Universidad de las Ciencias Informáticas

Facultad 6

Título: "Interfaz Web para la Plataforma de Tareas Distribuidas T-Arenal 2.0"

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas

Autoras:

Ana Li Peña Concepción

Yeimy Díaz Cruz

Tutores:

MSc. Longendri Aguilera Mendoza

Ing. César Raúl García Jacas

Junio 2011

"*La ciencia tiene una característica maravillosa, y es que aprende de sus errores, que utiliza sus equivocaciones para reexaminar los problemas y volver a intentar resolverlos, cada vez por nuevos caminos..*"

Ruy Perez Tamayo

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los _____ días del mes _____ del año _____.

Yeimy Díaz Cruz

Firma de la Autora

Ana Li Peña Concepción

Firma de la Autora

Ing. César Raúl García Jacas

Firma del Tutor

Msc. Longendri Aguilera Mendoza

Firma del Tutor

Datos del Contacto

Autoras:

Ana Li Peña Concepción.
Universidad de las Ciencias Informáticas.
e-mail: alpena@estudiantes.uci.cu

Yeimy Díaz Cruz.
Universidad de las Ciencias Informáticas.
e-mail: ydcruz@estudiantes.uci.cu

Tutores:

Longendri Aguilera Mendoza.
Licenciado en Ciencias de la Computación.
Master en Bioinformática.
Profesor Asistente.
e-mail: loge@uci.cu

César Raúl García Jacas.
Ingeniero en Ciencias Informáticas.
e-mail: crjacas@uci.cu

Agradecimientos

Le agradezco a mi madre por ser la razón de mi existencia, mi luz y guía; a mi abuela por ser mi fuerza de empuje ante cualquier cosa además por confiar en mi aún cuando yo no lo hacía. A mi padre y mis hermanos por su apoyo y los consejos brindados, a mi familia por todo su cariño, interés y entrega.

A César por estos cinco años en la universidad donde has sido más que un tutor un gran amigo, alguien con quien he podido contar sin importar nada. A Yeimy por ser la mejor compañera que cualquiera pudiera desear.

A todos los profesores que han aportado su grano de arena en mi formación profesional especialmente Longendri por las dudas aclaradas y por estar ahí cuando lo necesitábamos y Adisley muchas gracias por tu preocupación.

A mis amistades Sailin, Lipsy, Felipe, Eugenio, Herryman y Alberto, gracias por escucharme de vez en vez con mis locuras. A Betty por ayudarme en los momentos más difíciles y mostrarme ese mundo tan amplio de las posibilidades. A Carlos Alain muchas gracias por este tiempo compartido, tu constante vigilia y cariño.

En fin le agradezco a todos los que en algún momento se me acercaron y me dijeron ¿Hola, cómo va la tesis?, a todos los que me han acompañado estos años y a ustedes les digo simplemente: gracias.

Ana Li

Agradecimientos

A mis padres que gracias a ellos existo, he podido llegar hasta aquí y me han apoyado siempre en todas las decisiones que he tomado en la vida. Porque he tratado de que todo el esfuerzo y sacrificio que han hecho por mí no fuera en vano.

A mi hermana porque la quiero mucho, porque es ante todo una gran amiga, por su preocupación. Esta alegría también es tuya. Porque sé que este día llegará igualmente para ti, tienes de sobra para lograrlo.

A mis abuelos Ala, Amado, Isabel y Norberto, que de una forma u otra se han graduado junto conmigo, porque han vivido los mismos momentos de tensión y felicidad a lo largo de estos cinco años, por siempre estar pendiente de mí, de mis problemas y de sus soluciones. Porque son parte de lo más preciado que tengo en el mundo, por consentirme y quererme tanto.

A mis tíos, mis primos y familia en general, que también han aportado su granito de arena en este logro.

A Tico, por su amor, cariño, consejos, cuidado en este tiempo y por mostrarme que hay diferentes formas de ver la vida. Por los buenos momentos que no se olvidan. Por lo aprendido a su lado.

A mis tutores y en especial a César por su ayuda, consejos, enseñarme gran parte de lo que he aprendido y porque más que tutor es amigo, que ha estado presente desde mis primeros pasos en la Universidad.

A todos los profesores que han contribuido en mi formación profesional, en especial a los del departamento de Ingeniería de Software: Garnache, Aliosmi, Lázaro y Diana, que cada vez que los necesité pude contar con ellos.

A Yirina, mi confidente desde que entré a la escuela; a Yanet Rosales por considerarme su amiga y ser como es conmigo; a Yanet Pérez por acompañarme y por la confianza que me tiene. A Manuel, Orea, Georvys por ser buenos amigos y en general a todos mis compañeros de aula, del laboratorio y los que me ayudaron al desarrollo de este trabajo: Andry, Reisel, Osvel, Yadrian y Alberto.

A mis amistades de Cárdenas: Iliana, Jeydi, Hanny,Yuneisis, Liatny, Aylin, Jenny María, Lien, Yulien, Wendy, Heriberto, por su amistad incondicional, con las cual siempre pude contar cuando lo necesité.

A mi cuñadito Yunior, te quiero mucho.

A Ana Li por todo el trabajo que hicimos juntas para llegar hasta aquí.

Yeimy

Dedicatoria

A las dos mujeres más importantes de mi vida, mi madre y abuela.

Ana Li

A mi familia por ser mi inspiración, sostén y fuerza.

Yeimy

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se desarrolló la Plataforma de Tareas Distribuidas, T-arenal v1.0, con el objetivo de ofrecer una alternativa para el procesamiento paralelo de datos; uniendo en un solo conjunto un grupo de estaciones de trabajo para este fin. Esta primera versión cuenta con una interfaz de escritorio, que tiene como inconveniente que el usuario no siempre poseería la versión más actualizada y que además tendría que traerla consigo para ejecutarla en la estación de trabajo donde se encontrara. En la segunda versión de T-arenal, se desarrolló un nuevo front-end de escritorio, que estaría a disposición de los usuarios mediante el contenedor GridSphere y la tecnología Java Web Start. Esto nunca se logró por no contar con los recursos necesarios, persistiendo los mismos inconvenientes de la versión anterior. Por tales motivos en este trabajo se presenta la interfaz Web para la Plataforma de Tareas Distribuidas v2.0 como solución a los problemas anteriormente planteados. Para este desarrollo se utilizó el framework RichFaces para el diseño y construcción de la interfaz gráfica de usuario y la tecnología Java Server Pages (JSP) para la gestión de datos en el lado del servidor.

Palabras Claves: T-arenal, Front-end, Gridsphere, Java Web Start, Java Server Faces, RichFaces.

Índice general

Índice de figuras

Introducción

Los inicios de la informática actual se enfocan hacia los finales de la década del 40, cuando se construían computadoras utilizando la tecnología disponible en esa época. La interacción entre las personas y estos ordenadores se hacían a base de tarjetas perforadas o recableando las conexiones entre sus componentes.

Las primeras máquinas, como el Mark I y el ENIAC, ocupaban un amplio espacio, pesaban varias toneladas y se demoraban una decena de segundos para realizar una operación matemática simple como la división. A través de los años, los avances científico-técnicos han hecho posible pasar de aquellas gigantescas máquinas a los ordenadores actuales, los cuales son más potentes, versátiles y realizan millones de operaciones por segundo [1]. La interacción entre estas modernas computadoras y los usuarios son realizadas mediante interfaces gráficas.

