Universidad de las Ciencias Informáticas

Facultad 5

Plugin de Gestión de Requisitos para la Herramienta Trac.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Alezenny Sablón Laffita

Abel Chávez Agüero

Tutores: Ing. Amado Espinosa Hidalgo

Ing. Lannie Octavio Herrera Pérez

Ciudad de la Habana 2009

Datos de Contacto

Nombre y apellidos: Amado Espinosa Hidalgo

Edad: 29 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero Informático

Categoría docente: Profesor Asistente

Categoría científica: Investigador

E-mail: aespinosa@uci.cu

Graduado de la CUJAE desde hace 4 años, con 6 años de experiencia en el desarrollo de software, actualmente forma parte del proyecto SCADA “Guardián del Alba” en la Universidad de las Ciencias Informáticas.

Nombre y apellidos: Lannie Octavio Herrera Pérez

Edad: 26 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría docente: Profesor Instructor

E-mail: lherrera@uci.cu

Graduado en la UCI en 2007 con Título de Oro, siendo Premio Mella por sus resultados en los 5 años de carrera. Tiene 2 años de graduado y más de 4 en el desarrollo de software.

Agradecimientos

A todas las personas que han hecho su aporte para lograr que hoy se esté cumpliendo este sueño, a nuestros tutores y a la Universidad de las Ciencias Informáticas.

Dedicatoria

… el presente trabajo está dedicado a mi familia, en especial a mis padres, que siempre me han apoyado en todo y han hecho posible la realización de este sueño, a mi hermana que siempre ha estado junto a mí y a mi abuela por todo su cariño y comprensión.

… a mis amigos Lisandro, Yordan y Dalina por su apoyo en todo momento.

… a todas las personas involucradas en mi formación como profesional.

Abel

… a mis padres por su amor incondicional, el apoyo y comprensión que siempre me han dado, por ser los principales autores de mi formación y educación y porque sin ellos nada hubiese sido posible.

… a mis hermanos por ayudarme tanto cada vez que lo he necesitado y por el apoyo que siempre me han brindado.

… a mis abuelos por todo el cariño que siempre me han dado, y a mi cuñada Yudelmis por como ha sido siempre conmigo.

… a mi novia Taimy por dar lo mejor de sí para ayudarme y comprenderme en todo momento.

… a toda mi familia y las personas importantes en mi vida.

Alezenny

Resumen

La Gestión de Requisitos puede ser determinante en la aceptación del producto final de un proyecto software, pues el primer paso para obtener una solución con calidad es contar con buenos requisitos, que pueden cambiar a lo largo del ciclo de desarrollo del software. La Gestión de Requisitos permite llevar un seguimiento de los requerimientos del software y los cambios que estos pueden sufrir durante el ciclo de vida de un proyecto.

Con el fin de lograr la eficiente puesta en práctica de esta disciplina en los proyectos del Polo Productivo de Hardware y Automática (HA), se ha realizado el presente trabajo, que se centra en la Gestión de Requisitos de manera que permita la realización de tareas inherentes al proceso que facilitarán el trabajo de los miembros de los proyectos productivos del polo. El producto final será una extensión integrada a la herramienta Trac, utilizada en los proyectos del polo para diversas tareas de Gestión de Proyectos.

Palabras clave

Requisitos, Gestión, Extensión.

Tabla de contenido

Introducción

Desde hace algunas décadas hasta nuestros días, la Gestión de Requisitos se ha hecho indispensable en el desarrollo de los proyectos de software. Uno de sus objetivos es establecer una clara comunicación entre el cliente y el grupo de desarrolladores, facilitando la comprensión de los problemas a resolver y las posibles vías de solución de los mismos. Se ha demostrado mundialmente que esta fase es una de las más difíciles de todo el proceso de Ingeniería de Software y que los errores que se cometen en esta etapa, afectan en gran medida la construcción del sistema. Por otra parte la corrección de estos errores implica grandes costos de tiempo y recursos. Realizar una correcta Gestión de Requisitos desde los inicios de un proyecto puede ayudar a reducir los recursos utilizados en el desarrollo del software y disminuir el tiempo empleado en la realización del mismo, logrando la satisfacción del cliente.

A nivel mundial la Gestión de Requisitos es tratada con cuidado por su importancia y papel jugado en el desarrollo del software, es por ello que la mayoría de las compañías y especialistas en el tema utilizan herramientas para mejorar tanto la productividad como la calidad en el desarrollo del proyecto o software. Las compañías productoras de software han tenido la necesidad de emplear estas herramientas por la gran complejidad de la Gestión de Requisitos.

El polo de Hardware y Automática (HA) de la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI) cuenta con varios proyectos productivos. Los trabajadores de estos proyectos ponen su empeño en desarrollar sus productos con calidad, tarea que se dificulta debido a que no cuentan con una herramienta de Gestión de Requisitos, pues tras realizarse un estudio se llegó a la conclusión de que las herramientas libres existentes no satisfacen las necesidades requeridas o su proceso de instalación resulta demasiado complicado. En el polo se utiliza la herramienta Trac para la Gestión de Proyectos, pues esta ofrece ventajas que agilizan el trabajo en equipo de los miembros de los proyectos. La flexibilidad de Trac para incorporarle nuevas funcionalidades así como su libertad de uso y modificación, son características que condicionan el desarrollo de una solución que fortalezca la automatización de algunos procesos de la disciplina Gestión de Requisitos. Es por ello que se ha considerado el siguiente **problema científico**: ¿Cómo desarrollar una funcionalidad para la utilización de la herramienta Trac en la Gestión de Requisitos de software?

Teniendo en cuenta el problema científico expuesto anteriormente se define como **objeto de estudio:** el proceso de Gestión de Requisitos de software.

Por tanto, el **objetivo general** del trabajo es: desarrollar una solución que permita a la herramienta Trac ser utilizada para la Gestión de Requisitos de software, incorporando funcionalidades básicas acorde a las tendencias de esta disciplina, lo que determina que el **campo de acción** sea: Herramientas de Gestión de Requisitos de software.

De esta manera, las **tareas investigativas** en que se centrará el trabajo son las siguientes:

- Revisión de la información existente para la elaboración del Estado del arte en la Gestión de Requisitos.
- Estudio de diferentes herramientas libres y comerciales utilizadas en esta rama de la Ingeniería del Software para el análisis y aplicación al trabajo de los resultados y conocimientos obtenidos.
- Estudio de la herramienta Trac y su extensión mediante plugins para incorporar y aumentar los conocimientos acerca de cómo aumentar las funcionalidades de la misma.
- Definición del alcance de la solución y planificación de los ciclos de desarrollo para determinar hasta donde se quiere llegar en el desarrollo e implementación del trabajo.
- Desarrollo de la solución para la obtención de los resultados esperados según el estudio realizado.
- Realización de pruebas de aceptación para comprobar el correcto funcionamiento de la solución.

Acorde con el problema científico planteado la **idea a defender** es la siguiente: el desarrollo de una funcionalidad que permita utilizar la herramienta Trac para la Gestión de Requisitos facilitará el trabajo en los proyectos productivos del polo de HA.

Con el fin de cumplir las tareas que regirán la investigación se emplearán algunos métodos para la búsqueda de información, estos son:

Métodos teóricos:

Analítico-sintético: Con el fin de realizar un estudio de trabajos desarrollados con anterioridad referentes al tema, y extraer los datos y conceptos más importantes para el desarrollo del trabajo con el cumplimiento de sus objetivos.

Análisis histórico-lógico: Con el objetivo de conocer los antecedentes de la Gestión de Requisitos, sus tendencias hasta la actualidad y la evolución de su funcionalidad.

Métodos empíricos:

Entrevista: Para adquirir conocimientos, recomendaciones, y orientaciones en el tema Gestión de Requisitos.

Experimento: El trabajo culminará con la elaboración de un plugin para la herramienta Trac que se someterá a pruebas, y servirá para llevar a cabo diferentes tareas propias de la disciplina Gestión de Requisitos.

Capítulo 1: Fundamentación teórica

Introducción

El proceso de Gestión de Requisitos es una disciplina en auge y desarrollo en todo el mundo, lo que ha influido grandemente en que exista un creciente interés por parte de los especialistas de llevarlo a cabo de manera correcta y eficiente, pues esto sería un gran aporte al ahorro de tiempo y recursos en la realización de los proyectos y a la calidad final de los productos obtenidos.

En el presente capítulo se exponen algunos de los principales conceptos relacionados con los requisitos, sus clasificaciones y características, la gestión de los mismos, la necesidad de llevar a cabo el proceso y las tareas en que se centra. También se hace mención de algunas herramientas empleadas en la actualidad para automatizar las tareas de la disciplina Gestión de requerimientos, los sistemas operativos que las soportan y algunas características de las mismas. Para el desarrollo de la solución se requiere del empleo de varias herramientas y tecnologías a las que también se hace referencia en los siguientes epígrafes.

1.1 Concepciones generales

Para que un proyecto se ejecute con éxito y logre cumplir o superar las expectativas del cliente es necesario que cuente con buenos requisitos y que se pueda garantizar la calidad sobre los productos desarrollados, lo que se dificulta cada vez más, pues cada día que pasa las empresas desarrolladoras de software se mueven en un entorno más competitivo, viéndose en la necesidad de disminuir el tiempo de obtención del producto final con el fin de sacarlo a la venta, a la vez que se mantienen las mismas exigencias por parte de los clientes. En muchas ocasiones las empresas acaban retrasando la salida al mercado de sus productos, por lo que relegan frases relacionadas con la calidad del producto al plano meramente administrativo o incluso llegan a obviarlas. La calidad en un proceso de desarrollo de software debe ser garantizada desde los inicios del mismo, por lo que es decisiva a la hora de garantizar la misma una correcta especificación de los requisitos de la aplicación.

Establecer y controlar el alcance de los proyectos de software es una de las tareas más críticas en el proceso de desarrollo ya que el éxito o fracaso del proyecto está fuertemente relacionado con la calidad de sus requisitos. Los requisitos capturan las necesidades de los usuarios, fijando el alcance del proyecto y estableciendo la base sobre la cual desarrolla su trabajo todo el equipo de proyecto.

Para el desarrollo del trabajo, que tiene como objetivo principal añadir a la herramienta Trac una funcionalidad que permita llevar a cabo los procesos de Gestión de Requisitos es necesario conocer que es un requisito y sus clasificaciones. En este epígrafe se da una breve descripción de estos temas.

1.1.1 Requisitos

Un requisito es una condición o capacidad que debe cumplirse para resolver un problema o alcanzar un objetivo. Cada requisito vendrá caracterizado por: (1)

- Identificador único y nombre descriptivo.
- Persona u organización y fecha en la que se presenta el requisito.
- Sistemas afectados por el requisito
- Descripción de la capacidad o funcionalidad que debe presentar el sistema a desarrollar. Cada requisito deberá ser descrito de tal manera que satisfaga las características de necesario, conciso, completo, consistente, no ambiguo y verificable.
- Tipo de requisitos (funcionales y no funcionales).
- Prioridad tanto para el sistema como para el usuario.
- Estado del requisito.
- Documentación asociada.

Los requisitos pueden cambiar en alguna etapa del desarrollo del proyecto, estos cambios deben controlarse, documentarse y persistir a lo largo de todo el ciclo de vida del sistema. De los requisitos se debe conocer su origen, necesidad, relación con otros requisitos y la relación con elementos del diseño

y/o la implementación. Esta información es útil para saber qué afectación podría traer una modificación o cambio en un requisito.

La trazabilidad de requisitos se define como la habilidad para describir y seguir la vida de un requisito en ambos sentidos, hacia sus orígenes o hacia su implementación, a través de todas las especificaciones generadas durante el proceso de desarrollo de software. (2)

Los requisitos pueden agruparse o clasificarse en:

- Funcionales: indican características y restricciones sobre la funcionalidad del software, además son la condición necesaria de un atributo para que cumpla una función determinada. (3)

  Requisitos sobre la actualización de datos: son características sobre las funciones que cambian la información del sistema. Debe estudiarse de qué forma y bajo que restricciones el usuario desea que se introduzcan nuevos datos, se cambien los que ya existen o se eliminen.

  Requisitos sobre la estructura de información: son características de los datos que el software maneja. (4)

- No funcionales: son propiedades o cualidades que el producto debe tener. Los requisitos no funcionales deben establecer restricciones en el producto que está siendo desarrollado, en el proceso de desarrollo y en restricciones específicas que el producto pueda tener. (5) Pueden clasificarse en:

  Requisitos de rendimiento: son límites al rendimiento (para aquellas aplicaciones donde existan) y volúmenes de información que el software debe tratar.

  Requisitos de seguridad: son características de control de acceso al software y copias de seguridad, entre otros relacionados con la seguridad del sistema y la información.

  Requisitos de frecuencia de tratamiento: son características sobre la frecuencia con que se ejecutan las diferentes funciones del software. (4)

- De implantación

  Requisitos sobre la arquitectura: características de los nodos y el software de base que compondrán el sistema.

  Requisitos sobre el despliegue: controlan en qué nodos se podrán ejecutar las diferentes partes del software.

  Requisitos de comunicaciones: son características de la red de ordenadores que compone el sistema y necesidades de conexión con redes externas. (4)

Los requisitos constituyen uno de los primeros entregables, si no el primero, en el proceso de desarrollo de aplicaciones software. De hecho, es la aparición de unas determinadas necesidades lo que motiva el desarrollo de un sistema o aplicación, el cual tiene como fin solucionar esas necesidades o deficiencias detectadas. (6) Los requisitos son estas necesidades plasmadas de forma clara y concisa, es por ello que esta parte de la gestión de requisitos, es una de las más importantes del proceso de desarrollo y que en mayor o menor medida condicionará el éxito del proceso global.

1.1.2 Gestión de Requisitos

Uno de los mayores desafíos a que se puede enfrentar cualquier organización orientada al proceso de creación de software en la actualidad es asegurar que el desarrollo de sus productos se cumpla en los plazos establecidos, y que esto se logre con los recursos estimados, con el alcance y la calidad deseada y sobre todo logrando la completa satisfacción del cliente. Con mucha frecuencia aparecen situaciones en las que los requisitos se ven sujetos a cambios. En ocasiones el propio cliente no logra definir con claridad los requisitos para su producto, e incluso existen contradicciones en la serie de requisitos que plantea. En otras ocasiones el origen del problema está en el entendimiento común entre lo que desea el cliente y lo que entiende el desarrollador, pues aunque ambas partes creen enfocar el problema de la misma manera esto no es lo que ocurre en realidad y las consecuencias se reflejan más adelante. En este entorno la Gestión de Requisitos juega un papel fundamental.

La gestión de requisitos se identifica actualmente como una muy buena práctica que contribuye, como ninguna otra, al éxito de los proyectos de software, al posibilitar un entendimiento común entre el cliente y el grupo de desarrolladores de los requisitos del cliente que deben concebirse en el producto final, la comprensión de los problemas que se necesitan solucionar y las posibles vías de resolverlos. Asimismo, esta etapa se considera y se ha demostrado que es una de las más difíciles de todo el proceso de ingeniería de software y que, de ser errada, más afecta la construcción del sistema y más difíciles son de corregir los problemas resultantes. (7) Puede calificarse también como un enfoque sistemático mediante el cual se puede levantar, organizar y documentar los requisitos con que debe cumplir el sistema, o como un proceso que establece y mantiene el acuerdo entre el cliente y el equipo del proyecto sobre los requerimientos del sistema y los cambios que estos pueden sufrir en el futuro.

Los requisitos se inician cuando empieza un proyecto en las etapas de análisis y especificación de requisitos, posteriormente, dichos requisitos en el ciclo de vida de un proyecto pueden ser modificados por lo que se establece el concepto de Gestión de Requisitos, que es el tratamiento y control de las actualizaciones y cambios a los mismos. (8)

La Gestión de Requisitos puede ser definida como el proceso encargado de la identificación, asignación y seguimiento de los requisitos, incluyendo la interfaz, verificación, modificación y control a lo largo del ciclo de vida del software. Es el conjunto de actividades que se centra en el aseguramiento de las especificaciones, ejemplo de esto son los requisitos que debemos cumplir para la satisfacción del cliente.

Debido a que los proyectos informáticos son susceptibles a cambios, se debe proceder a la actualización, incorporación y/o eliminación de funcionalidades, esto obliga a mantener controlado y documentado el producto. Los cambios en los requisitos determinan modificaciones del tiempo en el que se va a implementar una característica o funcionalidad en particular, estas modificaciones a la vez pueden tener impacto en otros requerimientos. Estos cambios deben ser gestionados para asegurar que la calidad de los mismos se mantenga, pues los problemas suscitados por los cambios de requisitos podrían incurrir en altos costos.

- La Gestión de Requisitos implica:
    - Definir procedimientos que establezcan los pasos y los análisis que se realizarán antes de aceptar los cambios propuestos.

- Cambiar los atributos de los requisitos afectados.
- Mantener la trazabilidad entre requisitos.
- Controlar las versiones del documento de requisitos.

Tabla 1 Tareas principales de la Gestión de Requisitos (8)

ACTIVIDADES	**DESCRIPCIÓN**
Recolección	Recolección y documentación de requisitos es una actividad de comunicación iterativa entre clientes, gerentes y practicantes (stakeholders del proyecto), para descubrir, definir, refinar y registrar una representación precisa de los requisitos del producto. Varios métodos son utilizados para la recolección de requisitos. Algunos análisis iniciales como es la agrupación, categorización y priorización son desarrollados durante esta actividad.
Documentación	Después que los requisitos han sido recolectados, hay que analizarlos a detalle y documentarlos en una especificación de requisitos. El resultado de la especificación de requisitos y de cualquier especificación de requisitos de componentes hardware/software derivado sirve como registro de convenio con el cliente y compromiso con el proveedor. Estas especificaciones son rastreadas utilizando una matriz de trazabilidad de requerimientos y son sujetos a verificación y gestión de cambio a través del ciclo de vida del producto.
Verificación	Una vez que la especificación de requisitos ha sido desarrollada, los requisitos son verificados. La verificación de requisitos es un proceso para asegurar que la especificación de requisito del producto es una representación exacta de las necesidades del cliente. Este proceso también asegura que los requisitos sean

	trazados y verificados a través de varias fases del ciclo de vida; particularmente en el diseño, implementación y pruebas. Los requisitos deben ser trazados desde fuentes externas, tales como los clientes, para derivar requisitos del nivel del sistema, para especificar requisitos del producto hardware/software. Además, todos estos requerimientos deben ser trazados al diseño, implementación y pruebas para asegurarse que los requerimientos han sido satisfechos.
Gestión de Cambios	Gestión de cambios es un proceso formal para identificar, evaluar, trazar y reportar cambios propuestos y aprobados a la especificación del producto. Como el proyecto va evolucionando, los requerimientos pueden cambiar o expandirse para ajustar algunas modificaciones en el alcance o diseño del proyecto. Un proceso de gestión de cambios proporciona un rastreo completo y preciso de todos los cambios que son pertinentes al proyecto.

1.1.3 Elementos a tener en cuenta en la práctica de la Gestión de Requisitos

La Gestión de Requisitos es una actividad compleja que consume tiempo, y debe ser desarrollada por personas con experiencia. Antes de adentrarse en el proceso se debe tener en cuenta algunos aspectos como son:

- El equipo del proyecto sigue un apropiado ciclo de vida del proyecto.
- Son incluidos en el proyecto clientes, usuarios y demás stakeholders importantes para el desarrollo del mismo.
- Crear matrices de trazabilidad de los requisitos.
- Usar herramientas de gestión de requisitos.

- La buena comunicación es importante durante el proceso de derivar requerimientos.
- Se debe mantener una lista de requerimientos en una base de datos donde se puedan manejar más fácilmente.
- No añadir ni cambiar requisitos sin llevar un análisis de riesgos que determine el impacto en el plan de proyecto y reestimación del costo y planeación del proyecto.

