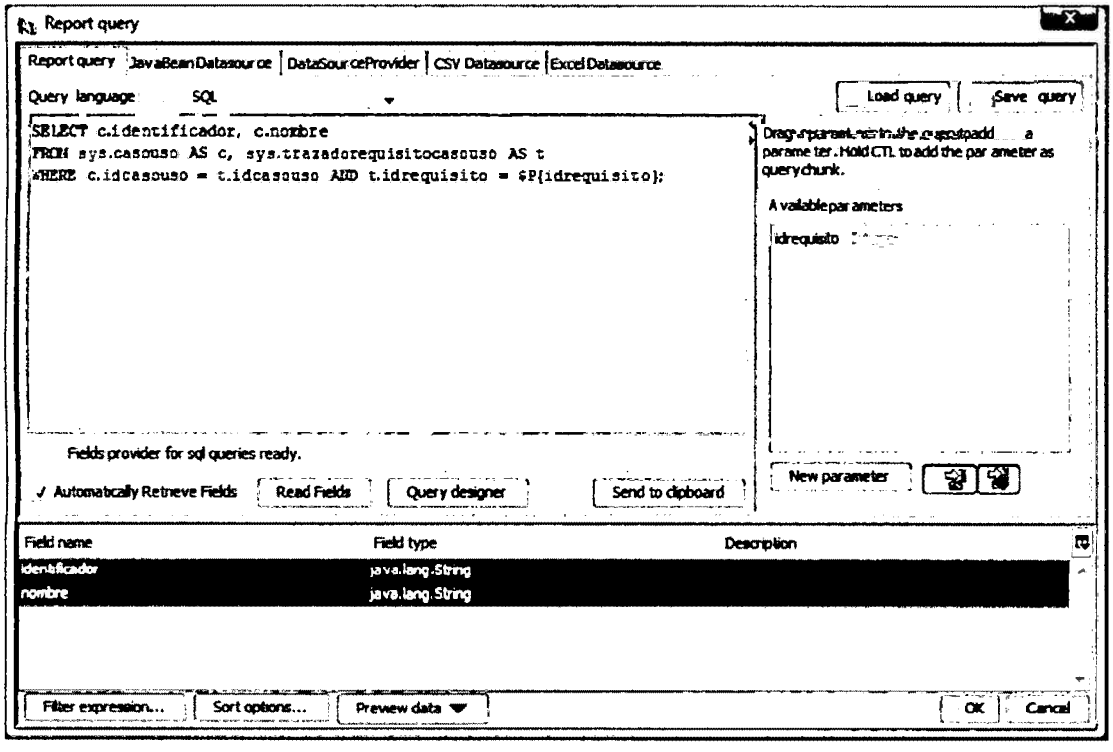


Figura N° G.7: Consulta del subreporte para la lista de trazado requisito – caso de uso.

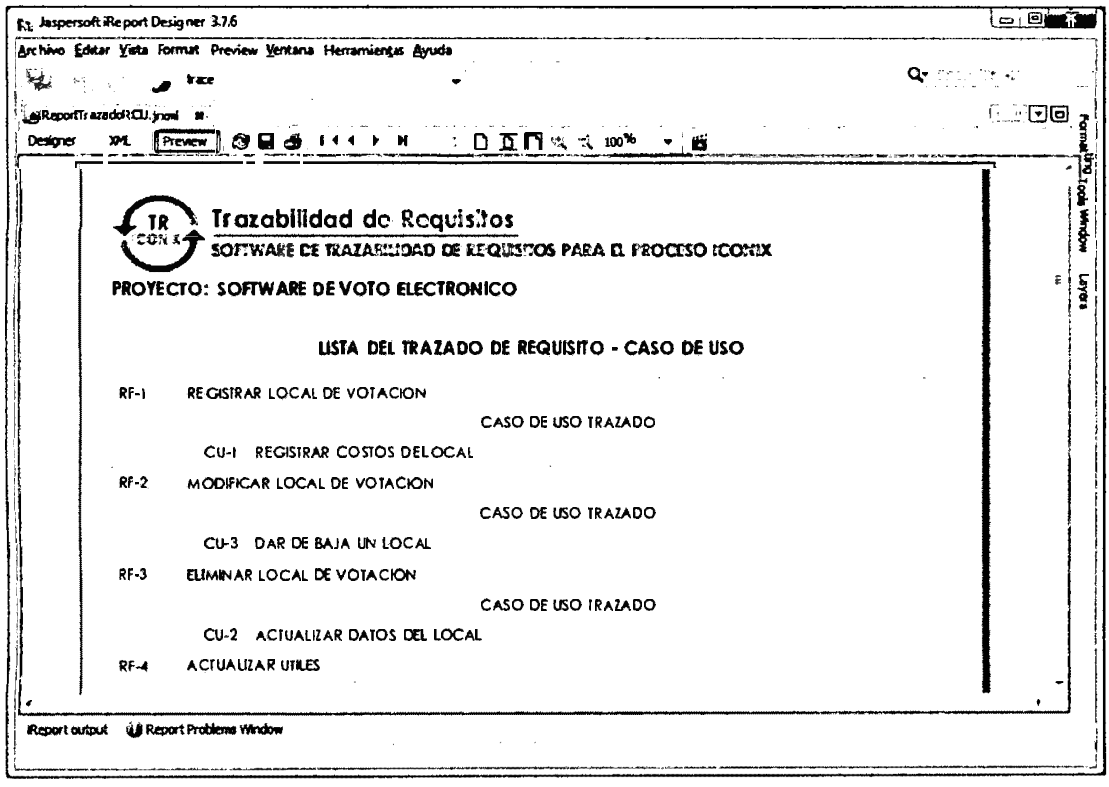


Figura N° G.8: Vista previa del reporte para la lista de trazado requisito – caso de uso.