La elaboración de una interfaz gráfica de usuario bien diseñada exige una gran dedicación, pues generalmente son grandes, complejas, difíciles de implementar, depurar y modificar. En la actualidad las interfaces de manipulación directa (también llamadas interfaces gráficas de usuario, GUI por sus siglas en inglés) son prácticamente universales. Las interfaces que utilizan ventanas, íconos y menús se han convertido en estándares en las aplicaciones informáticas que se desarrollan [2].

En la Universidad de las Ciencias Informáticas (UCI), creada en Cuba en el año 2002, está presente la Bioinformática como una rama de investigación y producción de software. Varios son los proyectos que se desarrollan con centros del Polo Científico de Cuba. La mayoría de estos proyectos necesitan realizar una gran cantidad de cálculos que demoran un tiempo excesivamente largo en una sola computadora.

Para satisfacer esta demanda de cómputo, se desarrolló la Plataforma de Tareas Distribuidas T-arenal v1.0, que tiene como objetivo unir en un solo conglomerado un conjunto de PCs distribuidas en una red LAN. La primera versión del sistema ha dado buenos resultados y ha sido de gran utilidad en varios proyectos. La misma cuenta con una interfaz de escritorio para la realización de las diferentes funcionalidades que el sistema brinda [3]. Esta interfaz tenía como inconveniente que el usuario no siempre poseería la versión más

actualizada, además de que tendría que traerla consigo para ejecutarla en la estación de trabajo donde se encontrara.

Por motivos de refinamiento se decidió mejorar la arquitectura de T-arenal v1.0, provocando el desarrollo de la segunda versión. En esta versión se desarrolló un nuevo front-end[1] de escritorio, que se pondría a disposición de los usuarios finales haciendo uso del contenedor de portlet Gridsphere [4] y la tecnología Java Web Start [5]. Esta solución requería la instalación y configuración del Midllewere Grid Globus Toolkit. Esto último nunca se logró por falta de recursos, por lo tanto se continuó utilizando la interfaz de escritorio para la versión 2.0, manteniendo los mismos inconvenientes que la versión precedente.

Por lo anteriormente expuesto el presente trabajo se circunscribe en las *interfaces de usuario Web,* específicamente en la *interfaz Web para la Plataforma de Tareas Distribuidas T-arenal v2.0.* Por lo tanto se obtiene como **problema a resolver,** ¿cómo garantizar la disponibilidad de acceso a la Plataforma de Tareas Distribuidas T-arenal v2.0?

Objetivo General

Desarrollar una interfaz Web para la Plataforma de Tareas Distribuidas T-arenal 2.0 que garantice mayor disponibilidad de acceso.

Objetivos específicos

1. Definir los requisitos para el desarrollo de una aplicación Web para interactuar con la Plataforma de Tareas Distribuidas T-arenal 2.0.
2. Diseñar una aplicación Web para la Plataforma de Tareas Distribuidas T-arenal 2.0.
3. Implementar la aplicación diseñada.
4. Realizar pruebas a la aplicación implementada.

Tareas

1. Estudio de las principales características de las aplicaciones Web.
2. Elaboración del modelo de dominio correspondiente a una aplicación Web.

[1] parte de las aplicaciones informáticas que interactúan con el usuario

3. Definición de los requisitos funcionales y no funcionales de la aplicación.
4. Desarrollo del diagrama de casos de uso del sistema.
5. Descripción de los casos de uso del sistema.
6. Descripción de la arquitectura de la aplicación.
7. Descripción de los patrones de diseño utilizados.
8. Desarrollo del diagrama de clases del diseño.
9. Implementación de la aplicación.
10. Realización de pruebas de caja negra a la aplicación desarrollada.

Estructura del documento

El presente documento se estructura en un Resumen, Introducción, cuatro capítulos que constituyen el cuerpo de la tesis, Conclusiones, Recomendaciones, Referencias bibliográficas y Anexos. Los capítulos son:

CAPITULO 1. Fundamento teórico: En este capítulo se realiza y muestra un estudio acerca del desarrollo de las interfaces gráficas de usuario. Se hace una breve valoración de las técnicas, plataformas, librerías y procedimientos utilizados para dar cumplimiento al desarrollo de la Interfaz Web para la Plataforma de Tareas Distribuidas T-arenal 2.0. Finalmente se hace una descripción de las herramientas y tecnologías a utilizar para desarrollar la solución propuesta.

CAPITULO 2. Características del sistema: En este capítulo se describen las principales características del sistema a desarrollar, sus requisitos funcionales y no funcionales, y los actores que intervienen en el mismo así como el modelo de dominio correspondiente al mismo. Se presenta el diagrama de casos de usos del sistema, así como una breve descripción de los casos de usos identificados.

CAPITULO 3. Diseño del sistema: En este capítulo se realizará una descripción del estilo arquitectónico utilizado para el desarrollo del sistema, se describirán los patrones de diseño empleados y los diagramas de clases de diseño e interacción de los casos de usos principales. Se mostrará el diagrama de despliegue correspondiente al sistema.

CAPITULO 4. Implementación y Prueba del sistema: En este capítulo se expondrá una explicación de la implementación del sistema en términos de componentes. Se ilustrarán los principales resultados obtenidos. Además, los casos de prueba para los casos de uso críticos del sistema.

Capítulo 1

Fundamento Teórico

Resumen del capítulo

En este capítulo se realiza una exposición detallada de los principales conceptos relacionados con el objeto de estudio, así como las tecnologías, herramientas y metodologías, con el fin de proponer una solución al problema planteado.

1.1. Interfaz de Usuario

El software puede agruparse de manera general en dos clases: los programas de sistema, los cuales controlan la operación de la computadora, y los programas de aplicación, los cuales resuelven problemas para sus usuarios [6].

Figura 1.1: Agrupación de software

Las interfaces de usuario entrarían en la clase de programas de aplicación, debido a que de un modo genérico, son dispositivos de adaptación a dos entidades distintas, con el objetivo técnico de la adaptación entre la complejidad de los sistemas y las capacidades del ser humano. Su creación ha sido un área del desarrollo de software que ha evolucionado dramáticamente a partir de la década de los setenta. Puede

definirse la interfaz de usuario como el vínculo entre el usuario y el programa de computadora, el cual contiene un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa [7].

Gracias a las interfaces, los usuarios no necesitan comprender la máquina física. Se crea la ilusión de que el ordenador es muy sencillo. Este conjunto de mecanismos de diálogo hombre-máquina que ocultan al usuario la complejidad del ordenador se conoce como "Ilusión de usuario". De esta manera se da la paradoja de que los ordenadores, siendo cada vez más complejos, parecen ser cada vez más simples. Es porque el usuario solo "ve" una máquina virtual hecha a medida de la complejidad que él comprende [8].

Algunas definiciones de interfaz gráfica de usuario son:

"La interfaz de usuario es un conjunto de protocolos y técnicas para el intercambio de información entre una aplicación computacional y el usuario. La IU es responsable de solicitar comandos al usuario, y de desplegar los resultados de la aplicación de una manera comprensible. La IU no es responsable de los cálculos de la aplicación, ni del almacenamiento, recuperación y transmisión de la información" [9].

"...tipo de entorno que permite al usuario elegir comandos, iniciar programas, ver listas de archivos y otras opciones utilizando las representaciones visuales (iconos) y las listas de elementos del menú (...); es aquella parte de un programa que comunica al usuario con el programa mediante representaciones gráficas..." [10].

1.2. Evolución de la Interfaz de Usuario

Con la aparición de los ordenadores y su posterior popularización, se hizo necesario la creación de mecanismos que hicieran posible la comunicación de los usuarios con los sistemas operativos y las aplicaciones que se ejecutaban en ellos, siendo estos sistemas las interfaces de usuario.