1.2 Herramientas de Gestión de Requisitos

Las actividades relacionadas con la gestión de requisitos, hasta hace no mucho tiempo eran realizadas de forma manual, mediante el uso de procesadores de texto. Con la aparición de modernas herramientas para la definición y gestión de los requisitos se ha logrado un gran avance.

Actualmente existen varias herramientas utilizadas para la realización de este proceso, algunas de uso libre y otras comerciales. El uso de estas herramientas tiene como finalidad mejorar la productividad y calidad en el desarrollo de los proyectos de software. Entre las necesidades a cubrir por estas herramientas está la disponibilidad de la información de los distintos proyectos en un repositorio al cual se pueda acceder por los distintos usuarios con el fin de ser reutilizada. Esta filosofía de reutilización también incluye la gestión de requisitos de un proyecto puesto que estos son potencialmente reutilizables dentro de otros proyectos. La reutilización de la información no solamente ahorrará tiempo y coste sino que también redundará en que la información esté más consolidada y se facilite el trabajo a los usuarios.

La Gestión de Requisitos es una tarea intensiva, en la que debe existir la posibilidad de relacionar diferentes documentos, de crear reportes especiales de estos documentos y de controlar los cambios hechos mediante los documentos de forma consistente entre otras actividades.

1.2.1 Herramientas comerciales de Gestión de Requisitos

Para el desarrollo de este trabajo se procederá primeramente al estudio de diferentes herramientas de uso comercial utilizadas para la disciplina Gestión de Requisitos a nivel mundial. Estas herramientas proporcionan casi todas las necesidades básicas que debe brindar una buena herramienta de Gestión de Requisitos. En general, todas se basan en sistemas centralizados de gestión de bases de datos para

almacenar la información correspondiente a los requisitos, que suele consistir en párrafos de texto libre con una serie de atributos predefinidos y a los que la mayoría de herramientas permiten asociar nuevos tipos de atributos por parte del usuario. Todas las herramientas asumen que la estructura de los requisitos es jerárquica, de forma que un requisito puede estar formado o tener asociados otros requisitos de nivel inferior. (8)

- **Rational Requisite Pro** es una herramienta centrada en documentos, que almacena los requisitos asociándolos a documentos (aunque también permite guardarlos directamente en la base de datos), mientras que las otras herramientas están orientadas a requisitos. Auxilia especialmente en el control de cambio de requisitos, con trazabilidad para especificaciones de software y pruebas. Está muy unido a MS Word ya que es partner de Microsoft Development. La herramienta permite el uso de Oracle soble Unix o Windows como "back-end database" y también soporta SQL Server sobre Windows. (8)

- **CaliberRM** es para sistemas grandes y complejos y proporciona una base de datos de requisitos con trazabilidad. La compañía ve a los requisitos como parte del proceso de gestión de la calidad del software, el cual es considerado también, las pruebas (testing) y el trazado de defectos (defect tracking). Caliber está basado en Internet y maneja referencia de documentos, responsabilidad de usuario, trazabilidad, prioridad y estado entre otras características. (8)

- **IRqA (Integral Requisite Analizer)** es una de las herramientas de Gestión de Requisitos más completas del mercado. Los requisitos que se capturan se almacenan en documentos Word y las descripciones de los mismos pueden referenciar a documentos externos como son tablas, gráficos y hojas de cálculo de Microsoft Excel. Permite establecer relaciones entre requisitos, además se puede integrar con Rational Rose.

- **Telelogic Doors** es un sistema multiplataforma diseñado para la Gestión de Requisitos mediante la captura, trazabilidad, enlazado, análisis y manejo de los cambios que en ellos se realicen. Mediante el uso de Doors se puede realizar un análisis de trazabilidad para identificar las áreas de riesgo, y resulta fácil manejar los cambios que tengan lugar en los requisitos. Además permite gestionar un gran número de requisitos de forma eficiente mediante el uso de una base de datos sencilla, lo que se conoce como característica de gran escalabilidad.

1.2.2 Herramientas libres de Gestión de Requisitos

También existen herramientas de uso libre para realizar las actividades pertinentes a la Gestión de Requisitos. Algunas de ellas son:

- **REM (Requisite Management)** aunque es una herramienta de uso libre puede ser utilizada únicamente sobre Windows, ha sido utilizada con frecuencia para fines educacionales. REM permite generar un documento normalizado en el que se pueden incluir los requisitos necesarios para el desarrollo de sistemas de información. La documentación es generada en formato HTML.

- **DRES** está basado en PHP y la administración de los proyectos se lleva a cabo en un navegador web. Puede ser de gran utilidad para grupos de trabajo distribuidos, ya que se puede acceder al trabajo realizado en incluso hacer modificaciones vía Internet. Para su uso se requiere, además del servidor PHP, de una base de datos MySQL o CORBA para el almacenamiento de los requisitos.

- **OSRMT (Open Source Requeriments Management Tool)** es una herramienta diseñada para dar cobertura a todo el ciclo de vida de desarrollo del software. Dispone de control de versiones, permite definir requerimientos derivados, es posible definir tanto casos de uso como casos de prueba y brinda la posibilidad de definir atributos para los requisitos como son el riesgo y esfuerzo entre otros.

1.3 La herramienta Trac (9)

El Trac es un gestor de proyectos que se utiliza para la gestión, seguimiento y el control de versiones del mismo. Es una herramienta de código libre con interfaz web que integra otras herramientas para comunicación y cuenta además con una Wiki integrada que permite mantener activa y vigente la documentación, una vista de los cambios recientes, un control de hitos para conocer el estado del desarrollo, una interfaz para la revisión del código fuente y una gestión de etiquetas con posibilidad de abrir, asignar y cerrar incidencias. Trac estuvo bajo licencia GPL hasta el 2005, desde la versión 0.9 está disponible bajo la licencia BSD modificada y proporciona un enfoque minimalista a la coordinación de proyectos de software vía web.

Ventajas:

- Desarrollado orientada a componentes.
- Fácil de administrar y usar.
- Extensible.
- Personalizable.

1.4 Tecnologías, herramientas de desarrollo, metodología y lenguajes utilizados

1.4.1 Sistema Operativo GNU/Linux

GNU/Linux es un Sistema Operativo muy eficiente que posee un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador permitiendo que múltiples procesos estén ejecutándose al mismo tiempo. En las plataformas Intel corre en modo protegido, protege la memoria para que un programa no pueda hacer caer al resto del sistema, para ello carga sólo las partes de un programa que se usan y comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria. Posee un funcionamiento muy rápido ya que es capaz de explotar todas las posibilidades de hardware del sistema, además es muy cómodo para el usuario ya que garantiza que este trabaje de manera segura por todas sus cualidades y características. El sistema será desarrollado sobre GNU/Linux,

ya que este es el Sistema Operativo que utilizan los proyectos productivos del polo HA, aunque el cliente puede acceder a la aplicación desde Windows.

1.4.2 Gestor de BD SQLite

SQLite es un sistema de gestión de base datos además de ser una biblioteca que almacena el esquema de la base de datos sobre un archivo. A diferencia de los sistemas de gestión de base datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción. (10)

Para el desarrollo de la solución se utilizará SQLite, ya que es el gestor de base datos utilizado en la herramienta Trac, esto proporciona más ventajas y facilidad en el desarrollo de la aplicación debido a que generalmente los plugins utilizan la misma BD del Trac para su control y manejo de datos. SQLite no necesita ser instalado, ni reiniciar o configurar un servidor. Esta cualidad permite que no haya un administrador de base de datos para realizar operaciones como crear tablas y vistas o asignar permisos.

1.4.3 Librería Genshi

Genshi Python es una librería que proporciona un conjunto integrado de componentes para analizar, generar y procesar HTML, XML u otro contenido de texto para la salida de la web. Esta librería proporciona un motor de plantillas que se pueden utilizar para la generación de HTML, XML o texto plano, de ahí su uso en la aplicación, con la que el cliente podrá interactuar a través de páginas HTML.

1.4.4 IDE de Programación Eclipse

El IDE Eclipse es una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta, en sus principios desarrollado por IBM, actualmente por la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Eclipse emplea módulos (*plugins*) para proporcionar todas sus funcionalidades, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software, adicionalmente para permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python. Eclipse también puede trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de Gestión de Base de Datos. (11)

Para el desarrollo del trabajo se utilizará Eclipse debido a que su integración de plugins lo hace muy rico en distintos lenguajes de programación que pueden ser utilizados en la implementación de proyectos. Eclipse es muy cómodo y amigable para el usuario a la hora de trabajar debido a numerosas características que lo hacen superior a otros IDE.

- Depura código que resida en una máquina remota.
- Avisa de los errores cometidos mediante una ventana secundaria.
- Modifica e inspecciona valores de variables.
- Compilación incremental de código.
- Editor visual con sintaxis coloreada.
- Editor de texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Control de versiones con CVS.
- Integración de plugins

1.4.5 Herramienta de Modelado UML: SDE NetBeans (12)

NetBeans es un entorno de desarrollo integrado desarrollado usando la Plataforma NetBeans. Constituye un proyecto de código abierto de gran éxito, que fue fundado en junio del 2000 por la compañía Sun MicroSystems que continúa siendo el patrocinador principal. SDE para NetBeans es una herramienta de modelado UML que soporta el ciclo de vida completo del desarrollo de software: análisis, diseño, implementación, pruebas y despliegue. Permite la captura de requisitos, el dibujo de diagramas UML, la realización de ingeniería inversa desde Java a UML y la generación de código Java. Esta herramienta fue usada en el modelado de la solución debido a que es libre, además de ser un requisito del cliente el uso de la misma para la creación de los diagramas pertinentes.

1.4.6 Metodología de Desarrollo de Software OpenUP

El OpenUP es un Proceso Unificado que aplica propuestas de gestión ágil como son desarrollo iterativo e incremental dentro del ciclo de vida, tratando de ser manejable en relación con el RUP, ya que mantiene sus características esenciales. OpenUP es un proceso de desarrollo de software completo ya que puede ser manifestado como todo el proceso para construir un sistema. Es extensible ya que en el proceso se puede agregar o adaptar según lo vayan requiriendo los sistemas. OpenUP es ágil, ligero y proporciona una comprensión detallada del proyecto, beneficiando a clientes y desarrolladores sobre productos a entregar y su formalidad. (13)

Su desarrollo es dirigido por casos de uso. OpenUP como metodología de desarrollo es conducida por el principio de colaboración para alinear intereses y para compartir su comprensión. Es un proceso para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias.

1.4.7 Lenguaje de Programación Python

Python es un lenguaje de programación interpretado, con tipado dinámico, multiplataforma y orientado a objetos**.** En la actualidad se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.

Python permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan entrada/salida de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK, Qt entre otros. (14)

Al ser un lenguaje interpretado ahorra un tiempo considerable en el desarrollo del programa, ya que no hay necesidad de compilar ni enlazar. El intérprete puede ser utilizado de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa.

El principal objetivo que persigue este lenguaje es la facilidad, tanto de lectura, como de diseño. Además permite varios estilos de programación: programación orientada a objetos, programación estructurada y programación funcional.

Para la solución se empleará este lenguaje debido a que la herramienta Trac, a la cual se le añadirá el módulo está implementada en Python. Es por ello que la decisión para la implementación del proyecto es dicho lenguaje.

1.4.8 Lenguaje de Programación Java Script

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. Con JavaScript se puede crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e

interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. (15)

Es un lenguaje interpretado, o sea, que no requiere compilación. Al igual que Java, JavaScript es un lenguaje orientado a objetos. En la implementación de la solución se utilizará JavaScript en las páginas clientes para la creación de efectos y validaciones de los campos de los formularios.

Conclusiones

A partir del estudio realizado se concluye que la Gestión de Requisitos es un proceso imprescindible en la industria de desarrollo de software, por lo que es necesario llevarlo a cabo con la calidad requerida. El contenido del presente capítulo estuvo enmarcado en su definición y tareas fundamentales, además de hacer mención de algunas herramientas utilizadas en la actualidad con el fin de automatizar las actividades que implica el mismo. También se ha realizado un estudio de diferentes herramientas y tecnologías que posibiliten el desarrollo de una solución que integrada a la herramienta Trac permita llevar a cabo algunas funcionalidades básicas de esta disciplina en los proyectos del polo productivo de Hardware y Automática.

Capítulo 2: Características del sistema

Introducción

En el presente capitulo se va obteniendo una visión más general del sistema a desarrollar. Además se muestran de forma precisa las características del sistema así como lo que debe hacer el mismo para dar solución a las necesidades del usuario.

2.1 Alcance de la Aplicación

Los proyectos productivos del polo de Hardware y Automática (HA) de la Facultad 5 de la UCI utilizan para la Gestión de Proyectos la herramienta Trac, ésta puede ser extendida mediante plugins para el enriquecimiento de funcionalidades que hagan de la misma una herramienta más completa. El polo tiene la necesidad de desarrollar una funcionalidad extendida al Trac que permita la automatización y puesta en práctica de algunas tareas propias de la disciplina Gestión de Requisitos. Las características de este gestor de proyectos condicionan el desarrollo de una solución que a partir de estudios realizados permita llevar a cabo diferentes tareas como adicionar, eliminar y modificar requisitos, definir vistas de matriz de trazabilidad, definir relaciones jerárquicas y obtener reportes de requisitos según los atributos seleccionados por el usuario. La mayor parte de estas funcionalidades están incluidas dentro de la Gestión de cambios, una de las tareas fundamentales de la Gestión de requisitos, donde se debe hacer un seguimiento de los cambios que ocurren en los requisitos como consecuencia de modificaciones propuestas y aprobadas al producto. Otras funcionalidades son implementadas ya que facilitarán el trabajo de las personas que utilicen la herramienta.

2.1.1 Requisitos funcionales

RF 1 Chequear que no exista un requisito en la BD con el mismo nombre.

RF 2 Dar valor al campo nombre del requisito.

RF 3 Adicionar requisito.

RF 4 Modificar requisito.

RF 5 Eliminar requisitos.

RF 6 Seleccionar los atributos por los cuales se desea obtener un reporte.

RF 7 Listar requisitos en dependencia de los atributos seleccionados.

RF 8 Seleccionar la vista que se desea obtener.

RF 9 Mostrar la vista de matriz de trazabilidad entre requisitos de tipos características y casos de uso.

RF 10 Mostrar la vista de matriz de trazabilidad entre requisitos de tipos necesidades de usuario y casos de uso.

RF 11 Mostrar la vista de matriz de trazabilidad entre requisitos de tipos características y no funcionales.

RF 12 Mostrar la vista de matriz de trazabilidad entre requisitos de tipos necesidades de usuario e historia de usuarios.

RF 13 Mostrar la vista de matriz de trazabilidad entre requisitos de tipos no funcionales e historia de usuarios.

RF 14 Agregar relación de trazabilidad.

RF 15 Eliminar relación de trazabilidad.

RF 16 Seleccionar el requisito del cual se quiere ver su lista de hijos.

RF 17 Mostrar hijos del requisito seleccionado.

2.1.2 Requisitos no funcionales

- **Usabilidad:** La aplicación de Gestión de Requisitos deberá visualizarse con gran calidad en diferentes navegadores web (Firefox, Opera) luego de ser instalada.

- **Confiabilidad:** El sistema debe mantener la calidad de los datos garantizando su integridad durante el procesamiento y almacenamiento de los mismos en la BD.
- **Funcionamiento:** Los tiempos de respuestas del sistema serán aproximadamente de 2 segundos.
- **Rendimiento:** El sistema debe funcionar en tiempo real.
- **Diseño e implementación:**

  Debe trabajar con la librería Genshi para la interfaz y llamar a dicha librería desde el lenguaje Python.

  Como IDE para programación se utilizará Eclipse.

  La herramienta de modelado UML utilizada será NetBeans.
- **Apariencia o interfaz externa:** El diseño del sistema estará basado en la interfaz existente del gestor de proyectos TRAC, sencillo, funcional y de fácil navegación para los usuarios.

2.2 Modelo de Casos de Uso

El modelo de Casos de Uso muestra de forma simple y efectiva los requisitos del sistema, ofreciendo una visión general de las principales funcionalidades del mismo.

En este epígrafe se mostrarán los actores y casos de uso del sistema con sus descripciones, además del diagrama de Casos de Uso del sistema.

2.2.1 Casos de Uso del sistema

1. Gestionar requisitos.
2. Obtener reporte.
3. Obtener vistas de matriz de trazabilidad.
4. Definir relaciones de trazabilidad.
5. Obtener vista jerárquica de requisitos.

2.2.2 Diagrama de Casos de Uso

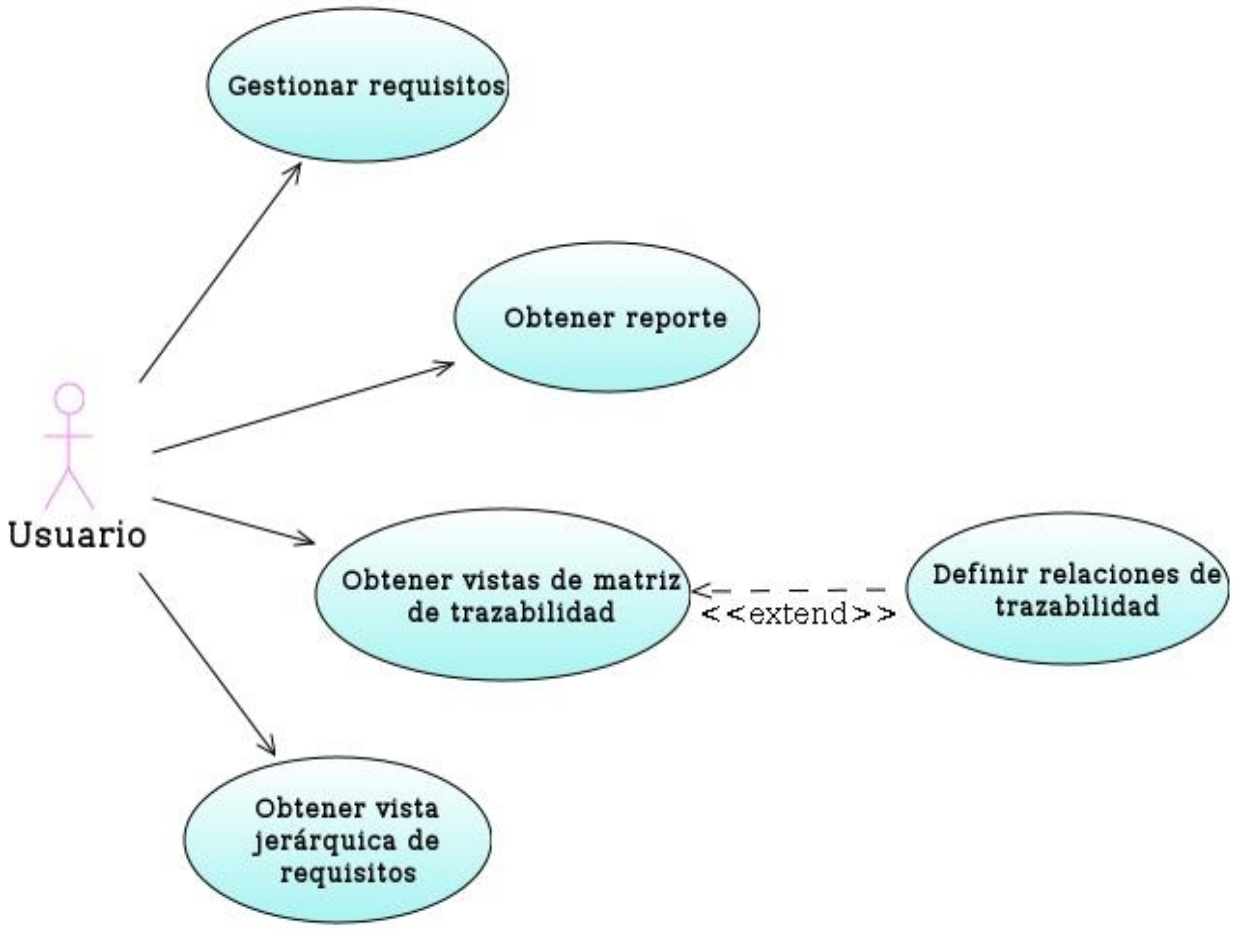


Figura 1 Diagrama de Casos de Uso

2.3 Descripción de los Casos de Uso del sistema

2.3.1 Caso de Uso: Gestionar requisitos

1 Breve Descripción

El caso de uso se inicia cuando el usuario accede al sistema con la intención de adicionar, modificar y eliminar requisitos.

2 Breve descripción de los Actores del sistema

Usuario: Estará en constante interacción con el sistema y tendrá los permisos necesarios para realizar cualquier operación que desee sobre la aplicación.

3 Precondiciones

3.1 El usuario debe estar autenticado.

4 Flujo Básico de Eventos

4.1 Gestionar Requisitos

4.1.1 El usuario selecciona la opción Gestionar Requisitos.

4.1.2 El sistema muestra tres opciones, dándole la oportunidad al usuario de Adicionar, Modificar y Eliminar requisitos.

- Si desea Adicionar requisitos, escoger la opción “Adicionar Requisitos”.
- Si desea Modificar requisitos, escoger la opción “Modificar Requisitos”.
- Si desea Eliminar requisitos, escoger la opción “Eliminar Requisitos”.

5 Escenarios clave

5.1 Escenario 1: Adicionar Requisitos

5.1.1 El sistema muestra un formulario con los campos a llenar del requisito: nombre, iteración, estado, origen, tipo, prioridad, dificultad, estabilidad, descripción, padre, hijos.

5.1.2 El usuario entra los datos correspondientes a cada unos de los campos del requisito y presiona el botón “Adicionar”.

5.1.3 El sistema verifica que el campo nombre no esté vacío.

5.1.4 El sistema verifica que el nombre entrado no existe en la Base de Datos.

5.1.5 En caso de escoger padre e hijos para el requisito el sistema verifica que no se escoja un mismo requisito como padre e hijo.

5.1.6 El sistema adiciona el requisito correctamente en la Base de Datos.

5.2 Escenario 2: Modificar Requisitos

5.2.1 El sistema muestra una tabla con todos los requisitos existentes en la Base de Datos.

5.2.2 El usuario escoge el requisito que desea modificar y presiona el botón “Modificar”.

5.2.3 El sistema muestra un formulario con todos los campos y valores del requisito escogido.

5.2.4 El usuario entra el nuevo valor de los campos deseados y presiona el botón “Modificar”.

5.2.5 En caso de modificar el nombre del requisito, el sistema verifica que no exista ese nombre en la Base de Datos.

5.2.6 En caso de modificarle los hijos al requisito, el sistema verifica que dentro de los hijos asignados no esté él mismo.

5.2.7 En caso de modificarle el padre al requisito, el sistema verifica que no se asigne como padre a él mismo.

5.2.8 En caso de modificarle el padre y los hijos al requisito, el sistema verifica que no se escoja dentro de los hijos el padre asignado.

5.2.9 El sistema modifica el requisito correctamente y actualiza los nuevos valores en la Base de Datos.

5.2.10 El sistema muestra la tabla actualizada.

5.3 Escenario 3: Eliminar Requisitos

5.3.1 El sistema muestra una tabla con todos los requisitos existentes en la Base de Datos.

5.3.2 El usuario escoge los requisitos que desea eliminar y presiona el botón “Eliminar”.

5.3.3 El sistema verifica que se haya seleccionado algún requisito.

5.3.4 El sistema elimina los requisitos correctamente y actualiza la Base de Datos.

5.3.5 El sistema muestra la tabla actualizada.

6 Flujos Alternos

6.1 Escenario 1: Adicionar Requisitos

5.1.3 El sistema muestra un mensaje notificando que debe entrarle un nombre al requisito.

5.1.4 El sistema muestra un mensaje notificando que ya existe ese nombre en la Base de Datos.

5.1.5 El sistema muestra un mensaje notificando que el padre asignado está dentro de los hijos asignados.

6.2 Escenario 2: Modificar Requisitos

5.2.5 El sistema muestra un mensaje notificando que ya existe ese nombre en la Base de Datos.

5.2.6 El sistema muestra un mensaje notificando que no puede asignarse a él mismo como su hijo.

5.2.7 El sistema muestra un mensaje notificando que no puede asignarse a él mismo como su padre.

5.2.8 El sistema muestra un mensaje notificando que no puede asignarle como hijo el padre asignado.

6.3 Escenario 3: Eliminar Requisitos

5.3.3 El sistema muestra un mensaje indicando que debe seleccionar al menos un requisito.

7 Poscondiciones

7.1 Requisito adicionado correctamente.

7.2 Requisito modificado correctamente.

7.3 Requisito eliminado correctamente.

2.3.2 Caso de Uso: Obtener reporte

1 Breve Descripción

El caso de uso se inicia cuando el usuario accede al sistema con la intención de obtener algún reporte de requisitos según los atributos seleccionados por el mismo.

2 Breve descripción de los Actores del sistema

Usuario: Estará en constante interacción con el sistema y tendrá los permisos necesarios para realizar cualquier operación que desee sobre la aplicación.

3 Precondiciones

3.1 El usuario debe estar autenticado.

4 Flujo Básico de Eventos

4.1 Obtener Reporte de Requisitos por Atributos

4.1.1 El usuario selecciona la opción “Obtener Reporte por Atributos”.

4.1.2 El sistema muestra un formulario con los atributos del requisito y una tabla con los requisitos existentes en la Base de Datos.

4.1.3 El usuario selecciona los atributos por los cuales desea obtener el reporte y presiona el botón "Ver Reporte".

4.1.4 El sistema verifica que se haya escogido al menos un atributo.

4.1.5 El sistema muestra el reporte.

5 Flujos Alternos

5.1 Obtener Reporte de Requisitos por Atributos

4.1.4 El sistema muestra un mensaje notificando que debe escoger al menos un atributo.

6 Poscondiciones

6.1 Reporte obtenido correctamente según los atributos seleccionados.

2.3.3 Caso de Uso: Obtener vistas de matriz de trazabilidad

1 Breve Descripción

El caso de uso se inicia cuando el actor accede al sistema con la intención de obtener alguna vista de matriz de trazabilidad entre requisitos.

2 Breve descripción de los Actores del sistema

Usuario: Estará en constante interacción con el sistema y tendrá los permisos necesarios para realizar cualquier operación que desee sobre la aplicación.

3 Precondiciones

3.1 El usuario debe estar autenticado.

3.2 Deben existir el tipo de requisitos de la matriz seleccionada.

4 Flujo Básico de Eventos

4.1 Obtener Vistas de Matriz de Trazabilidad

4.1.1 El usuario selecciona la opción "Vistas de Matriz de Trazabilidad".

4.1.2 El sistema muestra cinco opciones dándole la oportunidad al usuario de obtener las matrices de trazabilidad definidas: Características – Casos de Uso, Necesidades de Usuario – Casos de Uso, No Funcional – Historia de Usuarios, Necesidades de Usuario – Historia de Usuarios y Características – No Funcional.

- Si se desea obtener la matriz de trazabilidad entre requisitos Características y Casos de Uso, escoger opción "Características – Casos de Uso".

- Si se desea obtener la matriz de trazabilidad entre requisitos Necesidades de Usuario y Casos de Uso, escoger opción "Necesidades de Usuario – Casos de Uso".

- Si se desea obtener la matriz de trazabilidad entre requisitos No Funcionales e Historia de Usuarios, escoger opción "No Funcional – Historia de Usuarios".

- Si se desea obtener la matriz de trazabilidad entre requisitos Necesidades de Usuario e Historia de Usuarios, escoger opción "Necesidades de Usuario – Historia de Usuarios".

- Si se desea obtener la matriz de trazabilidad entre requisitos Características y No Funcionales, escoger opción "Características – No Funcional".

5 Escenarios clave

5.1 Escenario 1: Características – Casos de Uso

5.1.1 El sistema verifica que existan requisitos características y casos de uso en la Base de Datos.

5.1.2 El sistema muestra una tabla con los requisitos características y casos de uso y la matriz de trazabilidad con las relaciones existentes.

6 Flujos Alternos

6.1 Escenario 1: Características – Casos de Uso

5.1.1 El sistema muestra un mensaje notificando que tipo de requisito no se encontró en la Base de datos.

6 Poscondiciones

1. La vista de la matriz de trazabilidad entre requisitos de tipo necesidades de usuario y casos de uso fue obtenida satisfactoriamente.

Observación: *Este caso de uso cuenta además con los escenarios clave Necesidades de Usuario – Casos de Uso, No Funcional – Historia de Usuarios, Necesidades de Usuario – Historia de Usuarios y Características – No Funcional, que no se reflejan en las descripciones por la similitud de pasos con el escenario especificado.*

2.3.4 Caso de Uso: Definir relaciones de trazabilidad

1 Breve Descripción

El caso de uso se inicia cuando el actor accede al sistema con la intención de adicionar o eliminar alguna relación de trazabilidad entre requisitos en las vistas de matriz de trazabilidad existentes. El caso de uso termina cuando la relación es adicionada ó eliminada satisfactoriamente.

2 Breve descripción de los Actores del sistema

Usuario: El actor del sistema es un usuario que estará en constante interacción con el sistema y tendrá los permisos necesarios para realizar cualquier operación que desee sobre la aplicación.

3 Precondiciones

1. El usuario debe estar autenticado.
2. Deben existir los requisitos especificados.

4 Flujo Básico de Eventos

4.1 Definir relaciones de trazabilidad

4.1.1 El flujo básico de eventos transcurre de forma similar al flujo básico del caso de uso Obtener Vistas de Matriz de Trazabilidad, a continuación el usuario podrá agregar nuevas relaciones de trazabilidad o eliminar algunas existentes.

5 Escenarios clave

5.1 Escenario 1: Agregar relación de trazabilidad Características – Casos de Uso

5.1.1 El usuario selecciona los identificadores de los requisitos que desea relacionar y presiona el botón “Adicionar”.

5.1.2 El sistema verifica que se haya seleccionado requisitos de ambos tipos.

5.1.3 El sistema muestra la matriz de trazabilidad entre Características y Casos de Uso actualizada.

5.2 Escenario 2: Eliminar relación de trazabilidad Características – Casos de Uso

5.2.1 El usuario selecciona los identificadores de los requisitos que desea relacionar y presiona el botón “Eliminar”.

5.2.2 El sistema verifica que se haya seleccionado requisitos de ambos tipos.

5.2.3 El sistema muestra la matriz de trazabilidad entre Características y Casos de Uso actualizada.

6 Flujos Alternos

6.1 Escenario 1: Agregar relación de trazabilidad Características – Casos de Uso

5.1.2 El sistema muestra un mensaje notificando que se debe seleccionar requisitos de ambos tipos e indica el tipo de requisito que faltó por escoger.

6.2 Escenario 2: Eliminar relación de trazabilidad Características – Casos de Uso

5.2.2 El sistema muestra un mensaje notificando que se debe seleccionar requisitos de ambos tipos e indica el tipo de requisito que faltó por escoger.

7 Poscondiciones

1. La relación de trazabilidad fue adicionada correctamente.
2. La relación de trazabilidad fue eliminada correctamente.

Observación: *También se puede definir relaciones de trazabilidad para las vistas entre Necesidades de Usuario – Casos de Uso, No Funcional – Historia de Usuarios, Necesidades de Usuario – Historia de Usuarios y Características – No Funcional, que no se describen en el documento por la similitud con los escenarios especificados anteriormente.*

2.3.5 Caso de Uso: Obtener vista jerárquica de requisitos

1 Breve Descripción

El caso de uso se inicia cuando el actor accede al sistema con la intención de obtener las vistas jerárquicas de los requisitos existentes en la Base Datos.

2 Breve descripción de los Actores del sistema

Usuario: Estará en constante interacción con el sistema y tendrá los permisos necesarios para realizar cualquier operación que desee sobre la aplicación.

3 Precondiciones

3.1 El usuario debe estar autenticado.

4 Flujo Básico de Eventos

4.1 Obtener Vista Jerárquica de Requisitos

4.1.1 El usuario selecciona la opción “Vista Jerárquica de Requisitos”.

4.1.2 El sistema muestra una tabla con el identificador y nombre de los requisitos padres existentes.

4.1.3 El usuario selecciona el requisito padre al cual le desea ver los requisitos hijos y se presiona el botón “Ver Hijos”.

4.1.4 El sistema verifica que se haya seleccionado un requisito padre.

4.1.5 El sistema muestra una tabla jerárquica con el identificador del requisito padre seleccionado y los requisitos hijos.

5 Flujos Alternos

5.1 Obtener Vista Jerárquica de Requisitos

4.1.4 El sistema muestra un mensaje notificando que no se seleccionó ningún requisito padre.

6 Poscondiciones

6.1 Vista jerárquica de requisitos obtenida satisfactoriamente.

Conclusiones

En este capítulo se define el alcance de la solución, dando cumplimiento a uno de los objetivos específicos de la investigación. A partir de los requisitos del sistema se determinan las funcionalidades con que debe contar. También se definieron los actores del sistema y los casos de uso con sus respectivas descripciones y características.

Capítulo 3: Diseño e implementación del sistema

Introducción

En este capítulo se describe el diseño del sistema, el diagrama de clases de diseño con sus respectivas relaciones de generalización/especialización, los diagramas de secuencia para los casos de uso que intervendrán en el primer ciclo de desarrollo de la solución así como la arquitectura utilizada en el diseño del sistema.

3.1 Modelo de Diseño

Es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto a otras restricciones del entorno de implementación tienen su impacto en el sistema que se desarrolla. Además está conformado por los artefactos diagrama de clases de diseño y diagrama de secuencia por cada caso de uso del sistema, estos diagramas se presentarán en los siguientes epígrafes.

3.2 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular. (16) Estos identifican las Clases, Instancias y la distribución de responsabilidades. Un patrón de diseño es una solución estándar para un problema común de programación, una técnica para flexibilizar el código haciéndolo satisfacer diversos criterios, un lenguaje de programación de alto nivel, una manera más práctica de describir ciertos aspectos de la organización de un programa.

3.2.1 Patrón Fachada (Facade)

El patrón de diseño fachada provee una interfaz unificada que actúa como intermediaria entre un cliente y una o un grupo de interfaces. Con este patrón se brinda un acceso sencillo y se desacopla al máximo el sistema cliente (el que accede a la fachada) de los sistemas ocultos.

Típicamente tiene gran uso en sistemas diseñados en capas y en librerías. Si se está implementando una cantidad considerable de clases la mejor solución para mantener el bajo acoplamiento y claridad en el código es la utilización del patrón fachada.

Algunas ventajas que disfrutaremos en el propio desarrollo son:

- Facilitamos la utilización y comprensión (acompañando la interfaz con una documentación mínima) de los sistemas ocultados
- Los clientes se olvidan de toda la complejidad del negocio, sólo les importa los resultados obtenidos.

En la solución el patrón Fachada se evidencia en la clase Data_Access, que proporciona una fachada que permite a las diferentes clases controladoras acceder mediante esta a las entidades ocultas (Tablas de la BD).

3.3 Arquitectura en 3 capas

La Arquitectura en tres capas se divide en tres partes con un reparto claro de funciones: una capa para la presentación, otra para el cálculo (capa de negocio) y otra para el almacenamiento (capa de datos).

- Capa de presentación: Está compuesta por las interfaces de usuario, (formularios windows, páginas HTML) y sus controles visuales (textBox, comboBox, dataGrids) junto con sus eventos (clic y etc.)
- Capa de negocio (lógica del dominio): Aquí irá todo el código que define las reglas de negocio (cálculos, validaciones).
- Capa de acceso a datos: Tendrá el código que permite acceder a las fuentes de datos. Esencialmente trata sobre 4 operaciones básicas, llamadas CRUD (por Create-Retrieve-Update y

Delete), que se realizan sobre cualquier fuente de datos (normalmente alguna base de datos relacional).

Las capas de presentación, de negocio y de datos pueden residir en un único servidor (no sería lo normal), si bien lo más usual es que haya una multitud de servidores donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor).

Las capas de negocio y de datos pueden residir en el mismo servidor, como es el caso de la solución propuesta **(Ver figura 2)**. En estos casos, si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más servidores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios servidores, los cuales recibirán las peticiones del servidor en que resida la capa de negocio. Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más servidores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de servidores sobre los cuales corre la capa de datos.

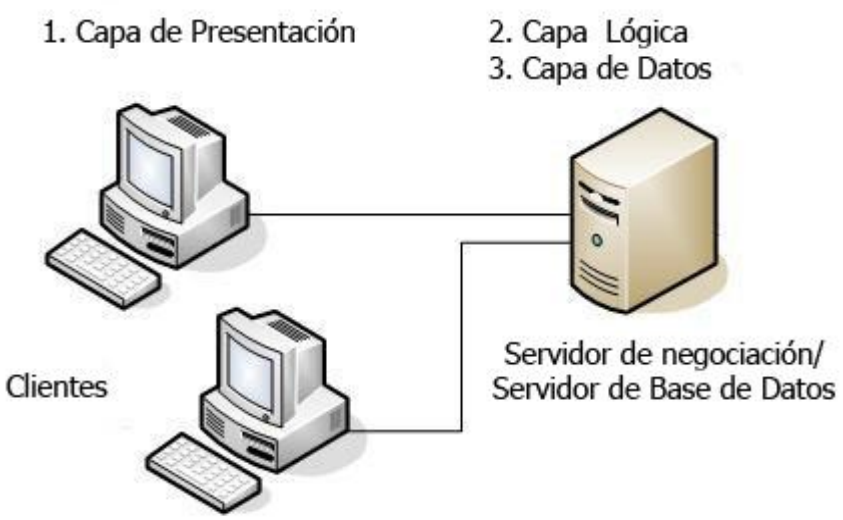


Figura 2 Arquitectura en 3 capas

3.4 Arquitectura de la herramienta Trac (9)

El Trac está compuesto por una serie de paquetes que conforman su arquitectura. El principal paquete es el núcleo (trac.core) que implementa solo los componentes necesarios que facilitaran ampliar las funcionalidades de otros componentes que se adicionen al sistema.

Se define un componente como un objeto que proporciona un determinado tipo de servicio en el contexto de la aplicación. Hay al menos una instancia de cualquiera de los componentes y utilizan el patrón de arquitectura instancia única (singleton), lo que implica que no sea referenciado directamente a una entidad de la aplicación del modelo de objetos, sino que representen las funcionalidades de los subsistemas.

Los componentes pueden declarar "la ampliación de puntos" para que otros puedan "conectarse" a ellos y así poder mejorar la funcionalidad extendida. Todo lo que se necesita es que el componente original exponga uno o más puntos de extensión.