Figura 1.2: Interfaz de usuario con modo de comandos.

Estas interfaces se crearon en un principio en modo comandos, pero su utilidad práctica era limitada, ya que no eran aptas para usuarios normales que no poseían conocimientos avanzados de informática. Fue necesario entonces crear interfaces basadas en iconos y menús, accesibles por medio del ratón, surgiendo los entornos de ventanas tales como Windows o MAC.

Cuando se produjo el sorprendente fenómeno que llamó a revolucionar la comunicación entre seres humanos -Internet y la WWW, con la aparición de la web se hizo posible que cualquier persona pudiera ofrecer información particularizada a los demás y encontrar documentos interactivos sobre cualquier tema, relacionados unos con otros mediante enlaces que permitían saltar de página en página alrededor del mundo.

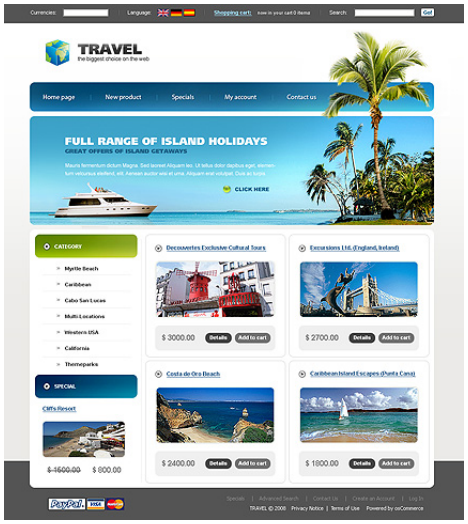

Figura 1.3: Interfaz de usuario en una página web.

Las páginas web supusieron la aparición de las interfaces web, interfaces gráficas de usuario con elementos comunes de presentación y navegación que pronto se convirtieron en estándares. Buscar una homogeneidad entre los millones de páginas web que existen actualmente en Internet ha provocado que su diseño haya evolucionado con el tiempo hacia un esquema general perfectamente definido, ofreciendo unas interfaces con un conjunto de componentes gráficos funcionales y similares, que hacen posible que sin importar la experiencia del usuario que accede a un sitio web la comunicación entre ellos sea posible y efectiva [11].

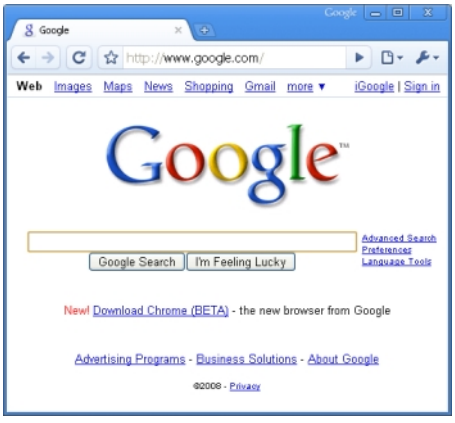

Figura 1.4: Interfaz de un navegador y de una página web.

1.3. Interfaces Gráficas e Interfaces Web de usuarios

El diseño de Interfaces Gráficas para Usuarios ((Graphical User Interface (GUI)) ha seguido durante mucho tiempo el estándar elaborado por diseñadores de sistemas operativos, tales como Microsoft o Apple, o por grupos industriales, tales como OSF. Estos convenios resultan obvios en la mayoría de las aplicaciones de software.

Tradicionalmente, en las interfaces gráficas de usuario, la parte superior de la pantalla es el hogar de los menús principales de un programa, así que, ¿por qué no hacer lo mismo en un sitio Web?. Dado que la forma más común de leer una típica página Web es de izquierda a derecha y de arriba hacia abajo, esta es una buena posición para la exploración [12].

La Interfaz Gráfica de Usuario utiliza imágenes y objetos gráficos para representar la información y las acciones disponibles en la interfaz, lo que posibilita un mejor entendimiento del usuario con aplicaciones informáticas. En la actualidad, son comunes las GUIs que usan además dispositivos apuntadores como el ratón o mouse e incluyen múltiples elementos gráficos intuitivos como iconos o botones.

Las interfaces web son interfaces gráficas de usuario con elementos comunes de presentación y navegación que deben servir de intermediarias entre usuarios genéricos, no acostumbrados al uso de aplicaciones informáticas, sistemas de información y procesos transaccionales, facilitando así la localización de la información deseada, el entendimiento claro de las funcionalidades ofrecidas, la realización práctica de tareas específicas por parte de los usuarios y la navegación intuitiva por las diferentes páginas que conforman el sitio web.

Una ventaja significativa es que las aplicaciones web deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux y otros sistemas operativos, la aplicación web se escribe una vez y se ejecuta igual en todas partes.

1.4. Características

Una interfaz debe cumplir las características según [13]:

- Naturalidad. El nuevo sistema automatizado debe tender a ser lo más similar al antiguo.
- Facilidad de aprendizaje y uso, dos aspectos que no siempre van unidos.
- Consistencia. La interfaz debe mantener uniformidad en cuanto a estilo, vocabulario, etc.

Naturalidad

Una interfaz es natural, cuando provoca al usuario sentimientos de "estar como en casa". Todo trabajador tiene:

- Una forma de actuar;
- Una forma de organizarse;
- Un vocabulario propio para las tareas habituales;
- Un entorno que ya domina, al que está acostumbrado y del que, tal vez, le sea difícil salir.

Facilidad de aprendizaje y uso

Proporcionar al usuario un sistema de ayuda potente. Pero el sistema de ayuda puede ser un obstáculo una vez que se domine el producto (esta ayuda no debe ser automática). Para disfrutar de esta característica, la interfaz debe incorporar:

- Administración de perfiles de usuario. Según el grado de perfil, la interfaz ejecutará unas u otras acciones.
- Mecanismos de realimentación que proporcionen al usuario información sobre la ejecución actual del trabajo.
- Mecanismos de prevención de desastres.
- Sistemas de ayuda: Tratan de evitar que el usuario tenga que acceder a los manuales para resolver una duda puntual. Los mejores sistemas de ayuda son los que se denominan "sensibles al contexto". Son capaces de determinar la circunstancia que origina la petición de ayuda y proporcionar un auxilio muy concreto sobre la materia que interesa.

Consistencia

Debe mantenerse una uniformidad a lo largo de toda la extensión de la interfaz: modo de operación y diseño. Si cada componente actúa con distinta filosofía, obliga al usuario a cambiar la mentalidad de trabajo.

1.5. Diseño Web centrado en el usuario

El Diseño Web Centrado en el Usuario se caracteriza por asumir que todo el proceso de diseño y desarrollo del sitio web debe estar conducido por el usuario, sus necesidades, características y objetivos. Centrar el diseño en los usuarios (en oposición a centrarlo en las posibilidades tecnológicas o en nosotros mismos como diseñadores) implica involucrar desde el comienzo a los usuarios en el proceso de desarrollo del sitio; conocer cómo son, qué necesitan, para qué usan el sitio; testear el sitio con los propios usuarios; investigar cómo reaccionan ante el diseño, cómo es su experiencia de uso; e innovar siempre con el objetivo claro de mejorar la experiencia del usuario.

El proceso de Diseño Web Centrado en el Usuario propuesto en este trabajo se divide en varias fases o etapas, algunas de las cuales tienen carácter iterativo. Sirva como aproximación el siguiente esquema:

Etapas del proceso

1. Diseño
    - Modelado del usuario
    - Diseño conceptual
    - Diseño visual y definición del estilo
    - Diseño de contenidos
2. Prototipado
3. Evaluación
    - Método por inspección: evaluación heurística
    - Método de test con usuarios
4. Implementación y lanzamiento
5. Mantenimiento y seguimiento
    - Opiniones de los usuarios
    - Comportamiento del usuario y uso del sitio [14].

Como indica el esquema, las fases de "diseño", "prototipado" y "evaluación" son cíclicas e iterativas. Esto quiere decir que todo lo que se diseñe debe ser constantemente evaluado a través de su prototipado, para así poder corregir errores de usabilidad desde los primeros momentos del desarrollo. Evaluar el sitio web únicamente una vez finalizado su desarrollo haría mucho más costosa la reparación de errores de usabilidad, ya que siempre es más económico reconducir un diseño que rediseñar completamente el sitio.

Todo proyecto debe comenzar por una correcta planificación. En esta etapa se identifican los objetivos del sitio, así como las necesidades, requerimientos y objetivos de la audiencia potencial. Confrontando esta información se definen los requisitos del sitio web, entre los que podemos contar los requisitos técnicos (back-end y front-end), recursos humanos y perfiles profesionales necesarios, y adecuación del presupuesto disponible.

Se trata de establecer un equilibrio entre lo que puede ofertar el proveedor y lo que necesita el usuario. El sitio web - sus contenidos y diseño - debe cumplir precisamente este cometido: servir de medio para la consecución de objetivos por parte de proveedor y usuario. Como se puede ver, la etapa de planificación se basa casi completamente en la recogida, orden y análisis de toda la información posible, con el objetivo de tener una base sólida sobre la que poder tomar decisiones de diseño en las siguientes etapas del proceso.

1.6. Componentes de una Interfaz Web. Cuerpo de una página

El cuerpo de la página es la parte de la interfaz web que presenta a los usuarios información específica sobre un tema concreto. Por lo tanto, es la parte de la página que la identifica e individualiza frente a las demás de un sitio web.

Al ser la parte más importante de la interfaz, el espacio destinado a ella debe ser el mayor de todos, ocupando generalmente entre el 50 % y el 85 % del total. Su ubicación es siempre central, bajo el dintel (si lo hay) y al lado del menú lateral de navegación (si lo hay). Los contenidos específicos del cuerpo de la página variarán según sea una página textual, un formulario, una ficha, una tabla o una página mixta.

Por ejemplo, es habitual que el cuerpo central lleve un título que identifique claramente la página a la que ha accedido el usuario. Este título se situará en la parte superior de esta zona, y puede ser reforzado mediante un menú de navegación. El tamaño de las letras del título de página debe ser superior al del resto de los contenidos, de tal forma que destaque sobre ellos, efecto que puede aumentarse dando al mismo un color que le aporte un mayor peso visual. En este caso no es imprescindible que el título de página sea de mayor tamaño, siempre que por su contraste de colores destaque lo suficiente.

También es común diferenciar visualmente el cuerpo central del resto de la página, bien usando para el mismo un color diferente, bien dejando a su alrededor espacios vacíos que lo separen de forma clara del resto de los elementos de la interfaz. Todos los elementos gráficos que situemos dentro del cuerpo de página deben presentar un aspecto similar al del resto de elementos de la interfaz, respetando sus estilos generales de diseño [15].

1.7. Elementos Interactivos en la Interfaz Gráfica

Los dispositivos de interfaz humana son los diseñados para conectar alguna parte del cuerpo del ser humano con la interfaz gráfica de modo que puedan ser entrados los datos en el sistema. Normalmente son dispositivos que permiten introducir directamente, y en tiempo real, información de "orientación" y "acción" al ordenador sincronizado simultáneamente con una interfaz gráfica [16].

Los dispositivos de interfaz humana como el ratón, pueden representar en la interfaz gráfica gestos físicos y movimientos, como "apuntar", "pulsar", "arrastrar", "trasladar" o "mover" de forma metafórica que de otro modo sería muy complejo simular.

La interfaz humana forma actualmente una parte indisoluble respecto a la interfaz gráfica de usuario. Son partes interconectadas de un mismo paradigma de interacción, donde se necesitan uno al otro indispensablemente para que la interacción con el sistema se realice adecuadamente.

Existen diferentes tipos de interfaces humanas las cuales han sido desarrolladas paralelamente a lo largo de la historia de la interfaz gráfica. Las más importantes han sido el teclado, el ratón de ordenador, el trackball (bola), el cursor táctil (touch pad), la tableta gráfica y el Joystick.

1.8. Arquitectura de la Información

El término Arquitectura de la Información (AI) fue utilizado por primera vez por Richard Saul Wurman en 1975, quién la define como: *"El estudio de la organización de la información con el objetivo de permitir al usuario encontrar su vía de navegación hacia el conocimiento y la comprensión de la información".*

Esta definición de actividades se ha visto más detenidamente establecida a raíz de la publicación de la primera obra de referencia que ha llevado el rigor y el método de la Ciencia de la Información a la Web. Si nos ceñimos exclusivamente a la AI en el campo de la Web puede sernos de más fácil comprensión la siguiente definición: *"El arte y la ciencia de estructurar y clasificar sitios Web e intranets con el fin de ayudar a los usuarios a encontrar y manejar la información"* [17].

El concepto AI no solo engloba la actividad de organizar información, sino también el resultado de dicha actividad. La AI de un sitio web, como resultado de la actividad, comprende los sistemas de organización y estructuración de los contenidos, los sistemas de rotulado o etiquetado de dichos contenidos, y los sistemas de recuperación de información y navegación que provea el sitio web.

1.9. Metodología y Tecnologías

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán [18]. Una creencia errónea es que el desarrollo del software debe girar en torno a las habilidades del personal altamente calificado que es el que sabe cómo hacer el trabajo y hacerlo bien. Por lo que es necesario un proceso que involucre a todos los interesados en el desarrollo. Todo esto unido a una correcta selección de las herramientas, métodos, técnicas y procedimientos contribuye a obtener un producto de elevada calidad.

Metodología de desarrollo de software

En las dos últimas décadas las notaciones de modelado y posteriormente las herramientas pretendieron ser las "balas de plata" para el éxito en el desarrollo de software. Hasta hace poco el proceso de desarrollo llevaba asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Este esquema "tradicional" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos). Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del buen hacer de la ingeniería del software, asumiendo el riesgo que ello conlleva. Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto. Entre

ellas podemos citar RUP [19], XP [20],OpenUP [21].

Esta última es una de las metodologías ágiles de desarrollo de software, que ofrece las mejores prácticas de una variedad de líderes en ideas sobre producción de software y comunidades de desarrollo. Preserva las características esenciales de RUP que incluye el desarrollo interactivo, casos de uso y escenarios de conducción de desarrollo, la gestión de riesgo y el enfoque centrado en la arquitectura. OpenUP/Basic es la forma más ágil y ligera de OpenUP, y se basa en una donación de código abierto al proceso de contenido, conocido como el Proceso Unificado Básico (BUP).