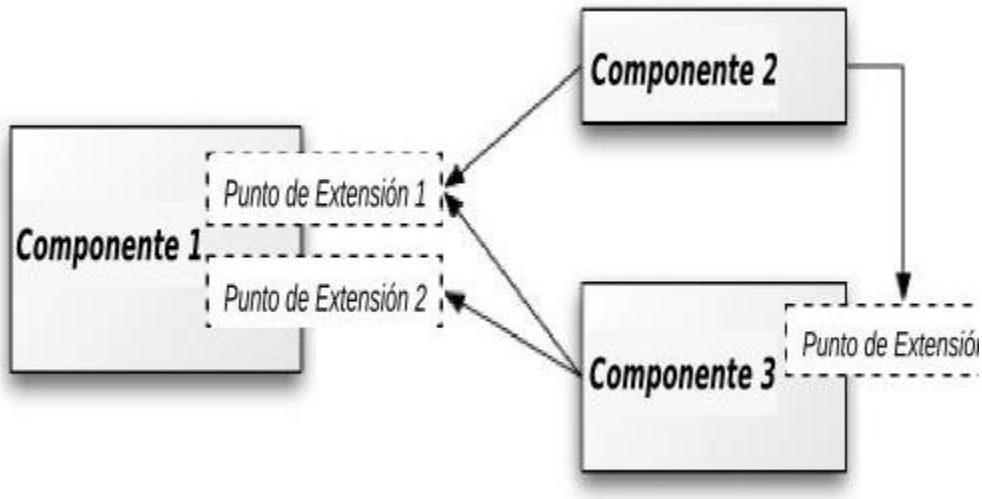


Figura 3: Ampliación de puntos de los componentes

Un componente puede extender a cualquier número de otros componentes y ofrecer sus propios puntos de extensión. Esta característica es la base de una arquitectura basada en extensiones.

Los plugins de Trac utilizan para su desarrollo esta arquitectura en la implementación de interfaces que son necesarias para su funcionamiento. Es por ello que el plugin de Gestión de Requisitos para su desarrollo implementará las interfaces INavigationContributor, IRequestHandler, ITemplateProvider. Estas poseen comportamientos específicos que son necesarios para dar solución al problema propuesto.

- INavigationContributor: Utilizada para su ubicación en la barra de navegación
- IRequestHandler: Utilizada para el tratamiento de las peticiones HTTP.
- ITemplateProvider: Utilizada para la utilización de documentos estáticos, como plantillas.

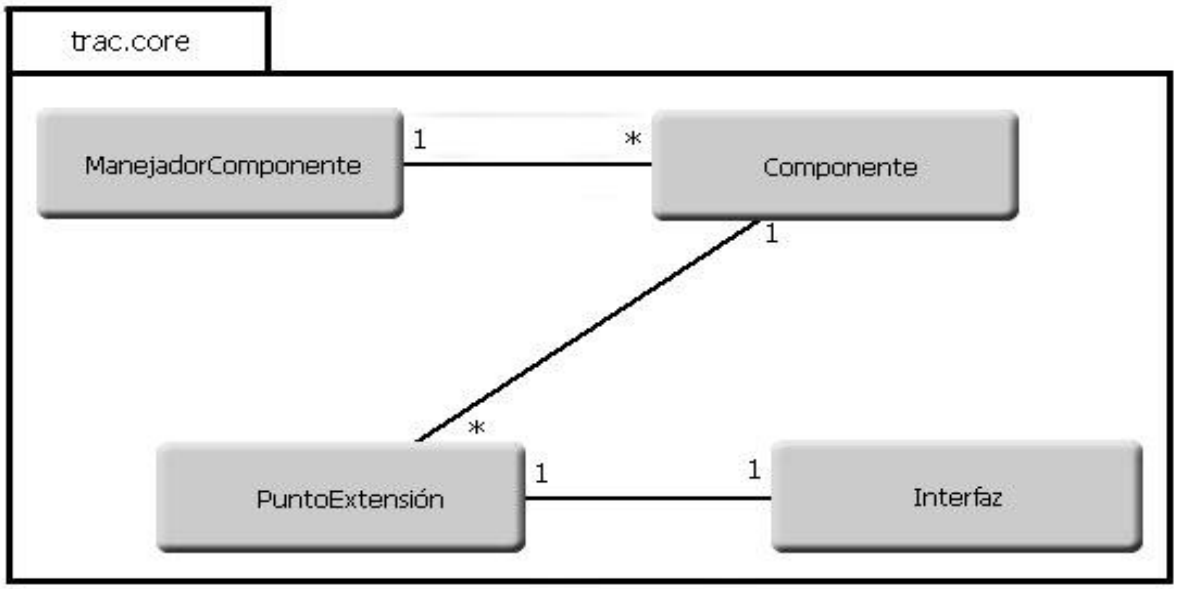


Figura 4 Arquitectura de la herramienta Trac

3.5 Diagrama de Clases de Diseño

En el presente epígrafe se representa el diagrama de clases de diseño en este primer ciclo de desarrollo de la solución del trabajo. El mismo está organizado según la arquitectura de 3 capas.

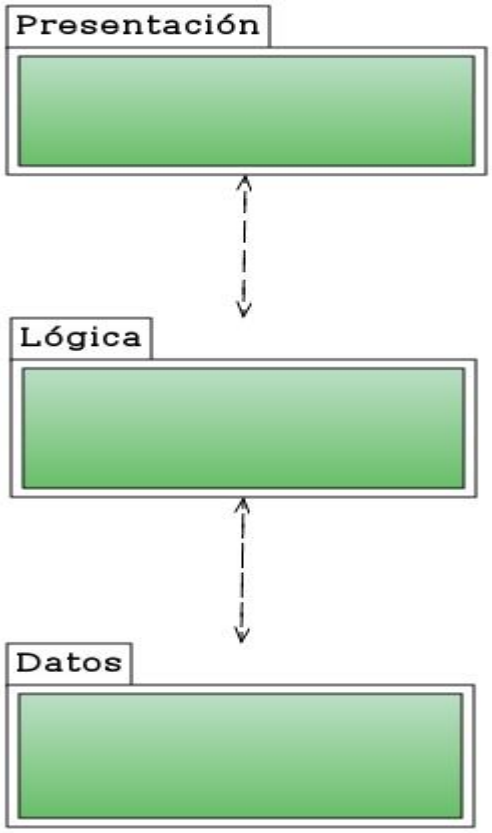


Figura 5 Diagrama de Paquetes de Clases de Diseño.

3.5.1 Paquete Capa de Presentación

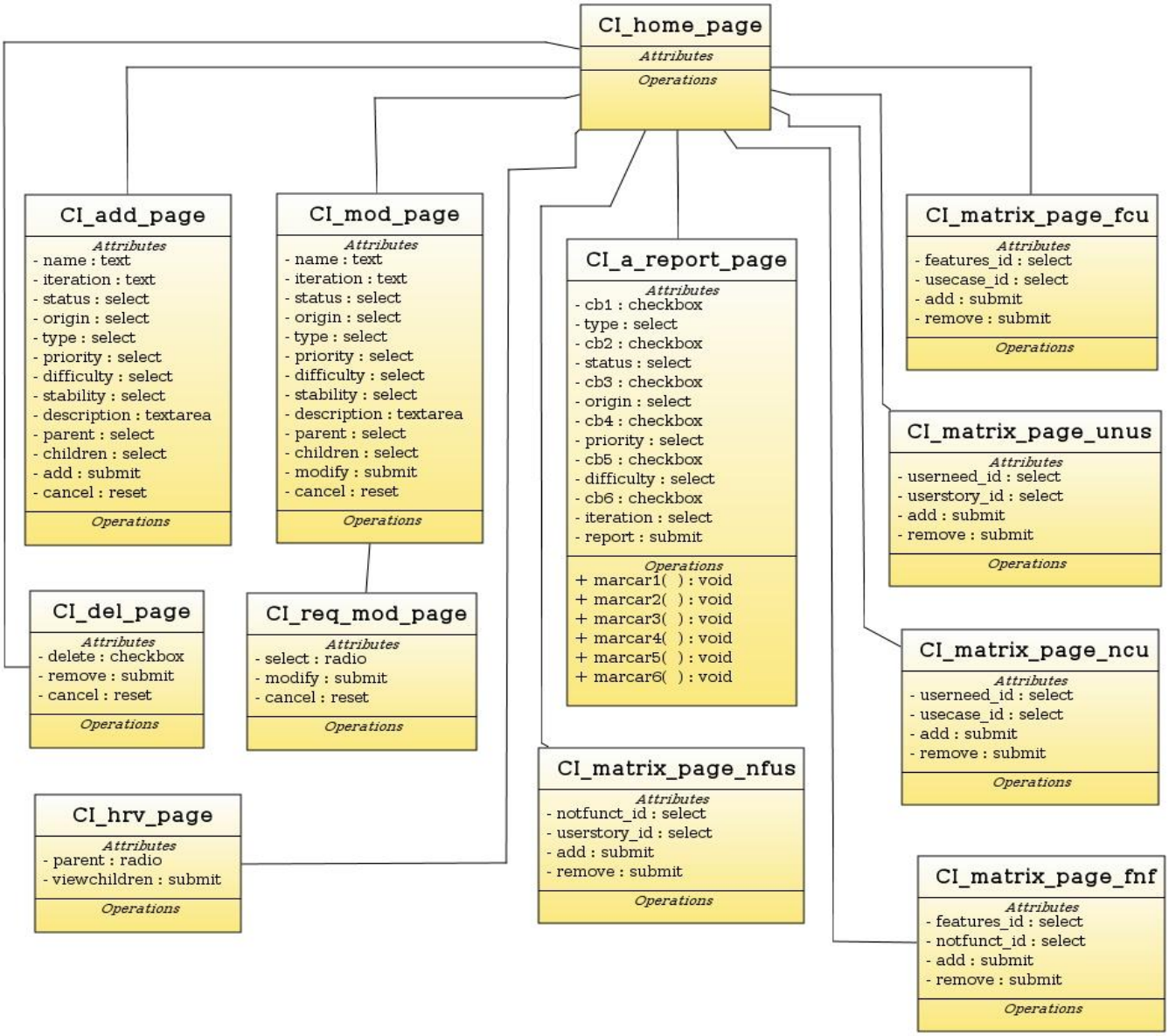


Figura 6 Diagrama de Clases de la *Capa de Presentación*

3.5.2 Paquete Capa de Negocio

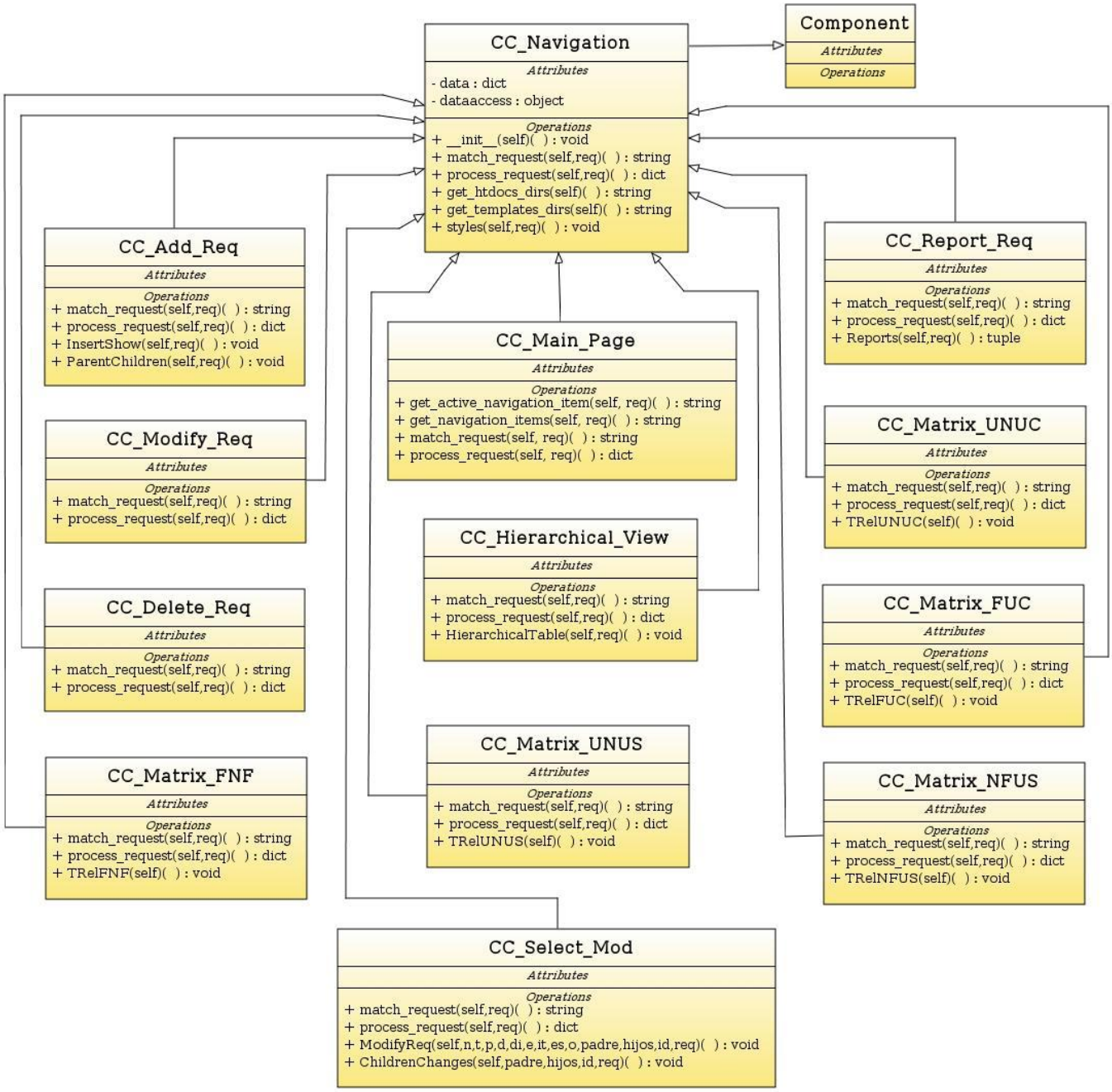


Figura 7 Diagrama de Clases de la *Capa de Negocio*

3.5.3 Paquete Capa de Datos

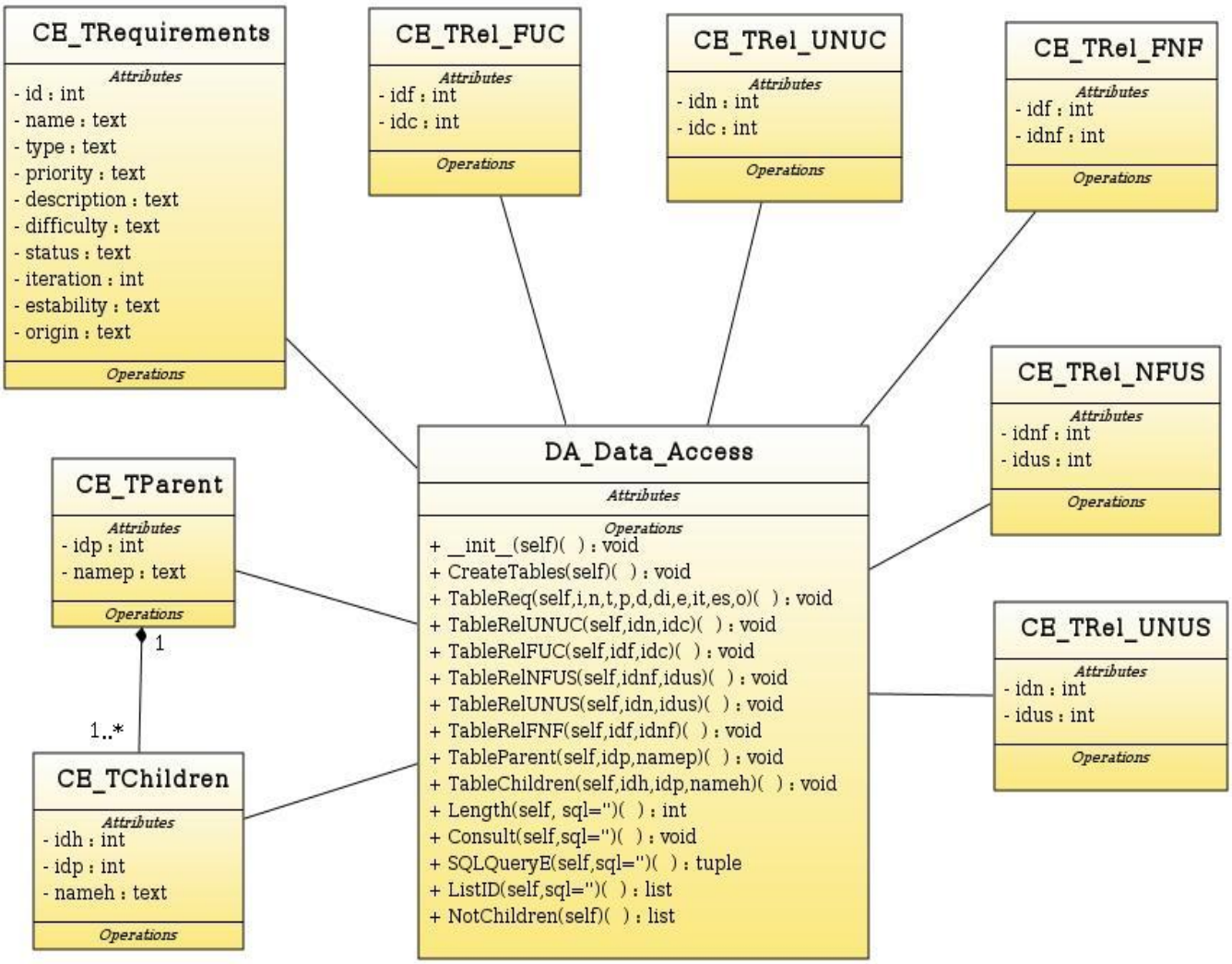
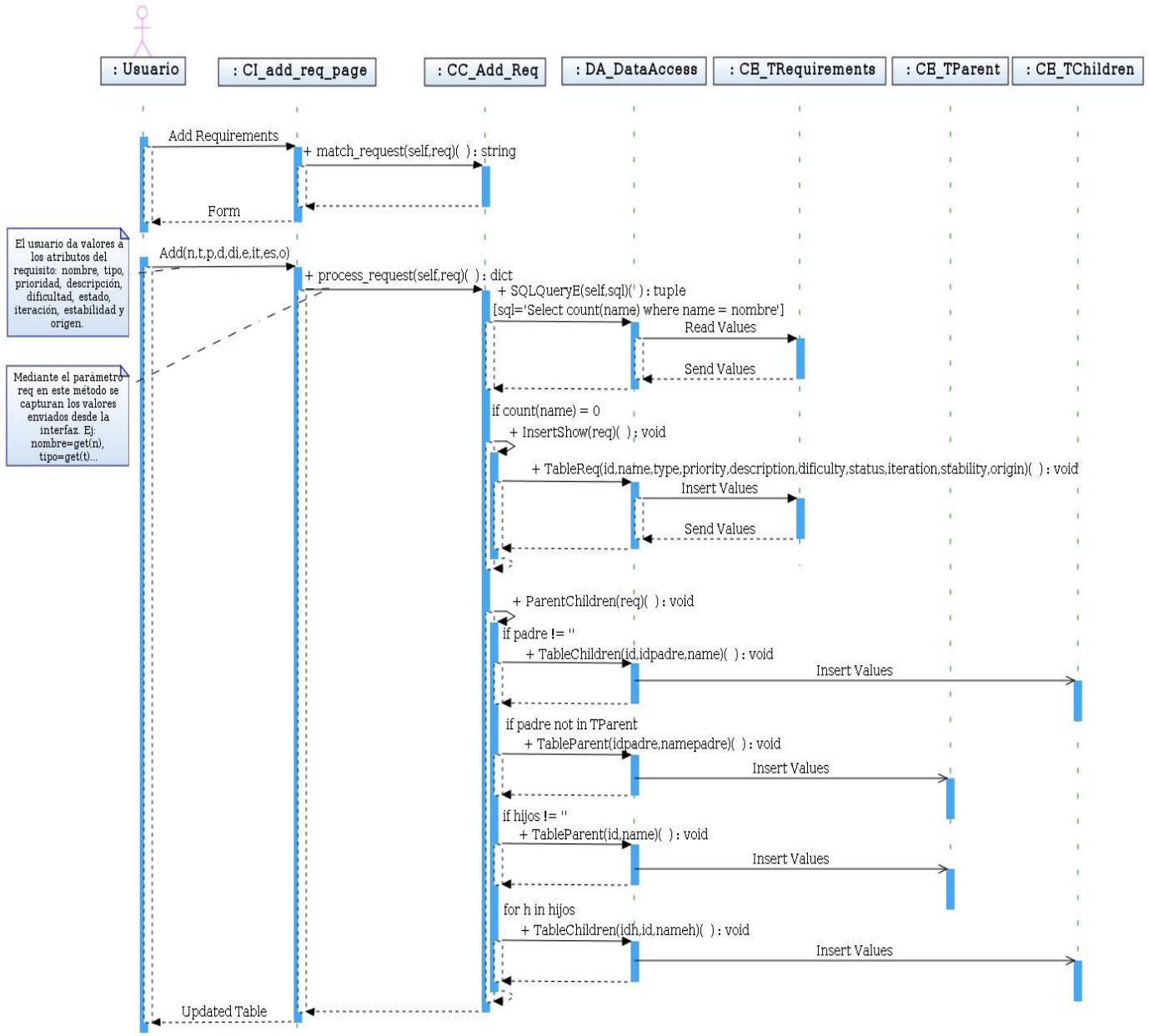


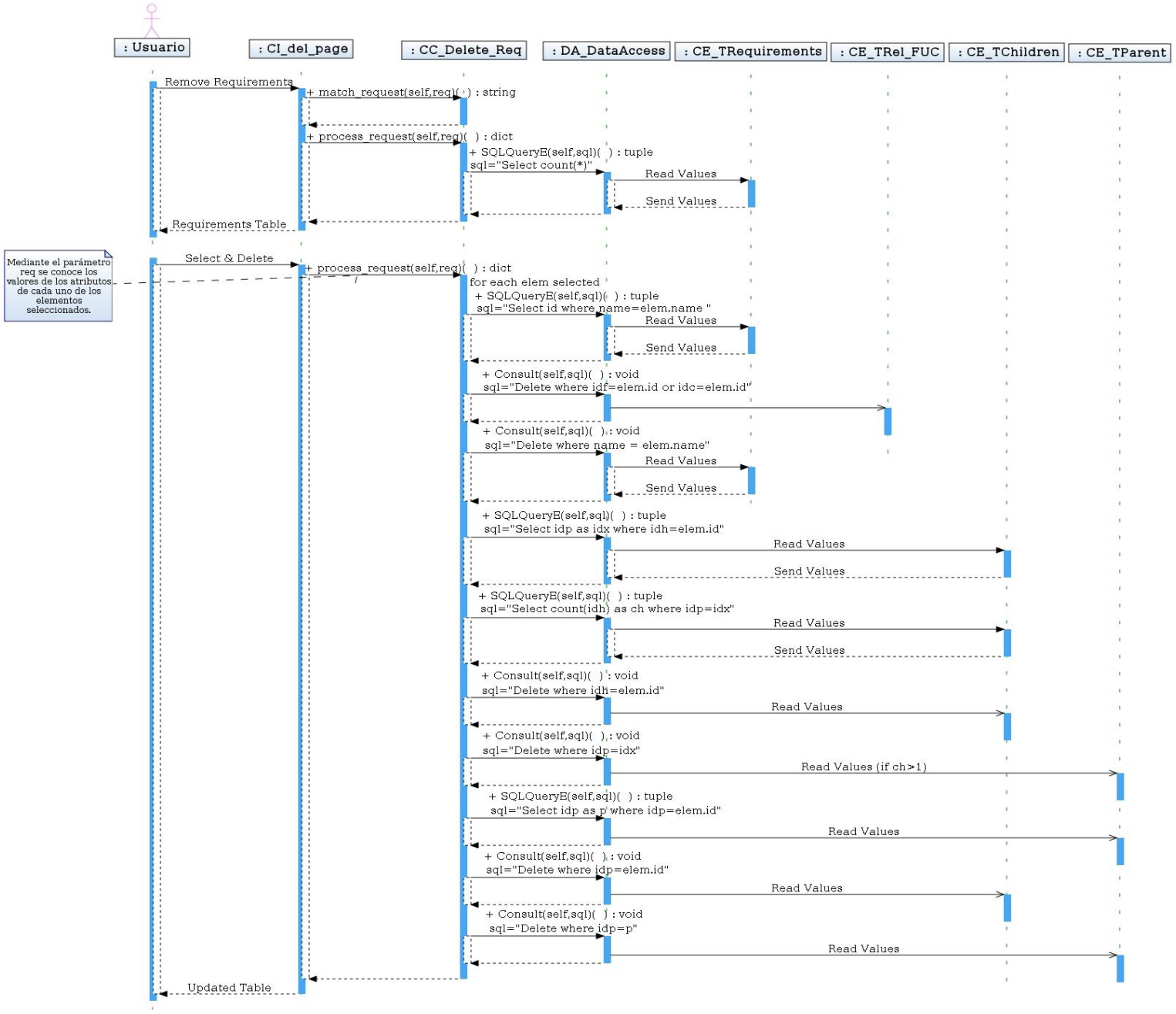
Figura 8 Diagrama de Clases de la *Capa de Datos*

3.6 Diagramas de Secuencia

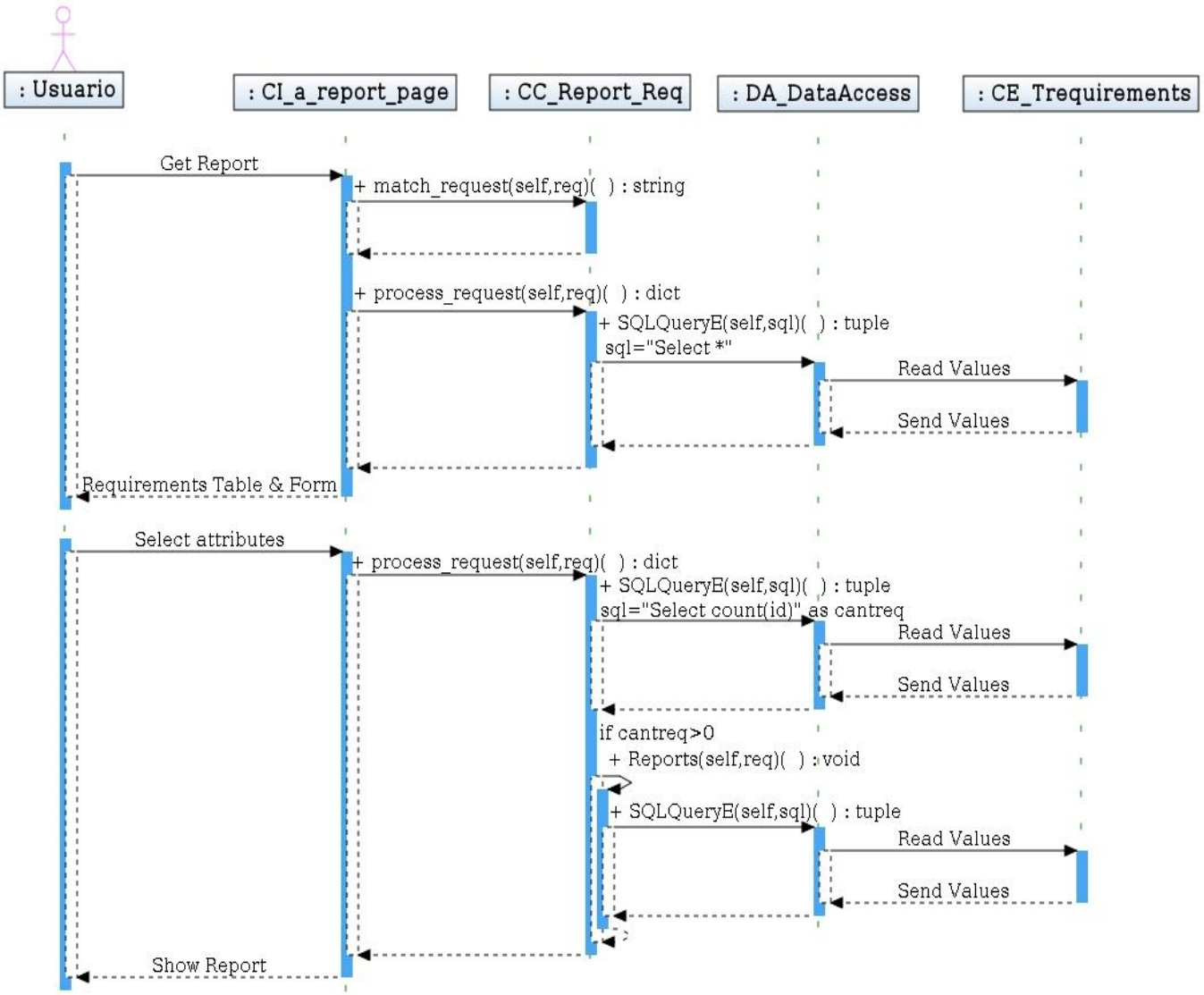
3.6.1 Diagrama de Secuencia CU Gestionar requisitos (Escenario Adicionar requisito)



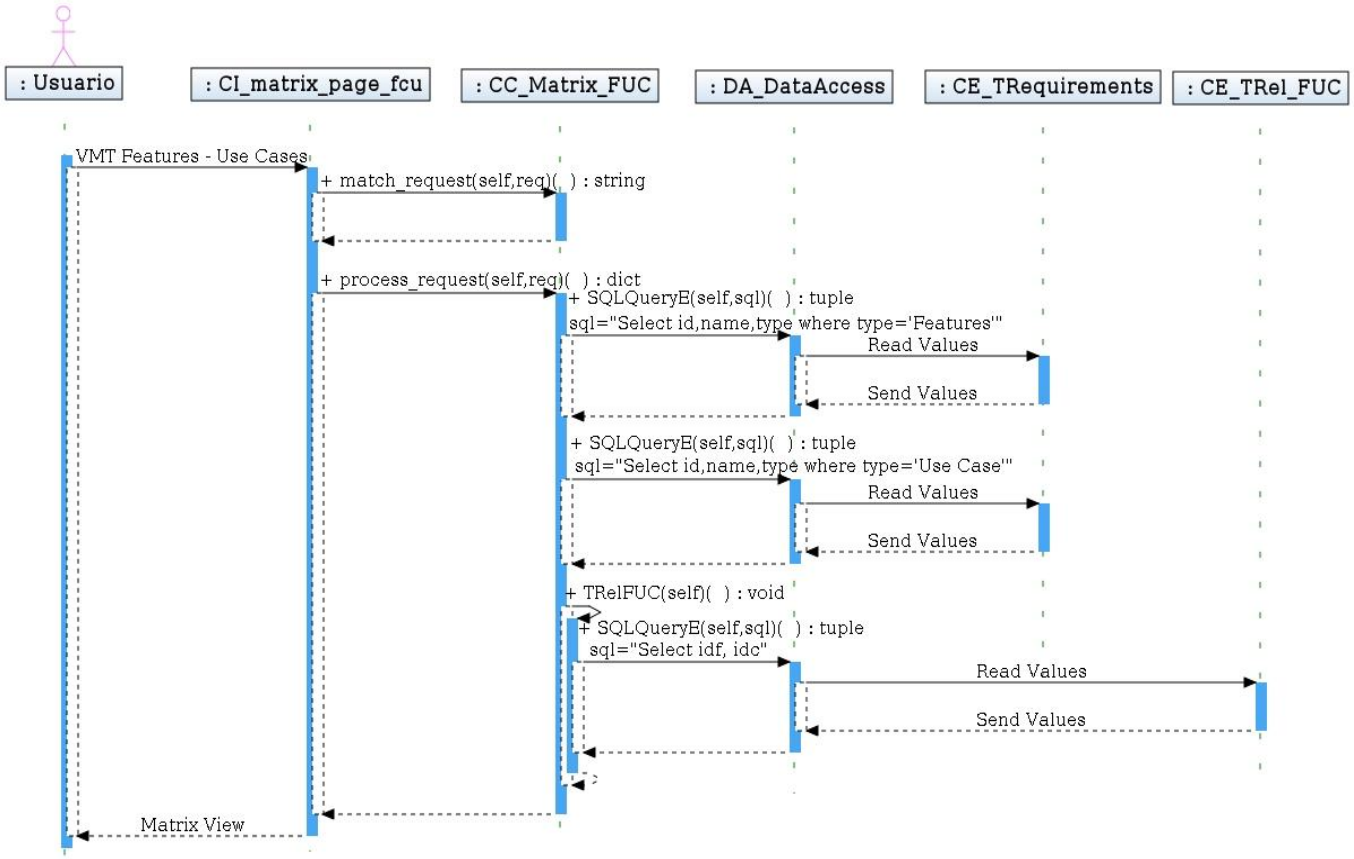
3.6.2 Diagrama de Secuencia CU Gestionar requisitos (Escenario Eliminar requisitos)



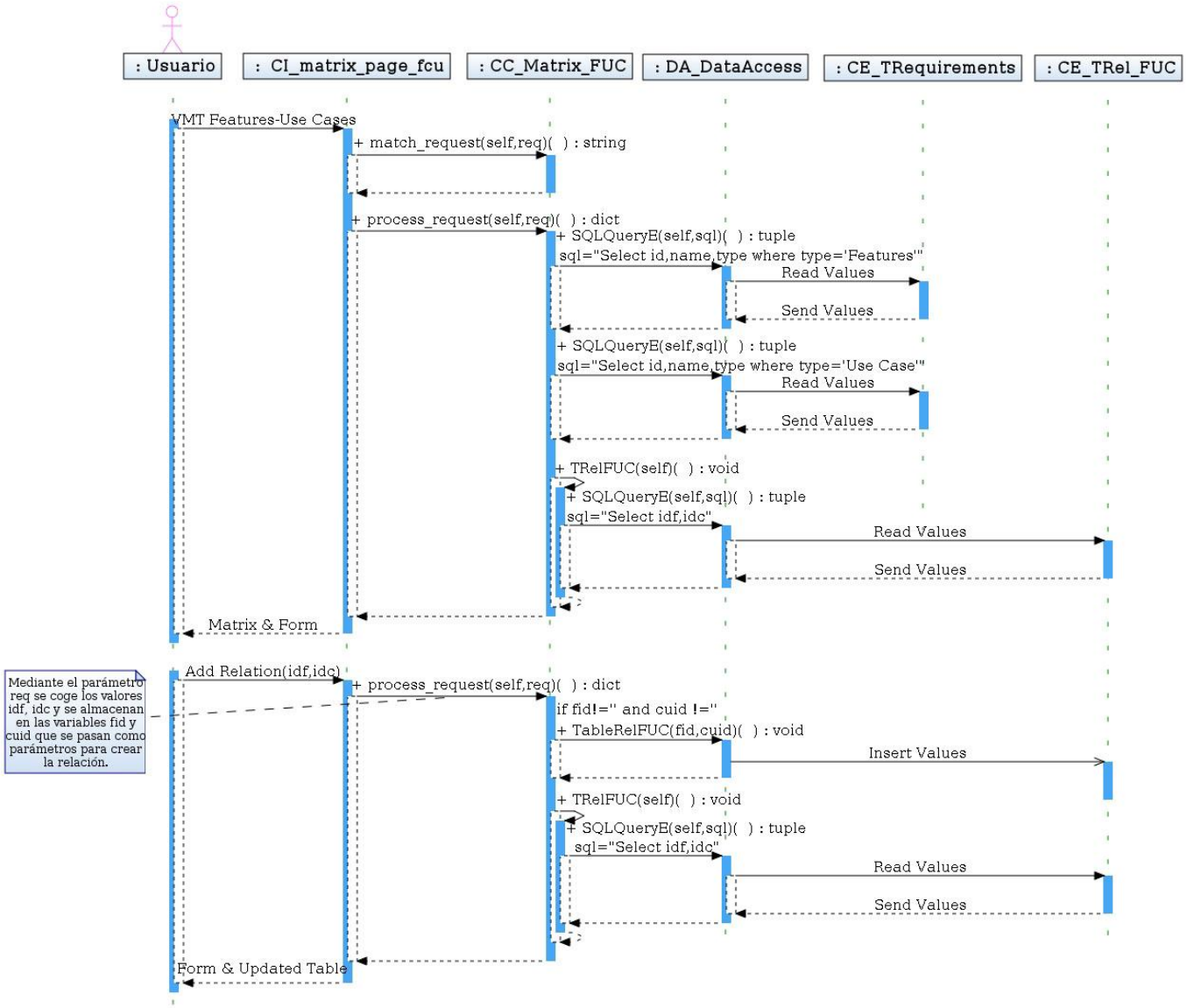
3.6.3 Diagrama de Secuencia CU Obtener reporte



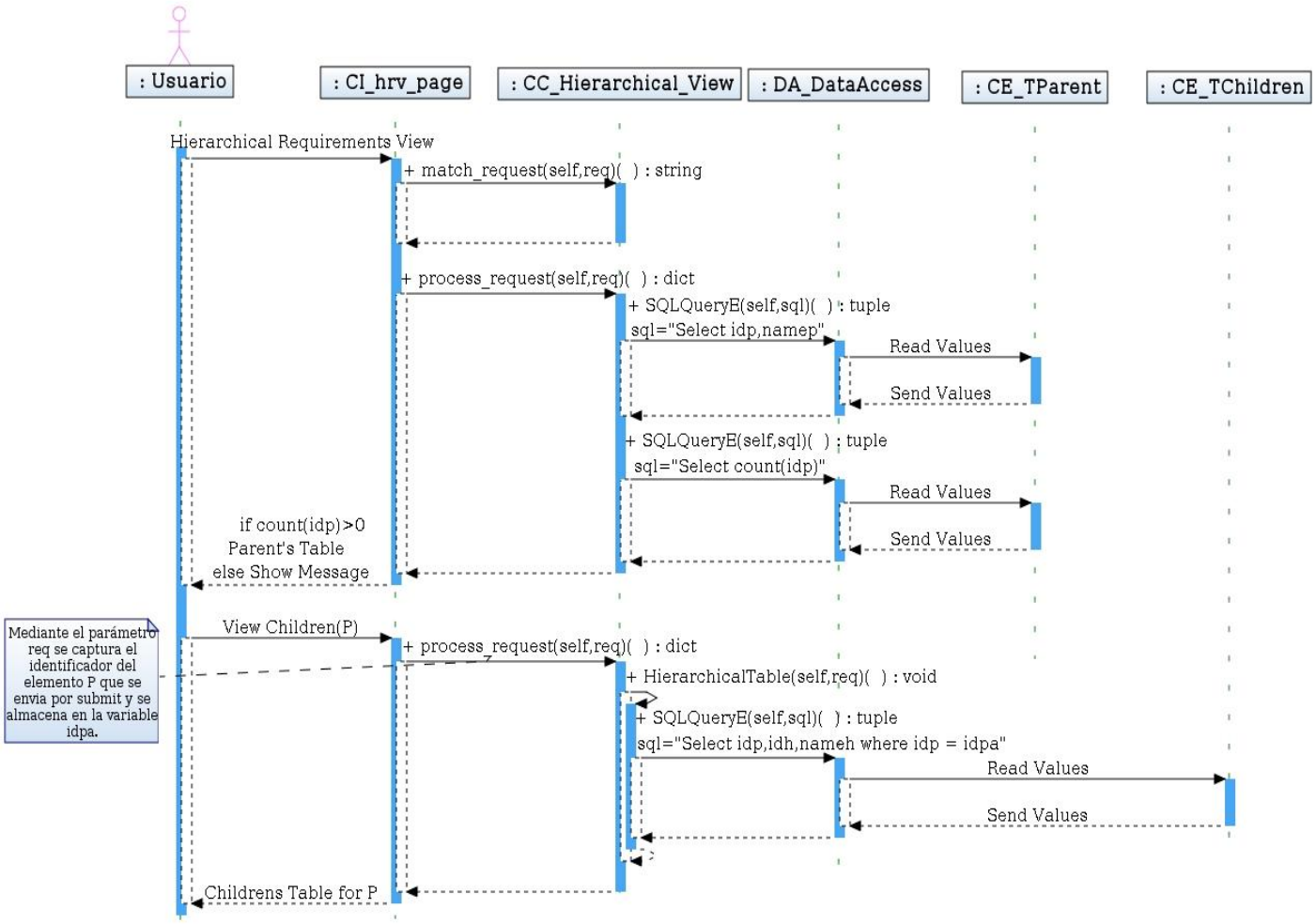
3.6.4 Diagrama de Secuencia CU Obtener vistas de matriz de trazabilidad (Escenario Vista de matriz de trazabilidad entre características y casos de uso)