La mayoría de las partes de RUP han sido excluidas de esta metodología y muchos elementos se han fusionado, siendo el resultado un proceso mucho más sencillo que coincide con los principios de RUP. OpenUP/Basic es aplicable a proyectos pequeños con grupos de 3 a 6 personas interesadas en el desarrollo rápido e interactivo. Además, OpenUP/Basic define un proceso de desarrollo de software mínimo y completo. Mínimo porque solamente lo fundamental es incluido dentro del proceso, y completo porque define un conjunto de componentes que guían y definen dicho proceso de desarrollo hasta la obtención del producto. Por las razones antes expuestas y por decisión del Grupo de Bioinformática de la UCI, OpenUP/Basic constituye la metodología seleccionada para el desarrollo de la aplicación que se propone en el presente trabajo.

Java Server Faces (JSF)

Es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE [22]. JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entradas, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Dos librerías de etiquetas personalizadas para Java Server Pages que permiten expresar una interfaz Java Server Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Beans administrados.

Java Server Faces, es la última tecnología Java para interfaces de usuario. Las principales ventajas que ofrece son:

- Uniformidad y coherencia de la aplicación completa, reduciendo en gran medida el costo de formación de usuarios
- La rapidez de desarrollo y reducción de los costos. El desarrollo que suponga una ventaja para el negocio estará disponible en menos tiempo.
- La robustez, ya que es la propia tecnología la responsable de validar y transmitir la información, dejando que el programador se limite a definir el negocio.

Esta tecnología es 100 % open source software (software de código abierto) y la apoyan importantes comunidades de usuarios (Apache Foundation, por ejemplo) y grandes empresas (Sun Microsystems entre otras).

RichFaces 3.2

Es una biblioteca de componentes para JSF y un avanzado framework para la integración de AJAX con facilidad en la capacidad de desarrollo de aplicaciones de negocio. Los componentes RichFaces vienen listos para su uso sin tener que hacer muchas configuraciones, por lo que los desarrolladores pueden ahorrar tiempo de inmediato para aprovechar las características de los componentes, para crear aplicaciones web que proporcionan mejoras en gran medida en la experiencia del usuario más fiable. También incluye un fuerte apoyo para el soporte de temas de aplicaciones JSF y aprovecha al máximo los beneficios de JSF framework, incluyendo la validación y conversión junto con la gestión estática y dinámica de los recursos [23].

JSP

JSP es un acrónimo de Java Server Pages, que es lo mismo que decir algo como Páginas de Servidor Java. Es una tecnología orientada a crear páginas web con programación en Java. Con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP podrán escribirse con un editor HTML/XML habitual [24].

CSS

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de definir un mecanismo que permitiera aplicar

de forma consistente diferentes estilos a los documentos electrónicos.

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, y demás. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, entre otros [25].

Apache Tomcat

Apache Tomcat es una implementación de software de código abierto de Java Servlet y tecnologías JavaServer Pages. Las especificaciones de Java Servlet y JavaServer Pages se desarrollan bajo la Java Community Process. Apache Tomcat es desarrollado en un entorno abierto y participativo y publicado bajo la licencia Apache versión 2. Esto tiene la intención de ser una colaboración de los mejores desarrolladores de su clase de todo el mundo [26].

El proyecto Apache comenzó en 1995 como un proyecto para la creación de un servidor Web de código abierto. Con los años se le añadieron muchas funcionalidades como la habilidad de un solo servidor de mantener múltiples sitios Web virtuales y esquemas de autenticación. Actualmente es uno de los servidores más usados para la creación de sitios Web comerciales. Está disponible para una gran cantidad de plataformas como GNU/Linux, Mac OS, Mac OS X Server, Netware, Open Step/Match, UNIX, Solaris, SunOS, UnixWare, Windows entre otras [27].

Lenguaje de modelado

Para dar solución al problema planteado se usará como lenguaje de modelado UML [28]. El UML (Unified Modeling Language, Lenguaje Unificado de Construcción de Modelos) nació como una notación estándar de la construcción de modelos [29]. Es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson,

quienes trabajaban en empresas distintas durante la década de los ochenta y principios de los noventa y cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos [30]. El lenguaje para modelamiento unificado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo.

UML no es un método de desarrollo. No nos va a indicar cómo pasar del análisis al diseño y de este al código. No son una serie de pasos que te llevan a producir código a partir de unas especificaciones.

UML al no ser un método de desarrollo es independiente del ciclo de desarrollo que vayas a seguir, puede encajar en un tradicional ciclo en cascada, o en un evolutivo ciclo en espiral o incluso en los métodos ágiles de desarrollo [31].

Herramientas CASE

Las herramientas CASE5 son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo del sistema de información, completamente o en algunas fases. Como ejemplo de herramientas CASE tenemos MagicDraw, [32], Rational Rose [33] , Umbrello [34], Visual Paradigm for UML, ArgoUML [35]y otras. Visual Paradigm es una herramienta CASE que soporta la última versión del Lenguaje de Modelado Unificado (UML), la Notación del Proceso de Modelado de Negocio (BPMN), y genera código para un gran número de lenguajes de programación. Además brinda una versión libre para uso no comercial.

La herramienta fue desarrollada para una amplia gama de usuarios incluyendo ingenieros de software, analistas de sistemas, analistas del negocio y arquitectos de sistemas. Permite la integración con varias herramientas de Java, y brinda además una gran interrelación con otras herramientas CASE como Rational Rose. Por decisión del Grupo de Bioinformática la Herramienta CASE a utilizar es Visual Paradigm for UML.

IDE y Lenguaje de programación

Lenguaje de programación

El lenguaje de programación seleccionado es Java. La razón principal de esta selección es su independencia respecto a la plataforma y la arquitectura de red, por eso, una aplicación escrita en Java puede ejecutarse en cualquier sistema.

En los primeros días de Java, gran parte de sus críticas se originaban por su pobre rendimiento respecto

a lenguajes nativos como C y Fortran. Actualmente se han producido enormes mejoras en el rendimiento de la Máquina Virtual de Java (JVM6).

Estas mejoras han dado lugar a que la ejecución de la JVM, sea comparable a la de otros lenguajes nativos. Otro de los aspectos más importante de Java es el modelo de seguridad. Este simplifica enormemente la implementación de una estricta política de seguridad para una aplicación Java, haciendo posible que las aplicaciones puedan cargar dinámicamente código, sin tener que preocuparse por las posibles implicaciones para la seguridad. Además de las características mencionadas con anterioridad, Java constituye un lenguaje "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico" [10].

El IDE Eclipse

El IDE Eclipse es un entorno de desarrollo de Java que emplea módulos (en inglés plugin) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes de programación además de Java, así como introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo.

Este IDE fue seleccionado principalmente por su potente editor de código, la interfaz amigable que presenta y la existencia en el ciberespacio de una gran cantidad de plugins que hacen del IDE una herramienta potente. Además es un software libre y multiplataforma [36].

1.10. Conclusiones

Las interfaces de usuarios son la presencia del software y es el punto más importante para valorar la utilidad y funcionalidad del software. La Interfaz Web para la Plataforma de Tareas Distribuidas se desarrollará teniendo en cuenta los principios básicos que debe cumplir una interfaz web y sus características como naturalidad, facilidad de aprendizaje, uso y consistencia. Se pondrán en práctica todos aquellos elementos interactivos para que el usuario se sienta cómodo y le sea fácil la interacción con la Plataforma de Tareas Distribuidas. Se clasificará la información que será mostrada al usuario para tener una mejor organización.

La metodología a emplear será OpenUP ya que se estará trabajando en un proyecto pequeño, lo que ayudará a mantener una estructura superior y a seguir utilizando la metodología que hasta ahora ha dado

buenos resultados en el grupo de Bioinformática, con Visual Paradigm como la herramienta CASE. Se trabajará con JSF, un framework para desarrollar interfaces de usuario en Java. La biblioteca RichFace será utilizada ya que ésta nos brinda una gran variedad de componentes para JSF y permite la integración con Ajax. Las hojas de estilo se emplearán para el posicionamiento de los componentes en la página web. El lenguaje de programación a usar será Java con el IDE Eclipse y como servidor de aplicaciones Apache Tomcat.