3.6.5 Diagrama de Secuencia CU Definir relaciones de trazabilidad (Escenario Adicionar relación de trazabilidad) para el escenario Vista de matriz de trazabilidad entre características y casos de uso del CU Obtener vistas de matriz de trazabilidad



3.6.6 Diagrama de Secuencia CU Obtener vista jerárquica de requisitos



Observación: *No se realizaron diagramas de secuencia para todos los escenarios por la similitud con los representados en este epígrafe.* (Ver Anexo 1)

3.7 Descripción de las clases

Tabla 2 Clase "TRequirements"

Nombre: TRequirements	
Descripción: Clase encargada de almacenar los datos de los requisitos y tener persistencia de los mismos.	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	int
name	text
type	text
priority	text
description	text
dificulty	text
status	text
iteration	int
stability	text
origin	text

Tabla 3 Clase "TRel_UNUC"

Nombre: TRel_UNUC	
Descripción: Clase encargada de almacenar los identificadores de los requisitos de tipos **necesidades de usuario** y **casos de uso** que poseen relaciones de trazabilidad.	
Tipo de clase: Entidad	
Atributo	**Tipo**
idn	int

idc	int

Tabla 4 Clase "TRel_FUC"

Nombre: TRel_FUC	
Descripción: Clase encargada de almacenar los identificadores de los requisitos de tipos **características** y **casos de uso** que poseen relaciones de trazabilidad.	
Tipo de clase: Entidad	
Atributo	**Tipo**
idf	int
idc	int

Tabla 5 Clase "TRel_NFUS"

Nombre: TRel_NFUS	
Descripción: Clase encargada de almacenar los identificadores de los requisitos de tipos **no funcionales** e **historia de usuarios** que poseen relaciones de trazabilidad.	
Tipo de clase: Entidad	
Atributo	**Tipo**
idnf	int
idus	int

Tabla 6 Clase "TRel_UNUS"

Nombre: TRel_UNUS

Descripción: Clase encargada de almacenar los identificadores de los requisitos de tipos **necesidades de usuarios** e **historia de usuarios** que poseen relaciones de trazabilidad.	
Tipo de clase: Entidad	
Atributo	**Tipo**
idn	int
idus	int

Tabla 7 Clase "TRel_FNF"

Nombre: TRel_FNF	
Descripción: Clase encargada de almacenar los identificadores de los requisitos de tipos **características** y **no funcionales** que poseen relaciones de trazabilidad.	
Tipo de clase: Entidad	
Atributo	**Tipo**
idn	int
idus	int

Tabla 8 Clase "TParent"

Nombre: TParent	
Descripción: Clase encargada de almacenar los datos de los requisitos que jerárquicamente son padres de otros.	
Tipo de clase: Entidad	
Atributo	**Tipo**
idp	int
namep	text

Tabla 9 Clase "TChildren"

Nombre: TChildren	
Descripción: Clase encargada de almacenar los datos de los requisitos que jerárquicamente son hijos de otros.	
Tipo de clase: Entidad	
Atributo	**Tipo**
idh	int
idp	int
nameh	text

Tabla 10 Clase "Data_Access"

Nombre: Data_Access	
Descripción: Clase encargada de crear las tablas y realizar las distintas operaciones en la BD	
Tipo de Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	def __init__(self): void
Descripción:	*Constructor de la clase.*
Nombre:	def CreateTables(self): void
Descripción:	*Crea las diferentes tablas en la BD para el procesamiento de los datos.*
Nombre:	def TableReq(self, i, n, t, p, d, di, e, lt, es, o): void
Descripción:	*Inserta en la tabla TRequirements los datos pasados por parámetros de los requisitos.*

Nombre:	def TableRelUNUC(self, idn, idc): void
Descripción:	*Inserta en la tabla TRel_UNUC los identificadores de los requisitos de tipos necesidades de usuarios y casos de uso que se relacionan.*
Nombre:	def TableRelFUC(self, idf, idc): void
Descripción:	*Inserta en la tabla TRel_FUC los identificadores de los requisitos de tipos características y casos de uso que se relacionan.*
Nombre:	def TableRelNFUS(self, idnf, idus): void
Descripción:	*Inserta en la tabla TRel_NFUS los identificadores de los requisitos de tipos no funcionales e historia de usuarios que se relacionan.*
Nombre:	def TableRelUNUS(self, idn, idus): void
Descripción:	*Inserta en la tabla TRel_UNUS los identificadores de los requisitos de tipos necesidades de usuarios e historia de usuarios que se relacionan.*
Nombre:	def TableRelFNF(self, idf, idnf): void
Descripción:	*Inserta en la tabla TRel_FNF los identificadores de los requisitos de tipos características y no funcionales que se relacionan.*
Nombre:	def TableParent(self, idp, namep): void
Descripción:	*Inserta en la tabla TParent el identificador y el nombre del requisito que es padre.*
Nombre:	def TableChildren(self, idh, idp, nameh): void
Descripción:	*Inserta en la tabla TChildren el identificador y el nombre del requisito además del identificador de su padre.*
Nombre:	def Length(self, sql=''): int
Descripción:	*Retorna la longitud más uno de la lista que devuelve la consulta pasada por parámetros.*
Nombre:	def Consult(self, sql=''): void
Descripción:	*Ejecuta la consulta pasada por parámetros.*
Nombre:	def SQLQuery(self, sql=''): tuple
Descripción:	*Retorna la tupla que devuelve la consulta pasada por parámetros.*

Nombre:	def ListID(self, sql=''): list
Descripción:	*Retorna la lista de la consulta pasada por parámetros.*
Nombre:	def NotChildren(self): list
Descripcion:	*Devuelve una lista con los requisitos que no son hijos para la posterior asignación de padres.*

Tabla 11 Clase "Navigation"

Nombre: Navigation	
Descripción: Clase padre que posee todos los métodos que serán redefinidos por las clases hijas.	
Tipo de Clase:	
Atributos	**Tipo**
data	dict
dataaccess	object
Para cada responsabilidad:	
Nombre:	def __init__(self): void
Descripción:	*Constructor de la clase.*
Nombre:	def match_request(self, req): string
Descripción:	*Captura y devuelve la direccción de las peticiones realizadas por el usuario .*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retornar el resultado del proceso a la misma página para su actualización.*
Nombre:	def get_htdocs_dirs(self): string

Descripción:	*Devuelve el directorio donde están las imágenes y los estilos.*
Nombre:	def get_templates_dirs(self): string
Descripción:	*Devuelve el directorio donde están las páginas o templates.*
Nombre:	def styles(self, req): void
Descripción:	*Adiciona los script y estilos que serán utilizados.*

Tabla 12 Clase "Main_Page"

Nombre: Main_Page	
Descripción: Clase encargada de activar el vínculo del plugin en el trac y establecer la navegación con la página principal.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def get_active_navigation_item(self, req): string
Descripción:	*Activa la página principal del plugin.*
Nombre:	def get_navigation_items(self, req): string
Descripción:	*Activa el vínculo del plugin en la página principal del Trac y establece la comunicación entre vínculo del plugin con la página principal del mismo.*
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página home_page.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict

Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*

Tabla 13 Clase "Add_Req"

Nombre: Add_Req	
Descripción: Clase encargada de adicionar los datos de los requisitos.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página add_page.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def InsertShow(self, req): void
Descripción:	*Inserta en los campos de la tabla TRequirements de la BD los valores pasados por formulario html.*
Nombre:	def ParentChildren(self, req): void
Descripción:	*Inserta en la tabla TChildren los requisitos que son hijos y sus padres y en la tabla TParent los requisitos que son padres según sean asignados en la adición de un requisito.*

Tabla 14 Clase "Modify_Req"

Nombre: Modify_Req	
Descripción: Clase encargada de la entrada de los datos a modificar en los requisitos.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página mod_page.html, capturas la petición y eventos de la misma y retornas mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*

Tabla 15 Clase "Select_Mod"

Nombre: Select_Mod	
Descripción: Clase encargada de modificar los datos de los requisitos según desee el usuario y mostrar los cambios.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página req_mod_page.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*

Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def ModifyReq(self, n, t, p, d, di, e, it, es, o, padre, hijos, id, req): void
Descripción:	*Modifica los datos de la tabla TRequirements según los parámetros pasados, actualiza las tablas Trel_UNUC, Trel_FUC, Trel_NFUS, Trel_UNUS y TRel_FNF si es necesario modificar el campo tipo de requisito y actualiza las tablas TParent y TChildren en caso de modificar la jerarquía del requisito.*
Nombre:	def ChildrenChanges(*self*, padre, hijos, id, req): void
Descripción:	*Efectúa los cambios en la jerarquía de hijos del requisito especificado.*

Tabla 16 Clase "Delete_Req"

Nombre: Delete_Req	
Descripción: Clase encargada de eliminar los datos de los requisitos deseados por el usuario y mostrar la actualización de los cambios.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página del_page.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*

Tabla 17 Clase "Report_Req"

Nombre: Report_Req	
Descripción: Clase encargada de obtener reportes según los atributos seleccionados por el usuario y mostrar el resultado.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página a_report_page.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def Reports(self, req): tuple
Descripción:	*Realiza la búsqueda del reporte solicitado.*

Tabla 18 Clase "Matrix_UNUC"

Nombre: Matrix_UNUC	
Descripción: Clase encargada de mostrar la matriz de trazabilidad entre los requisitos **casos de uso** y **necesidades de usuario** y establecer las relaciones de trazabilidad.	
Tipo de Clase: Controladora	
Atributos	**Tipo**

Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página matrix_page_ncu.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def TRelUNUC(self): void
Descripción:	*Define las relaciones de trazabilidad entre requisitos de tipos necesidades de usuarios y casos de uso seleccionados por el usuario.*

Tabla 19 Clase "Matrix_FUC"

Nombre: Matrix_FUC	
Descripción: Clase encargada mostrar la matriz de trazabilidad entre los requisitos **características** y **casos de uso** y establecer las relaciones de trazabilidad.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página matrix_page_fcu.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def TRelFUC(self): void
Descripción:	*Define las relaciones de trazabilidad entre requisitos de tipos características y*

	casos de uso seleccionados por el usuario.

Tabla 20 Clase "Matrix_NFUS"

Nombre: Matrix_NFUS	
Descripción: Clase encargada de mostrar la matriz de trazabilidad entre los requisitos **no funcionales** e **historia de usuarios** y establecer las relaciones de trazabilidad.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página matrix_page_nfus.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def TRelNFUS(self): void
Descripción:	*Define las relaciones de trazabilidad entre requisitos de tipos no funcionales e historia de usuarios seleccionados por el usuario.*

Tabla 21 Clase "Matrix_UNUS"

Nombre: Matrix_UNUS

Descripción: Clase encargada de mostrar la matriz de trazabilidad entre los requisitos **necesidades de usuario** e **historia de usuarios** y establecer las relaciones de trazabilidad.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página matrix_page_unus.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def TRelUNUS(self): void
Descripción:	*Define las relaciones de trazabilidad entre requisitos de tipos necesidades de usuarios e historia de usuarios seleccionados por el usuario.*

Tabla 22 Clase "Matrix_FNF"

Nombre: Matrix_FNF	
Descripción: Clase encargada de mostrar la matriz de trazabilidad entre los requisitos **características** y **no funcionales** y establecer las relaciones de trazabilidad.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página matrix_page_fnf.html, captura la petición y*

	eventos de la misma y retorna mediante el request la dirección de la petición.
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def TRelFNF(self): void
Descripción:	*Define las relaciones de trazabilidad entre requisitos de tipos características y no funcionales seleccionados por el usuario.*

Tabla 23 Clase "Hierarchical_View"

Nombre: Hierarchical_View	
Descripción: Clase encargada de mostrar la vista jerárquica de los requisitos.	
Tipo de Clase: Controladora	
Atributos	**Tipo**
Para cada responsabilidad:	
Nombre:	def match_request(self, req): string
Descripción:	*Establece conexión con la página hrv_page.html, captura la petición y eventos de la misma y retorna mediante el request la dirección de la petición.*
Nombre:	def process_request(self, req): dict
Descripción:	*Procesa las peticiones y eventos de la página enviada por request y retorna el resultado del proceso a la misma página para su actualización.*
Nombre:	def HierarchicalTable(self): void
Descripción:	*Realiza una búsqueda de los hijos del padre seleccionado.*

3.8 Diagrama de Componentes

Un diagrama de componentes representa las organizaciones y dependencias lógicas entre componentes de software, estos pueden ser componentes de código fuente, binarios o ejecutables. Además tiene en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y restricciones impuestas por los lenguajes de programación y herramientas utilizadas en el desarrollo. Los elementos dentro de este diagrama serán componentes y paquetes.

La estructura del plugin de Gestión de Requisitos *(Plugin GR)* está compuesta por varios paquetes y componentes, los cuales se relacionan entre si según el diseño trazado para la implementación del mismo.

El diagrama de componentes de la aplicación **(Ver figura 9)** cuenta con tres paquetes fundamentales: *presentation*, *business_logic* y *data_access*. Este se relaciona mediante el componente *dataaccess.py* con el componente *trac.db* (Base de Datos del Trac) ubicado en el paquete *db* de la herramienta Trac **(Ver figura 12)**.

A continuación se describen los paquetes que conforman el diagrama de componentes del Plugin GR.

- El paquete *presentation* está compuesto por dos paquetes: *templates* y *htdocs***.** El paquete *templates* **(Ver figura 10)** está formado por varios componentes y el paquete *htdocs* **(Ver figura 11)** está compuesto por otro paquete *css* contenedor de varios componentes.
- El paquete *business_logic* está compuesto por dos componentes: *req_manag.py* e *__init__.py*.
- El paquete *data_access* está compuesto por un componente: *dataaccess.py*.

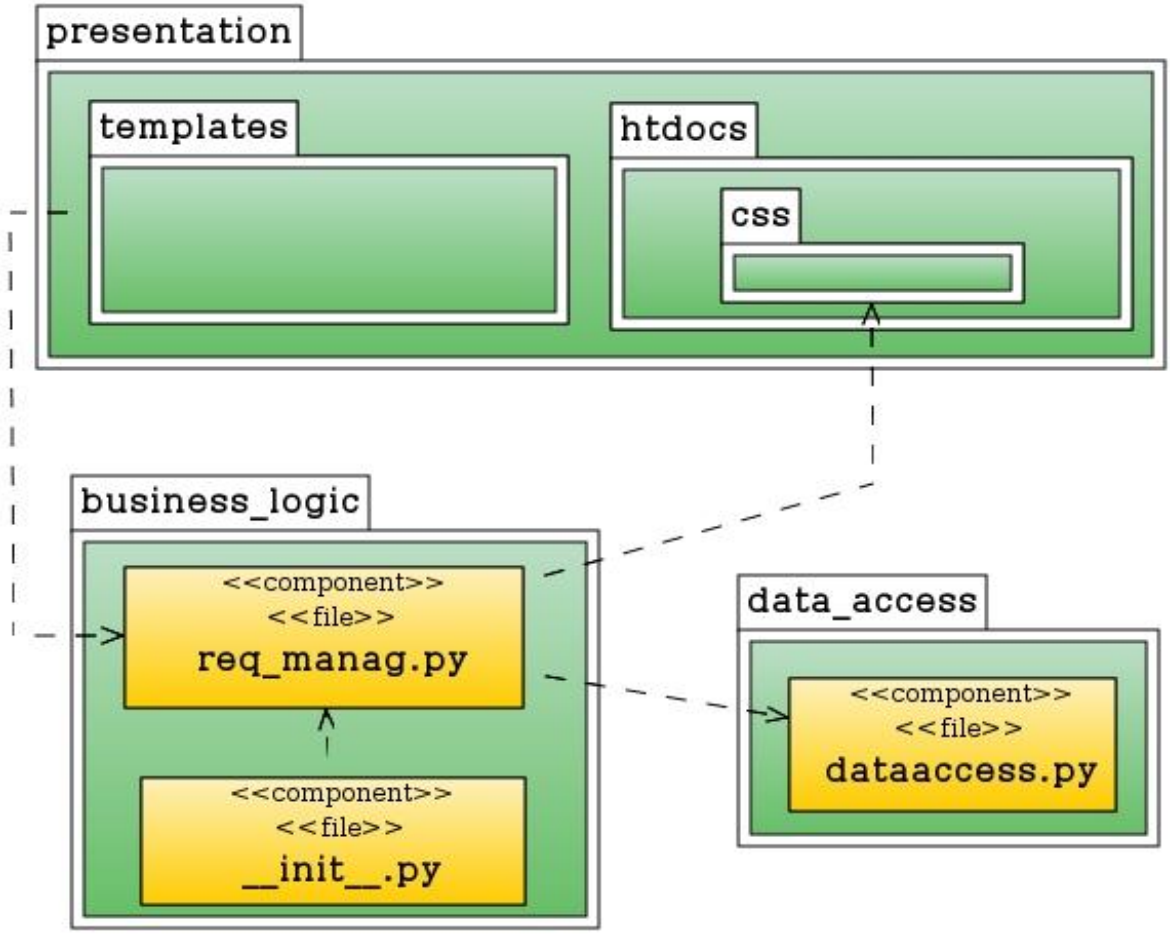


Figura 9 Diagrama de Componentes del *Plugin GR*

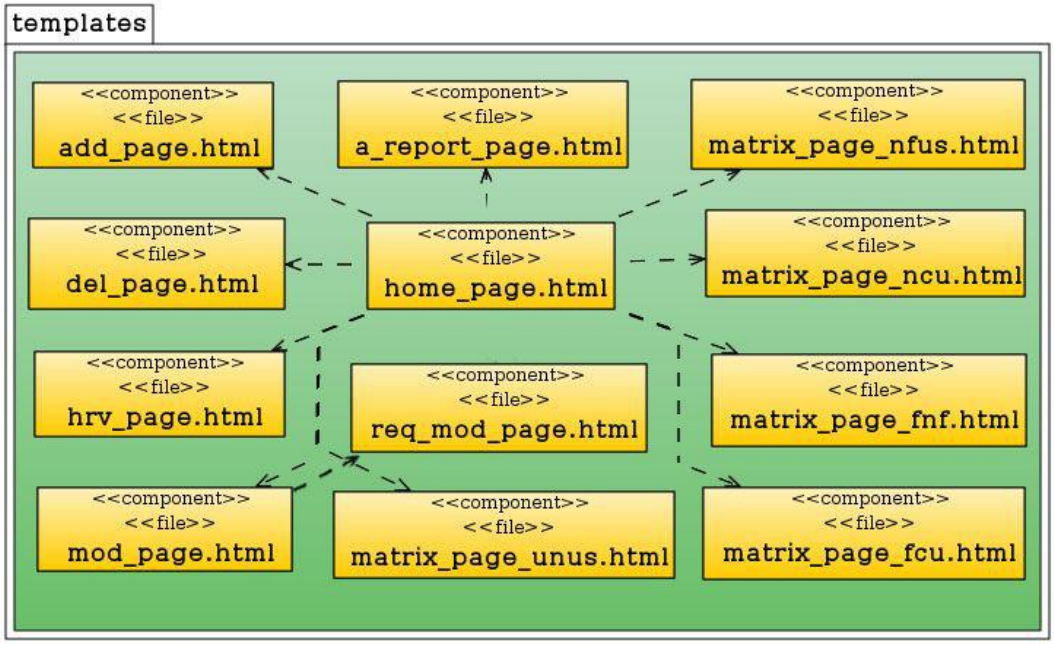


Figura 10 Paquete *templates*

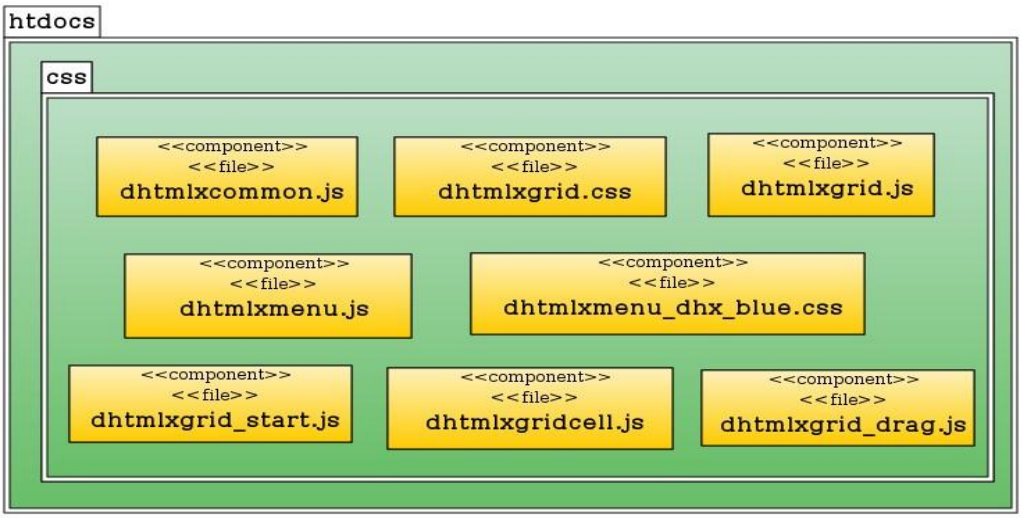


Figura 11 Paquete *htdocs*

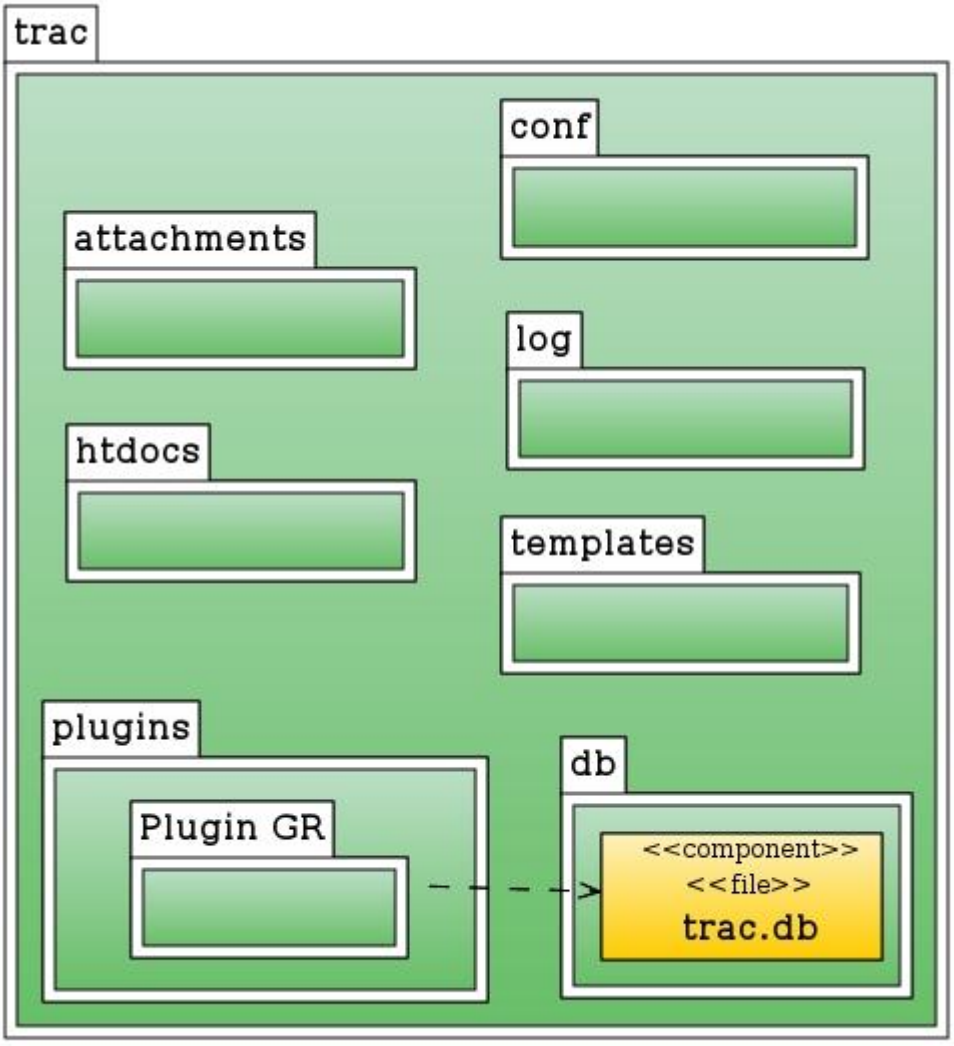


Figura 12 Diagrama de paquetes de la herramienta Trac

3.9 Diagrama de Despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que entran en la composición de un sistema y el reparto de los programas ejecutables sobre estos nodos. (17) Además muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

La herramienta de Gestión de Proyectos Trac estará instalada en el Servidor Apache utilizando como sistema Gestor de Base de Datos SQlite. Cada PC cliente podrán acceder a dicho servidor mediante el protocolo de comunicación <HTTP>, permitiendo la plena comunicación con la herramienta.

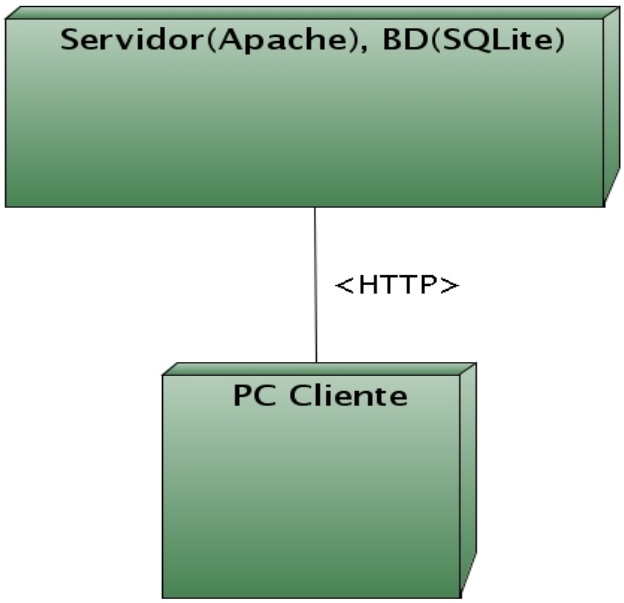


Figura 13 Diagrama de Despliegue

Conclusiones

En el presente capítulo, a partir del estudio realizado se concluye que el patrón de arquitectura en 3 capas y el patrón de diseño Fachada ofrecen ventajas para la implementación de la solución, por lo que fueron empleados durante el desarrollo de la misma. Para una buena comprensión del trabajo se realizaron las descripciones de las clases involucradas en la implementación de la aplicación, el diagrama de clases del diseño, los diagramas de componentes, despliegue y de secuencia por cada caso de uso del sistema.

Capítulo 4: Pruebas del sistema

Introducción

En este capítulo se lleva a cabo el proceso de pruebas del software para la detección y corrección de los errores que pueda presentar la aplicación con el objetivo de darles solución y respuesta a los mismos antes de dar por terminada la aplicación, la pruebas no excluyen la aparición de defectos en el software pero permiten desarrollar una serie de técnicas y métodos para la solución de los posibles problemas que puedan surgir.

4.1 Plan de Pruebas

Para garantizar que una aplicación cuenta con la calidad requerida es necesario llevar a cabo un buen Plan de Pruebas, este es el factor fundamental para lograr el éxito en la puesta en práctica de un proceso de pruebas que permita entregar un software que cumpla o supere las expectativas del cliente. El Plan de Pruebas, para ser ejecutado correctamente debe contener casos de prueba que cumplan con las siguientes características:

- **Precisos**: Las pruebas deben describir correctamente lo que verificarán.
- **Económicos**: Sólo ha de contener los pasos necesarios para su propósito.
- **Repetible**: Las pruebas deben ser consistentes y recoger todo lo necesario (entorno, estado inicial) para que los resultados de cualquier ejecución sean los mismos.
- **Adecuado**: La prueba debe describir la situación en la que es aplicable (validación, regresión).
- **Trazable**: La prueba debe estar relacionada con el requisito funcional que cubre con objeto de facilitar la identificación del fallo y permitir hacer un seguimiento de las correcciones más fácilmente.

Para la realización del proceso de pruebas en la solución obtenida se pone en práctica la estrategia de pruebas definida por el nivel, tipo y método de prueba representados en los siguientes subepígrafes.

4.1.1 Nivel de Prueba: Prueba de Sistema

Las pruebas de sistema verifican el correcto funcionamiento del sistema completo incluyendo casos de prueba que busquen los fallos del sistema. Son pruebas destructivas y persiguen demostrar la robustez del sistema aun en condiciones adversas. Verifican requisitos funcionales y no funcionales. (18)

4.1.2 Tipo de Prueba: Funcionalidad

Una vez que se ha desarrollado una aplicación, es necesario verificar que su funcionamiento es correcto y que cumple los objetivos de su diseño. Las pruebas de funcionalidad determinan la medida en la que la aplicación satisface los requisitos funcionales esperados. Durante el proceso de pruebas se simulan varios escenarios para confirmar que todos los resultados satisfacen las expectativas.

4.1.3 Método de Prueba: Caja Negra

Las pruebas de caja negra son pruebas funcionales que se aplican al sistema empleando un conjunto de datos de entrada y observando las salidas que se obtienen para determinar si la función se está desarrollando correctamente por el sistema bajo prueba. Su realización parte de los requisitos funcionales.

Algunos ejemplos básicos de pruebas de caja negra son la comprobación de valores límite, pruebas de integridad de la base de datos o pruebas de excepciones.

Para las pruebas a la aplicación se introdujeron diferentes valores con el objetivo de verificar el correcto funcionamiento de la misma en diferentes escenarios, al introducir datos válidos e inválidos a partir de los requerimientos funcionales.

4.2 Diseño de Casos de Prueba

El diseño de casos de prueba tiene como objetivo verificar que las funcionalidades de la aplicación cumplen su propósito con los resultados esperados.

4.2.1 Caso de Uso Gestionar requisitos

Descripción General

Mediante este caso de uso el usuario podrá realizar diversas operaciones como adicionar, modificar y eliminar requisitos.

Casos de Pruebas a realizar:

- Adicionar requisito.
- Eliminar requisitos.
- Modificar requisito.

CP 1 Adicionar requisito

Breve Descripción

Este caso de prueba le permite al usuario adicionar requisitos al sistema.

Flujo Central

1. El usuario selecciona la opción "Managing Requirements".
2. El sistema muestra tres opciones: "Add Requirements", "Modify Requirements" y "Remove Requirements".
3. El usuario selecciona la opción "Add Requirements".

4. El sistema muestra un formulario con los campos a llenar del requisito: Name, Iteration, Status, Origin, Type, Priority, Difficulty, Stability, Description, Parent, Children.
5. El usuario entra los datos correspondientes a cada unos de los campos del requisito y presiona el botón "Add".
6. El sistema adiciona el requisito satisfactoriamente en la BD.

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Seleccionar la opción "Add Requirements"		El sistema debe mostrar un formulario con los campos de los atributos del requisito.	El sistema muestra un formulario con cada los campos de los atributos del requisito.	

Presionar el botón "Add". Adicionar un nuevo requisito (campo Name no puede estar en blanco) Ejemplo: Name: Gestionar requisitos Status: Proposed Origin: End Users Type: Use Case Priority: High		El sistema debe adicionar el nuevo requisito en la BD.	El sistema adiciona el requisito en la BD.	
	Adicionar un requisito dejando el campo Name en blanco. Ejemplo: ***Name:*** Status: Proposed Origin: End Users Type: Use Case Priority: High	El sistema debe mostrar un mensaje indicando que se le debe poner un nombre al requisito.	El sistema muestra un mensaje indicando que se le debe poner un nombre al requisito.	

	Adicionar un requisito con el nombre de uno ya existente. Ejemplo: ***Name:*** Gestionar requisitos Status: Proposed Origin: End Users Type: Use Case Priority: High	El sistema debe mostrar un mensaje alertando que ya existe un requisito con ese nombre.	El sistema muestra un mensaje notificando que existe un requisito con ese nombre.	
	Adicionar un requisito asignándole como padre e hijo un mismo requisito. Ejemplo: Name: Adicionar requisito Status: Proposed Origin: End Users Type: Use Case Priority: High ***Parent:*** 1 ***Children:*** 1	El sistema debe mostrar un mensaje que indique que se escogió como padre e hijo un mismo requisito y el resultado de la operación.	El sistema muestra un mensaje notificando que se escogió como padre e hijo el mismo requisito y que este fue asignado como padre.	

CP 2 Eliminar requisitos

Breve Descripción

Este caso de prueba le permite al usuario eliminar requisitos del sistema.

Flujo Central

1. El usuario selecciona la opción "Managing Requirements".
2. El sistema muestra tres opciones: "Add Requirements", "Modify Requirements" y "Remove Requirements".
3. El usuario selecciona la opción "Remove Requirements".
4. El sistema muestra una tabla con todos los requisitos existentes en la BD.
5. El usuario escoge los requisitos que desea eliminar y presiona el botón "Remove".
6. El sistema elimina los requisitos correctamente y actualiza la BD.
7. El sistema muestra la tabla actualizada.

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Seleccionar la opción "Remove Requirements".		El sistema debe mostrar una tabla con todos los requisitos existentes en la BD.	El sistema muestra una tabla con todos los requisitos existentes en la BD.	
Seleccionar los requisitos a eliminar.		El sistema debe permitir seleccionar los requisitos que se desean eliminar.	El sistema permite seleccionar los requisitos que se desean eliminar.	
Presionar el botón "Remove". Eliminar los requisitos seleccionados.		El sistema debe eliminar los requisitos seleccionados y mostrar la tabla actualizada.	El sistema elimina los requisitos seleccionados y muestra la tabla actualizada.	
	El usuario presiona el botón "Remove" sin haber seleccionado ningún requisito.	El sistema debe mostrar un mensaje indicando que se debe seleccionar al menos un requisito.	El sistema muestra un mensaje notificando que debe seleccionar al menos un requisito.	

CP 3 Modificar requisito

Breve Descripción

Este caso de prueba le permite al usuario modificar un requisito existente.

Flujo Central

1. El usuario selecciona la opción "Managing Requirements".
2. El sistema muestra tres opciones: "Add Requirements", "Modify Requirements" y "Remove Requirements".
3. El usuario selecciona la opción "Modify Requirements".
4. El sistema muestra una tabla con todos los requisitos existentes en la BD.
5. El usuario escoge el requisito que desea modificar y presiona el botón "Modify".
6. El sistema muestra un formulario con todos los campos y valores del requisito escogido.
7. El usuario entra el nuevo valor de los campos deseados y presiona el botón "Modify".

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Seleccionar la opción "Modify Requirements".		El sistema debe mostrar una tabla con todos los requisitos existentes en la BD.	El sistema muestra una tabla con todos los requisitos existentes en la BD.	

Seleccionar el requisito que desea modificar.		El sistema debe permitir seleccionar el requisito que se desea modificar.	El sistema permite seleccionar el requisito que se desea modificar.	
Presionar el botón "Modify".		El sistema debe mostrar un formulario con los campos y valores del requisito seleccionado.	El sistema muestra un formulario con los campos y valores del requisito seleccionado.	
Presionar el botón "Modify". Modificar el requisito seleccionado. Ejemplo: ID: 1 Name: Gestionar requisitos Status: Aproved		El sistema debe modificar los valores de los atributos del requisito y mostrar la tabla de requisitos actualizada.	El sistema modifica los valores de los atributos del requisito y muestra la tabla actualizada.	
	Se trata de asignar como padre el requisito que se está modificando.	El sistema debe mostrar un mensaje indicando que no se puede asignar a un requisito como su	El sistema muestra un mensaje notificando que no se puede asignar a un requisito a sí	

	Ejemplo: ***ID:*** 1 ***Parent:*** 1	padre.	mismo como su padre.	
	Se trata de asignar como hijo el requisito que se está modificando. Ejemplo: ***ID:*** 1 ***Children:*** 1	El sistema debe mostrar un mensaje indicando que no se puede asignar a un requisito como su hijo.	El sistema muestra un mensaje notificando que no se puede asignar a un requisito a sí mismo como su hijo.	
	Se trata de asignar como padre e hijo un mismo requisito. Ejemplo: ID: 1 ***Parent:*** 2 ***Children:*** 2	El sistema debe mostrar un mensaje notificando que no se puede asignar a un requisito como padre e hijo e indicar la operación realizada.	El sistema muestra un mensaje notificando que no se puede asignar a un requisito como padre e hijo y que este fue asignado como padre.	

4.2.2 Caso de Uso Obtener reporte

Descripción general

Mediante este caso de uso el usuario podrá obtener reportes de requisitos.

Casos de Prueba a realizar:

- Obtener reporte de requisitos por atributos.

CP 4 Obtener reporte

Breve Descripción

Este caso de prueba le permite al usuario obtener reporte de requisitos según los atributos seleccionados.

Flujo Central

1. El usuario selecciona la opción "Get Report by Attributes".
2. El sistema muestra un formulario con los atributos del requisito y una tabla con los requisitos existentes en la Base de Datos.
3. El usuario selecciona los atributos por los cuales desea obtener el reporte y presiona el botón "View Report".
4. El sistema muestra el reporte.

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Seleccionar la opción "Get Report by Attributes".		El sistema debe mostrar un formulario con los atributos del requisito y una tabla con los requisitos existentes en la BD.	El sistema muestra un formulario con los atributos del requisito y una tabla con los requisitos existentes en la BD.	
Seleccionar los valores de los atributos para obtener el reporte.		El sistema debe permitir seleccionar los valores de los atributos para obtener el reporte.	El sistema permite seleccionar los valores de los atributos para obtener el reporte.	
Presiona el botón "View Report". Obtener reporte según los valores de los atributos seleccionados. Ejemplo: Difficulty: High Iteration: 2		El sistema debe mostrar los requisitos que cumplen con el reporte solicitado.	El sistema muestra los requisitos que cumplen con el reporte solicitado.	

	El usuario presiona el botón "View Report" sin haber seleccionado ningún atributo.	El sistema debe mostrar un mensaje notificando que debe seleccionar algún atributo para obtener el reporte.	El sistema muestra un mensaje notificando que debe seleccionar algún atributo para obtener el reporte.	

4.2.3 Caso de Uso Obtener vistas de matriz de trazabilidad

Descripción General

Mediante este caso de uso el usuario accederá a diferentes vistas de la matriz de trazabilidad entre requisitos.

Casos de Prueba a realizar:

- Obtener vista de matriz de trazabilidad entre requisitos Features y Use Cases.