Capítulo 2

Características del Sistema

En este capítulo se describen las principales características del sistema a desarrollar, sus requisitos funcionales y no funcionales, la realización del modelo de dominio y los actores que intervienen en el mismo. Además, se presenta el diagrama de casos de uso del sistema, así como una breve descripción de los casos de uso identificados.

Breve descripción del sistema

El sistema a desarrollar permitirá interactuar con la Plataforma de Tareas Distribuidas, la cual posibilitará realizar todas las utilidades de administración, creación de tareas y monitoreo de recursos, según el privilegio del usuario autenticado. La interfaz web le brindará al usuario la ventaja de poder acceder desde cualquier lugar debido a que estará siempre disponible y con la última versión.

2.1. Modelo de dominio

Debido a que la problemática a resolver presenta una relativa simplicidad del entorno donde está enmarcado el sistema y existe cierto conocimiento acerca de su funcionamiento, no es necesario realizar un Modelo de Negocio, siendo suficiente un Modelo de Dominio (Modelo Conceptual). Este último es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. El diagrama de clases del modelo de dominio del presente trabajo se muestra en la figura 2.1.

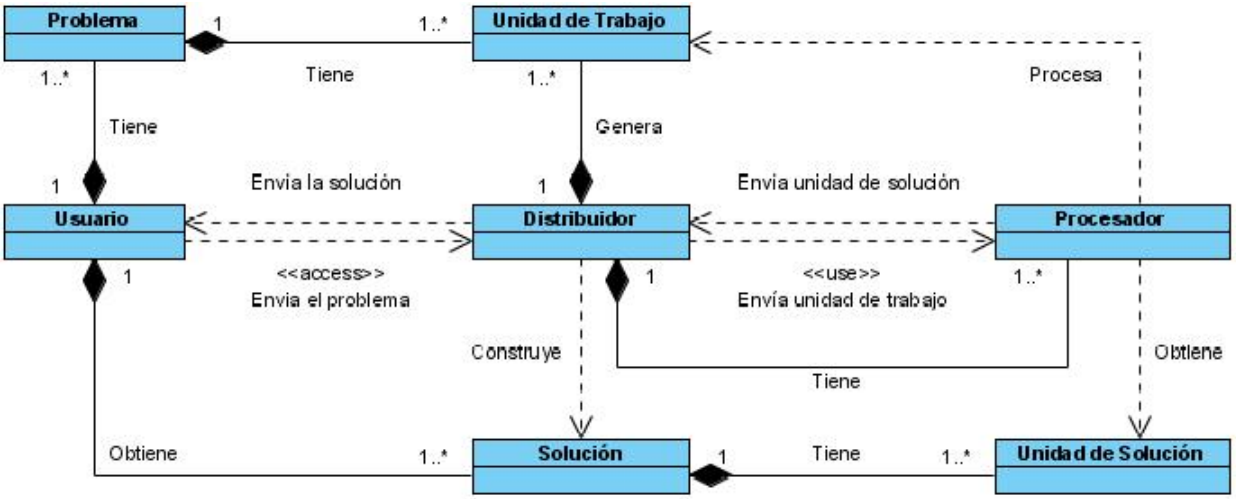

Figura 2.1: Diagrama de clases del modelo de dominio.

Para una mejor comprensión, a continuación se describen cada una de las clases que intervienen en el diagrama mostrado:

- Usuario: representa la entidad que accede a un problema y desea distribuirlo.
- Problema: representa la entidad que es accedida por un determinado usuario y que se va a resolver de manera distribuida.
- Distribuidor: representa la entidad encargada de aceptar un problema, dividirlo en pequeñas unidades de trabajo, repartir estas unidades por los procesadores, construir la solución final y devolver esta última al usuario.
- Unidad de Trabajo: representa una porción del problema, cuando éste es dividido en pequeñas partes.
- Procesador: representa la entidad encargada de procesar la unidad de trabajo asignada y obtener una unidad de solución correspondiente al procesamiento realizado, la cual es retornada al distribuidor para confeccionar la solución final del problema.
- Unidad de Solución: representa la entidad obtenida a partir del procesamiento de una unidad de trabajo. Es utilizada por el distribuidor para obtener la solución.
- Solución: entidad final que se le devuelve al usuario, obtenida a partir de las unidades de solución retornadas por cada procesador.

2.2. Especificación de los requisitos del sistema

Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, y se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Se identificaron los siguientes:

1. Autenticar usuario
2. Gestionar grupo
   2.1. Adicionar un grupo
   2.2. Eliminar grupo
3. Mostrar listado con los grupos existentes
4. Gestionar usuario
   4.1. Adicionar un usuario
   4.2. Eliminar usuario
   4.3. Cambiar la contraseña de un usuario
   4.4. Habilitar o no la cuenta de un usuario
5. Mostrar listado con los usuarios existentes
6. Gestionar rango IP
   6.1. Adicionar un rango IP
   6.2. Permitir o no un rango IP
   6.3. Eliminar rango IP
7. Mostrar listado con los rangos IP existentes
8. Gestionar cliente
   8.1. Permitir o no un cliente
   8.2. Eliminar cliente
9. Mostrar listado de los clientes

10. Gestionar problema
    10.1. Adicionar un problema
    10.2. Eliminar problema
11. Mostrar listado con los problemas accedidos por un usuario
12. Mostrar listado con los problemas administrados por un usuario
13. Ejecutar un problema
14. Detener ejecución
15. Mostrar listado con las ejecuciones existentes
16. Mostrar listado con las ejecuciones que pertenecen a un usuario
17. Monitorear ejecución
    17.1. Devolver el estado de una ejecución
    17.2. Devolver los errores de una ejecución
18. Descargar solución
19. Eliminar solución
20. Mostrar listado con las soluciones existentes
21. Mostrar listado con las soluciones que pertenecen a un usuario
22. Listar servidor de peticiones
23. Mostrar reporte gráfico
24. Mostrar Información del Servidor Central

Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, y que harán del mismo un sistema confiable y seguro.

Usabilidad

- Las personas que interactúan con la interfaz serán investigadores de los centros del polo científico del oeste de la Habana, y de la Universidad de la Habana.
- La interfaz será usada también por los productos desarrollados en el Polo de Bioinformática de la UCI.
- El sistema tendrá un ambiente sencillo y fácil de manejar para los usuarios, incluso aquellos que no han tenido mucha experiencia en el trabajo con computadoras o con sistemas informáticos.

Fiabilidad

- Las interfaces y menús serán creados según el privilegio del usuario que se autentique.
- Se deben garantizar comunicaciones seguras entre la aplicación web y el back-end de T-arenal, encriptando todo el tráfico de información con la utilización de algoritmos y protocolos.

Disponibilidad

- Se garantizará el funcionamiento de la aplicación durante las 24 horas del día y los siete días de la semana con el menor tiempo posible de recuperación de fallos.

Portabilidad

- El sistema será multiplataforma, razón por la cual se podrá utilizar en cualquier Sistema Operativo y arquitectura de hardware.

Software

- Un servidor Apache Tomcat para el despliegue de la solución y en el caso del usuario final, el navegador web Mozilla Firefox.