CP 5 Vista de matriz Features - Use Cases

Breve Descripción

Este caso de prueba le permite al usuario ver la matriz de trazabilidad entre requisitos Features y Use Cases.

Flujo Central

1. El usuario selecciona la opción "Traceability Matrix Views".

2. El sistema muestra cinco opciones: "Features – Use Cases", "User Needs – Use Cases", "Not Functional – User Storyboard", "User Needs – User Storyboard" y "Features – Not Functional".
3. El usuario escoge la opción "Features – Use Cases".
4. El sistema muestra una tabla con los requisitos Features y Use Cases y la matriz de trazabilidad con las relaciones existentes entre ellos.

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Seleccionar opción "Features – Use Cases".		El sistema debe mostrar una tabla con los requisitos Features y Use Cases, y la matriz de trazabilidad con las relaciones existentes entre ellos.	El sistema muestra una tabla con los requisitos Features y Use Cases, y la matriz de trazabilidad con las relaciones existentes entre ellos.	

	Si no existen requisitos Features y Use Cases.	El sistema debe mostrar un mensaje notificando que no existen requisitos Features y Use Cases.	El sistema muestra un mensaje notificando que no existen requisitos Features y Use Cases.	
	Si no existen requisitos Features.	El sistema debe mostrar un mensaje notificando que no existen requisitos Features.	El sistema muestra un mensaje notificando que no existen requisitos Features.	
	Si no existen requisitos Use Cases.	El sistema debe mostrar un mensaje notificando que no existen requisitos Use Cases.	El sistema muestra un mensaje notificando que no existen requisitos Use Cases	

Observación: *Este caso de uso cuenta además con los Casos de Prueba clave User Needs – Use Cases, Not Functional – User Storyboard, User Needs – User Storyboard y Features – Not Functional, que no se reflejan en los casos de prueba por la similitud de pasos con el Caso especificado.*

4.2.4 Caso de Uso Definir relaciones de trazabilidad

Descripción General

Mediante este caso de uso el usuario tendrá la posibilidad de agregar y eliminar relaciones de trazabilidad entre diferentes tipos de requisitos.

Casos de Prueba a realizar:

- Agregar relación de trazabilidad a los requisitos Features y Use Cases.
- Eliminar relación de trazabilidad a los requisitos Features y Use Cases.

CP 6 Agregar relación Features – Use Cases

Breve descripción:

Este caso de prueba le permite al usuario agregar relación de trazabilidad entre Features y Use Cases.

Flujo Central:

1. El flujo central transcurre de forma similar al flujo central del caso de uso Obtener Vistas de Matriz de Trazabilidad, a continuación el usuario podrá agregar nuevas relaciones de trazabilidad o eliminar algunas existentes.

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la	Observac

			Prueba	**iones**
Escoger los identificadores de los requisitos Features y Use Cases.		El sistema debe permitir escoger los identificadores de los requisitos Features y Use Cases que se desean relacionar.	El sistema permite escoger los identificadores de los requisitos Features y Use Cases que se desean relacionar.	
Presiona el botón "Add". Agregar relación de trazabilidad entre requisitos de tipo Features y Use Cases. Ejemplo: F: 2 UC: 1		El sistema debe mostrar la matriz de trazabilidad actualizada.	El sistema muestra la matriz de trazabilidad actualizada.	
	El usuario presiona el botón "Add" si haber seleccionado ningún requisito.	El sistema debe mostrar un mensaje advirtiendo que se no seleccionó ningún requisito.	El sistema muestra un mensaje advirtiendo que se no seleccionó ningún requisito.	
	El usuario presiona el botón "Add" si haber seleccionado el requisito de tipo	El sistema debe mostrar un mensaje notificando que no se seleccionó el requisito	El sistema muestra un mensaje notificando que no se seleccionó el	

	Feature.	de tipo Feature.	requisito de tipo Feature.	
	El usuario presiona el botón "Add" si haber seleccionado el requisito de tipo Use Case.	El sistema debe mostrar un mensaje notificando que no se seleccionó el requisito de tipo Use Case.	El sistema muestra un mensaje notificando que no se seleccionó el requisito de tipo Use Case.	

CP 7 Eliminar relación Features – Use Cases

Breve descripción:

Este caso de prueba le permite al usuario eliminar relación de trazabilidad entre Features y Use Cases.

Flujo Central:

2. El flujo central transcurre de forma similar al flujo central del caso de uso Obtener Vistas de Matriz de Trazabilidad, a continuación el usuario podrá agregar nuevas relaciones de trazabilidad o eliminar algunas existentes.

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Escoger los identificadores de los requisitos Features y Use Cases.		El sistema debe permitir escoger los identificadores de los requisitos Features y Use Cases que se les desea eliminar la relación de trazabilidad.	El sistema permite escoger los identificadores de los requisitos Features y Use Cases que se les desea eliminar la relación de trazabilidad..	
Presiona el botón "Remove". Eliminar relación de trazabilidad entre requisitos de tipo Features y Use Cases. Ejemplo: F: 2 UC: 1		El sistema debe mostrar la matriz de trazabilidad actualizada.	El sistema muestra la matriz de trazabilidad actualizada.	
	El usuario presiona el botón "Remove" si haber seleccionado ningún requisito.	El sistema debe mostrar un mensaje advirtiendo que se no seleccionó ningún	El sistema muestra un mensaje advirtiendo que se no seleccionó	

		requisito.	ningún requisito.	
	El usuario presiona el botón "Remove" si haber seleccionado el requisito de tipo Feature.	El sistema debe mostrar un mensaje notificando que no se seleccionó el requisito de tipo Feature.	El sistema muestra un mensaje notificando que no se seleccionó el requisito de tipo Feature.	
	El usuario presiona el botón "Remove" si haber seleccionado el requisito de tipo Use Case.	El sistema debe mostrar un mensaje notificando que no se seleccionó el requisito de tipo Use Case.	El sistema muestra un mensaje notificando que no se seleccionó el requisito de tipo Use Case.	

Observación: *También se puede definir relaciones de trazabilidad para las vistas entre User Needs – Use Cases, Not Functional – User Storyboard, User Needs – User Storyboard y Features – Not Functional, que no se reflejan en los casos de prueba por la similitud de pasos con el Caso especificado.*

4.2.5 Caso de Uso Obtener vista jerárquica de requisitos

Descripción general

Mediante este caso de uso el usuario podrá tener una vista de las relaciones jerárquicas existentes en los requisitos.

Casos de Prueba a realizar:

- Obtener vistas jerárquicas entre requisitos.

CP 11 Obtener vista jerárquica de requisitos

Breve Descripción

Este caso de prueba le permite al usuario ver las relaciones jerárquicas entre requisitos.

Flujo Central

1. El usuario selecciona la opción "Hierarchical Requirements View".
2. El sistema muestra una tabla con el identificador y nombre de los requisitos padres existentes.
3. El usuario selecciona el requisito padre al cual le desea ver los requisitos hijos y se presiona el botón "View Children".
4. El sistema muestra una tabla jerárquica con el identificador del requisito padre seleccionado y los requisitos hijos.

Condiciones de ejecución.

- El usuario debe estar autenticado.

Iteraciones

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Seleccionar la opción "Hierarchical Requirements View".		El sistema debe mostrar una tabla con el identificador y nombre de los requisitos padres existentes	El sistema muestra una tabla con el identificador y nombre de los requisitos padres existentes	
Seleccionar el requisito padre al cual se le desean ver sus hijos.		El sistema debe permitir seleccionar el requisito padre al cual se le desean ver sus hijos.	El sistema permite seleccionar el requisito padre al cual se le desean ver sus hijos.	
Presionar el botón "View Children" Ver los hijos del requisito padre seleccionado. Ejemplo: ID Parent: 2 ID Children: 5 Name Children: adicionar requisito.		El sistema debe mostrar una tabla con el requisito padre seleccionado y sus hijos.	El sistema muestra una tabla con el requisito padre seleccionado y sus hijos.	

	Si se presiona el botón "View Children" si haber seleccionado ningún requisito padre.	El sistema debe mostrar un mensaje notificando que no se seleccionó ningún requisito padre.	El sistema muestra un mensaje notificando que no se seleccionó ningún requisito padre.	

4.3 Resumen de Pruebas

En el proceso de pruebas efectuado se diseñaron 11 Casos de Prueba con el objetivo de verificar el correcto funcionamiento de la aplicación y su cumplimiento con las principales funcionalidades. Se obtuvo un total de 5 no conformidades, las cuales fueron solucionadas. Como resultado final se logró una aplicación que cumple con los requisitos funcionales. No se asegura el cumplimiento de los requisitos no funcionales dado que el producto no fue sometido a pruebas para la verificación de los mismos.

4.4 Resultados presentes y visión futura

Con el desarrollo de la solución incorporada a la herramienta de Gestión de Proyectos Trac, se han obtenido resultados significativos para la mejora de tareas pertinentes a la Gestión de Requisitos que es necesario realizar como parte de la producción en los proyectos del polo productivo HA.

La solución expuesta permitirá acometer diversas tareas propias de la Gestión de Requisitos de forma automatizada y en un entrono colaborativo, donde las personas implicadas tendrán acceso a la información necesaria ganando en el tiempo dedicado al ciclo de desarrollo de los productos.

Conclusiones

En este capítulo se realizaron pruebas al sistema con el objetivo de verificar el correcto funcionamiento de la aplicación en cada uno de los escenarios. Al término del mismo se puede concluir que se cumplió con los objetivos propuestos: detectar no conformidades en el sistema tras la puesta en práctica de los casos de prueba diseñados y dar solución a las no conformidades encontradas. Es necesario destacar que las pruebas realizadas validan el correcto funcionamiento de la solución atendiendo a los requisitos funcionales de la misma.

Conclusiones

Luego de la investigación realizada y puesta en práctica de los conocimientos adquiridos para el desarrollo del presente trabajo y tras la ejecución de las tareas planteadas para el cumplimiento del proyecto se concluye que:

- Se realizó el estudio de información existente relacionada con la Gestión de Requisitos obteniéndose conocimientos necesarios para el desarrollo de la solución.
- Se realizó el diseño de una extensión que permite llevar a cabo tareas propias de la Gestión de Requisitos, respetando la arquitectura de la herramienta Trac.
- Se implementó una herramienta que cumple con los objetivos propuestos en el alcance definido para la misma, permitiendo la realización de diferentes tareas importantes en el trabajo dentro de los proyectos del polo.
- La puesta en práctica del Plan de Pruebas diseñado permitió la detección de no conformidades, cuya solución ayudó a mejorar la calidad de la aplicación.

Recomendaciones

Se recomienda al siguiente trabajo:

- Definir nuevas vistas de matrices de trazabilidad.
- Integrar la solución con el resto de las extensiones desarrolladas para la Gestión de Proyectos.
- Aumentar el número de funcionalidades al software de modo que sea una herramienta más completa para la Gestión de Requisitos.

Referencias Bibliográficas

1. Anciano Martín, María J. *Gestión y Desarrollo de Requisitos en Proyectos Software.* 2003.

2. Anaya, Víctor y Letelier, Patricio. *SmarTTrace: Una Herramienta para Trazabilidad de Requisitos en Proyectos basados en UML.* Valencia: s.n.

3. Bringas González, Raquel y Gómez Díaz, Raquel. *Normalización y requisitos funcionales de la descripción archivística, una propuesta metodológica.* Salamanca: s.n., 2005.

4. García Fanjul, José y de la Riva Alvarez, Claudio. *Requisitos del software.* Oviedo: s.n., 2005.

5. Díez González, Oscar. *Safety y Requisitos No Funcionales.* Madrid: s.n., 2006.

6. Autores, Varios. *Proyecto Vulcano. Forja de proyectos software de calidad.* California: s.n., 2006.

7. Pérez Agusti, Yadira. *Modelado de negocio y gestión de requisitos como etapas imprescindibles para el desarrollo de los sistemas automatizados de información.* La Habana: s.n., 2008.

8. McDonald Landazuri, Bárbara A. *Definición de Perfiles en Herramientas de Gestión de Requisitos.* Madrid: s.n., 2005.

9. TRAC. *TRAC.* [En línea] [Citado el: 18 de Febrero de 2009.] http://trac.edgewall.org.

10. www.slideshare.net. *www.slideshare.net.* [En línea] [Citado el: 20 de Febrero de 2009.] http://www.slideshare.net/juan920924/sqlite.

11. Honor García, José Manuel. *TuriCiudad, un portal web del turismo de tu ciudad.* Málaga: s.n., 2007.

12. NetBeans UML Project. *NetBeans UML Project.* [En línea] [Citado el: 1 de Marzo de 2009.] http://uml.netbeans.org.

13. Torres Flores, Lizeth Carmina. *Establecimiento de una Metodología de Desarrollo de Software para la Universidad de Navojoa usando OpenUP.* Navojoa : s.n., 2008.

14. González, Carlos D. www.usabilidadweb.com.ar. *www.usabilidadweb.com.ar.* [En línea] [Citado el: 26 de Marzo de 2009.] http://www.usabilidadweb.com.ar/python.php.

15. Alvarez, Miguel Angel. www.desarrolloweb.com. *www.desarrolloweb.com.* [En línea] [Citado el: 15 de Abril de 2009.] http://www.desarrolloweb.com/articulos/25.php.

16. Prieto, Félix. *Patrones de diseño.* Valladolid: s.n., 2008.

17. Visconti, Marcello y Astudillo, Hernán. *Fundamentos de Ingeniería de Software.* Santiago: s.n.

18. Kynetia. *Kynetia.* [En línea] [Citado el: 20 de Mayo de 2009.] http://www.kynetia.es/calidad/diseno-de-pruebas.html.

Bibliografía

1. CSOFT. *CSOFT.* [Online] http://csoftintl.com/solutions_test_software.php.

2. *Guidelines for Requirements Management.* 2000.

3. Lycka Bonita. *Lycka Bonita.* [En línea] http://www.hachisvertas.net/blog/01/2008/11/12/832.

4. Fuentes, José Miguel. *Optimización del proceso de Gestión de Requisitos en el desarrollo de aplicaciones software.* 2007.

5. SoftwareProjects. *SoftwareProjects.* [En línea] http://www.softwareprojects.org/software-requirement-management-01.htm.

6. www.eclipse.org. *www.eclipse.org.* [En línea] http://www.eclipse.org/epf/general/OpenUP.pdf.

7. www.edgewall.org. *www.edgewall.org.* [En línea] http://pypi.python.org/pypi/Genshi/.

Anexos

Anexo 1.

CU 1. Gestionar requisitos.

Escenarios:

1. Adicionar requisito.
2. Modificar requisito.
3. Eliminar requisitos.

Se realizaron diagramas de secuencia para los escenarios 1 y 3.

CU 2. Obtener reporte.

Escenarios:

1. Obtener reporte por atributos.

Se realizó diagrama de secuencia para este escenario.

CU 3. Obtener vistas de matriz de trazabilidad.

Escenarios:

1. Características – Casos de Uso.
2. Necesidades de Usuario – Casos de Uso.
3. Necesidades de Usuario – Historia de Usuarios.
4. No Funcionales – Historia de Usuarios.
5. Características – No Funcionales.

Se realizó diagrama de secuencia para el escenario 1.

CU 4. Definir relaciones de trazabilidad. (Extendido de CU 3)

Escenarios:

1. Agregar relación de trazabilidad entre requisitos Características – Casos de Uso.
2. Eliminar relación de trazabilidad entre requisitos Características – Casos de Uso.
3. Agregar relación de trazabilidad entre requisitos Necesidades de Usuario – Casos de Uso.
4. Eliminar relación de trazabilidad entre requisitos Necesidades de Usuario – Casos de Uso.
5. Agregar relación de trazabilidad entre requisitos Necesidades de Usuario – Historia de Usuarios.
6. Eliminar relación de trazabilidad entre requisitos Necesidades de Usuario – Historia de Usuarios.
7. Agregar relación de trazabilidad entre requisitos No Funcionales – Historia de Usuarios.
8. Eliminar relación de trazabilidad entre requisitos No Funcionales – Historia de Usuarios.
9. Agregar relación de trazabilidad entre requisitos Características – No Funcionales.
10. Eliminar relación de trazabilidad entre requisitos Características – No Funcionales.

Se realizó el diagrama de secuencia para el escenario 1.

CU 5. Obtener vista jerárquica de requisitos.

Escenarios:

1. Obtener vista jerárquica de requisitos.

Se realizó el diagrama de secuencia para este escenario.

Glosario de términos

Matriz de trazabilidad: es una herramienta que ayuda a seguir, ordenar y mostrar como realizamos la validación de un sistema partiendo de los requerimientos de usuarios y mostrando en cada etapa de calificación como verificamos los mismos.

Gestión: es el conjunto de trámites que se llevan a cabo para resolver un asunto o concretar un proyecto.

Gestión de Proyectos: es la disciplina que se encarga de organizar y de administrar los recursos de manera tal que se pueda concretar todo el trabajo requerido por un proyecto dentro del tiempo y del presupuesto definido.

BD: Base de Datos.

Trazabilidad de Requisitos: habilidad para seguir la vida de un requisito en ambos sentidos, hacia sus orígenes o hacia su implementación, a través de las especificaciones generadas durante el proceso de desarrollo.

Plugin: complemento o extensión de hardware o software que constituye una aplicación que puede ser incorporada a otra con el objetivo de aportarle nuevas funcionalidades.

Stakeholders: cualquier persona o entidad que es afectada por las actividades de una organización; por ejemplo, los trabajadores de esa organización, sus accionistas, las asociaciones de vecinos, sindicatos, organizaciones civiles y gubernamentales.

IDE (Integrated Development Environment): es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

GUI (Graphical User Interface): la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

XML (Extensible Markup Language): es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Además se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

MS Word: Microsoft Word es un software destinado al procesamiento de texto.

Oracle, MySQL: sistemas de gestión de base de datos relacional.

Back - end database: es una base de datos que se accede por los usuarios externos indirectamente a través de una aplicación en lugar de programación de aplicaciones almacenados en la base de datos propia o por bajo nivel de manipulación de los datos (por ejemplo, a través de SQL de comandos).

Microsoft SQL Server: es un sistema de gestión de base de datos relacionales.

Escalabilidad: es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

HTML (**HyperText Markup Language):** es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

PHP (PHP Hypertext Pre-processor): es un lenguaje de programación interpretado diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servido.

CORBA (Common Object Request Broker Architecture): arquitectura común de intermediarios en peticiones a objetos, es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.
Wiki: es un sitio web cuyas páginas web pueden ser editadas por múltiples voluntarios a través del navegador web. Los usuarios pueden crear, modificar o borrar un mismo texto que comparten.

Web: es un sistema de documentos interconectados por enlaces de hypertext.

Tracd: es un servidor HTTP propio del Trac.

HTTP: El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol).

CGI (Common Gateway Interface): es un método para la transmisión de información hacia un compilador instalado en el servidor. Su función principal es la de añadir una mayor interacción a los documentos web que por medio del HTML se presentan de forma estática.

PostgreSQL: es un sistema de gestión de base de datos relacional orientado a objetos de software libre.

RUP (Rational Unified Process): es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

UML (Unified Modeling Language): es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

Tk: es una aplicación software libre, multiplataforma, es una biblioteca de elementos básicos para construir una interfaz gráfica (GUI).

GTK: es una biblioteca que contiene los objetos y funciones para crear interfaces gráficas de usuario.

Qt: es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario.

CVS (Concurrent Versions System): es una aplicación informática que implementa un sistema de control de versiones, mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto y permite que distintos desarrolladores colaboren.

LaTeX: es un lenguaje de marcado y un sistema de preparación para documentos.

Telnet (TELecommunication NETwork): es un protocolo de red que sirve para acceder mediante una red a otra máquina, para manejarla remotamente compo si estuviéramos sentados delante de ella.