Hardware

- La máquina donde radicará el servidor de la plataforma debe tener como mínimo 1 GB de RAM y 3.0 GHz de velocidad del microprocesador, además debe contar con una capacidad suficiente para almacenar todos los problemas, soluciones y ejecuciones que existan en el sistema.
- La red de interconexión debe tener como mínimo una velocidad de 100 mbps.

2.3. Actores y casos de uso del sistema. Su relación

Actores del sistema

Los actores representan a terceros fuera del sistema que interactúan con él. En el sistema que se describe se identificaron los siguientes actores:

Actor	Descripción
Usuario	Representa al usuario que va hacer uso del sistema, teniendo la posibilidad de interactuar con las funcionalidades más básicas de este.
Usuario Avanzado	Representa al usuario que va hacer uso del back-end teniendo la posibilidad de interactuar con las funcionalidades de administración que le son permitidas.
Administrador de Problemas	Representa al usuario que va hacer uso del back-end teniendo la posibilidad de gestionar problemas.
Administrador	Representa al usuario que va hacer uso del back-end teniendo la posibilidad de realizar todas las funcionalidades que brinda.

Listado de casos de uso del sistema

La forma en que los actores usan el sistema es representada a través de los casos de uso. Estos últimos son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor. Los casos de uso identificados en el presente trabajo son enunciados a continuación:

Orden	Nombre	Prioridad	Breve descripción
1	Autenticar Usuario (ver Anexo A pág 58)	Crítico	El usuario inicia el caso de uso cuando decide interactuar con el back-end. Entonces introduce los datos necesarios y acepta o cancela la autenticación.
2	Gestionar Grupo	Secundario	El caso de uso inicia cuando el administrador decide adicionar, modificar, eliminar o visualizar los grupos existentes en el back-end. Una vez realizada alguna de estas acciones finaliza el caso de uso.

3	Gestionar Usuario	Secundario	El caso de uso inicia cuando un usuario interactúa con el front-end y decide adicionar, eliminar, cambiar contraseña, activar o desactivar una cuenta o visualizar los usuarios existentes. Una vez realizada alguna de estas acciones finaliza el caso de uso.
4	Gestionar Rango IP	Secundario	El caso de uso inicia cuando el administrador decide adicionar, permitir, eliminar o visualizar rangos IP. Una vez realizada alguna de estas acciones finaliza el caso de uso.
5	Gestionar Cliente	Secundario	El caso de uso comienza cuando el administrador visualiza los clientes existentes y posteriormente decide autorizarlos o eliminarlos.
6	Gestionar Problema (ver Anexo A pág 59)	Crítico	El caso de uso se inicia cuando el administrador decide adicionar, eliminar o visualizar los problemas existentes. Una vez realizada alguna de estas acciones finaliza el caso de uso.
7	Ejecutar Problema (ver Anexo A pág 62)	Crítico	El caso de uso comienza cuando el usuario visualiza los problemas a los cuales accede, y posteriormente decide ejecutarlos.
8	Detener Ejecución	Secundario	El caso de uso comienza cuando el usuario visualiza las ejecuciones a las cuales tiene acceso y posteriormente decide detenerlas.
9	Monitorear Ejecución	Secundario	El caso de uso comienza cuando el usuario visualiza las ejecuciones a las cuales tiene acceso y posteriormente decide mostrar el estado o los errores de las mismas.
10	Descargar Solución	Secundario	El caso de uso comienza cuando el usuario visualiza las soluciones a las cuales tiene acceso y posteriormente decide descargar alguna de ellas.
11	Eliminar Solución	Secundario	El caso de uso comienza cuando el usuario visualiza las soluciones a las cuales tiene acceso y posteriormente decide eliminarlas.

12	Listar Servidor de Peticiones	Secundario	El caso de uso inicia cuando el administrador decide visualizar los servidores de peticiones existentes en el back-end. Una vez realizada alguna de estas acciones finaliza el caso de uso.
13	Mostrar Reporte Gráfico	Secundario	El caso de uso comienza cuando el usuario selecciona el tipo de reporte gráfico que desea visualizar. El back-end busca los datos necesarios y construye la gráfica a mostrar. Finaliza el caso de uso.
14	Mostrar Información del Servidor Central	Secundario	El caso de uso inicia cuando el administrador decide visualizar la información sobre el servidor central del back-end. Después de realizada esta acción finaliza el caso de uso.

Diagrama de casos de uso del sistema

Los diagramas de casos de uso del sistema representan gráficamente a los procesos y su interacción con los actores. El diagrama de casos de uso del sistema del presente trabajo se muestra en la figura 2.2.

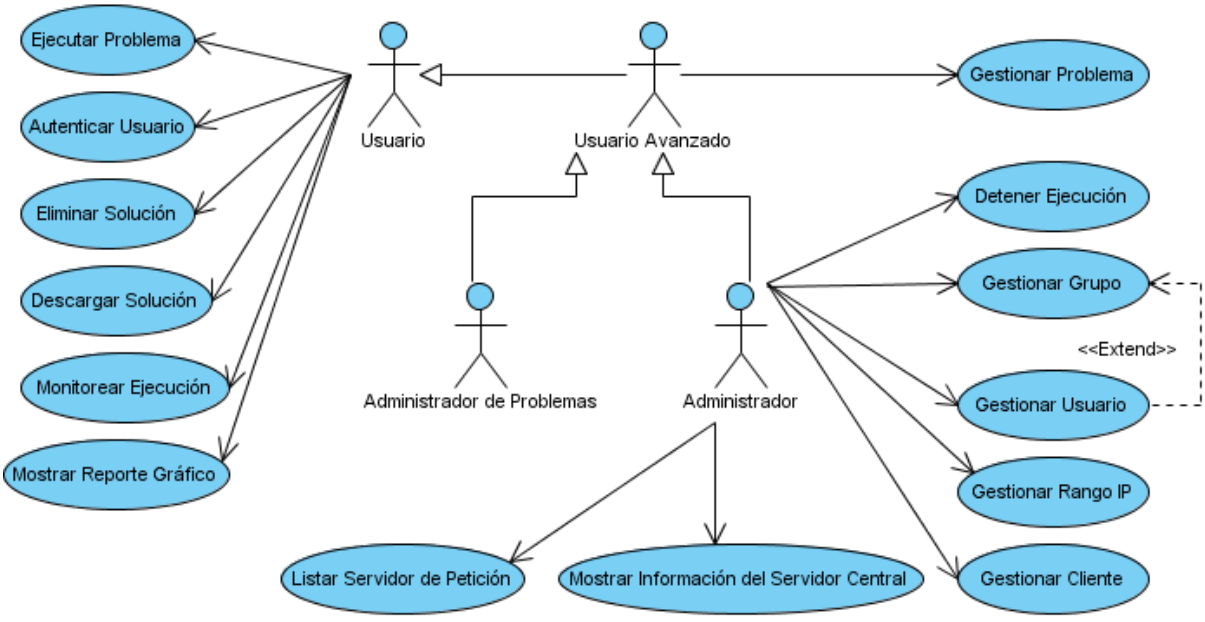

Figura 2.2: Diagrama de casos de uso del sistema.

Vista de casos de uso arquitectónicamente significativos

En síntesis podemos decir que los casos de uso valorados como arquitectónicamente significativos son aquellos que nos ayudan a mitigar los riesgos más importantes. Deben ser los más importantes para los usuarios del sistema y que ayuden a cubrir las funcionalidades significativas. A continuación se muestra la vista de esos casos de uso:

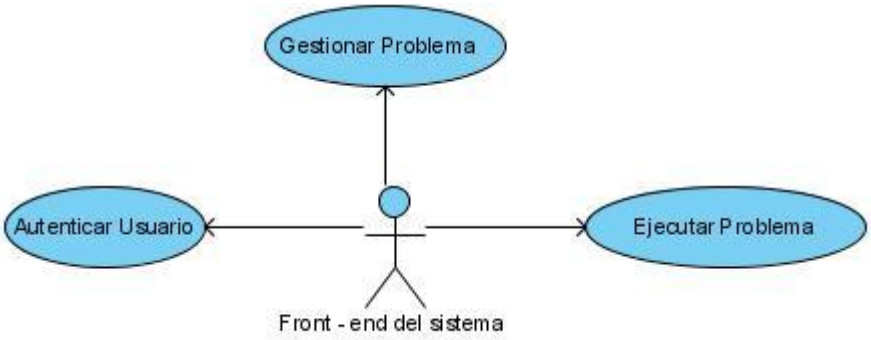

Figura 2.3: Vista de casos de uso arquitectónicamente significativos.

2.4. Conclusiones

En el presente capítulo se realizó un modelo de dominio, ya que existe un conocimiento previo del problema a resolver lo cual brinda cierta simplicidad a la hora de determinar los conceptos que intervienen en él. Se identificaron los requisitos funcionales y no funcionales necesarios para el desarrollo de la aplicación web, manteniendo una alta concordancia con la aplicación de escritorio; los actores y casos de uso del sistema que sirvieron de base para realizar el diseño del producto que se presenta en este trabajo. Se mostraron las relaciones existentes entre actores y casos de uso y los casos de uso considerados como críticos para la arquitectura; también se encuentran presentes los diagramas de casos de uso del sistema y la vista de casos de uso más significativos del sistema.

Capítulo 3

Diseño del Sistema

En este capítulo se describe la representación arquitectónica del sistema, haciendo énfasis en el estilo y el patrón de arquitectura utilizado, así como en los patrones de diseño más importantes. Se muestran los diagramas de clases de los paquetes relevantes de cada capa y los diagramas de secuencias necesarios para comprender el funcionamiento de la propuesta de solución que se presenta.

3.1. Representación arquitectónica

La misión principal de la arquitectura del sistema es mostrar una panorámica general de los entes y subsistemas que lo integran, explicando cada uno de estos en diferentes vistas. El modelo de representación arquitectónica está basado en el modelo de las 4 + 1 vistas. La vista de procesos no se describirá porque no se considera relevante la información que ésta brinda. La vista de datos está descrita en [36]. La vista de casos de uso se describió en el capítulo anterior y la vista de implementación se encuentra descrita en el siguiente capítulo.

Patrón de arquitectura

El patrón de arquitectura seleccionado es el Modelo Vista Controlador (MVC) [37] que pertenece a la clasificación de los estilos de Llamada y Retorno [38]. Es un patrón de diseño de arquitectura de software, usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas, donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de una mejor manera. El mismo facilita la programación en diferentes capas de manera paralela e independiente.

Aplicación del patrón MVC

Para un mejor entendimiento de la selección de este patrón se puede identificar el contexto en el cual se ubica el problema existente y la solución propuesta por el mismo:

- Contexto: aplicación con interfaz hombre-máquina.
- Problema: es muy frecuente la solicitud de informaciones a los servidores del sistema, lo cual generará constantes cambios en la interfaz. Los mismos deben ser fáciles y efectuados en tiempo de ejecución, y no deben tener consecuencias para el núcleo del código de la aplicación.
- Solución: separación del sistema en tres partes: Modelo, Vista y Controlador.

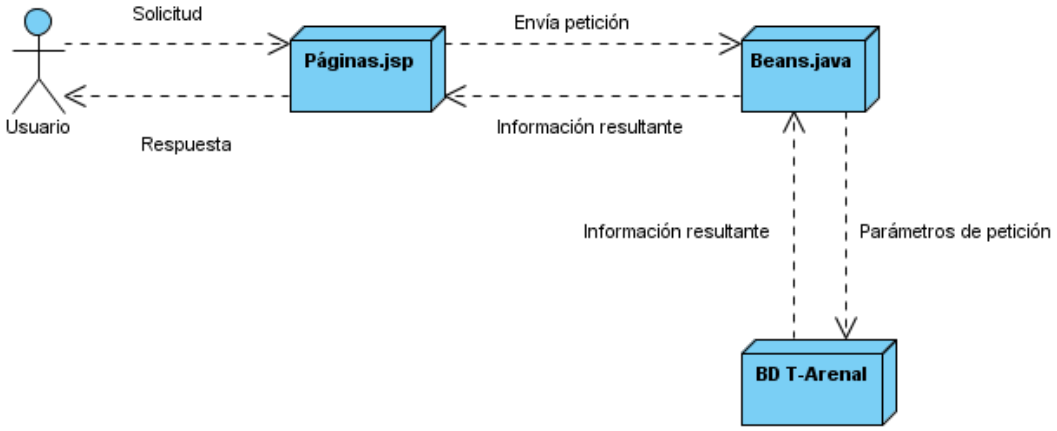

Figura 3.1: Aplicación del patrón *Modelo Vista Controlador*.

- **Modelo:** es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de éstos y permite derivar nuevos datos. Esta función la realiza la base de datos de T-arenal por lo cual no se enfatizará en ella.
- **Vista:** despliega la información contenida en el modelo. Presenta el modelo en un formato adecuado para interactuar con el sistema y para mejor entendimiento del usuario. En este caso son las páginas jsp las encargadas de esta función.
- **Controlador:** este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Está representado como Beans.java, librería que permite ejecutar las funcionalidades solicitadas por el usuario que interactúa con la base de datos e interactuar con el subsistema T-arenal.

Ciclo de Vida del patrón MVC

El primer paso de este ciclo de vida comienza cuando el usuario hace una solicitud a las *vistas* (paginas.jsp) con información sobre lo que el usuario desea realizar. Seguido las *vistas* le envían la petición al *controlador* (beans.java) y este decide a quien delegar las tareas, es aquí donde el *modelo* (Base de datos de T-arenal) empieza su trabajo. En esta etapa el *modelo* se encarga de realizar operaciones sobre la información que maneja para cumplir con lo solicitado. Una vez terminada su labor le regresa al *controlador* la información resultante de sus operaciones, el cual a su vez se dirige a la *vista* correspondiente ya que son las encargadas de transformar los datos en información visualmente entendible para el usuario. Finalmente la representación gráfica es transmitida al usuario.

3.2. Patrones de diseño utilizados

Un patrón es una solución de un problema en determinado contexto, un contexto es una situación donde el patrón es aplicable y que generalmente es recurrente. Los patrones de diseño más importantes aplicados en el desarrollo del sistema son el View Helper [39] y el Composite View [40]. A continuación se presenta una breve descripción de cada uno de ellos.

View Helper

- **Problema:** los cambios en la capa de presentación ocurren muy frecuentemente y son difíciles de desarrollar y mantener cuando la lógica de acceso a los datos de negocio y la lógica del formateo de la presentación se mezclan. Esto hace al sistema menos flexible, menos reutilizable y generalmente menos adaptable a los cambios.

- **Solución:** las vistas (addGroup.jsp) delegan sus responsabilidades de procesamiento en su clase de ayuda implementada como JavaBeans (GroupManagerBean.java). Las clases de ayuda son las encargadas de interactuar con el sistema T-arenal y de retornar los resultados obtenidos a la vista que los solicitó.

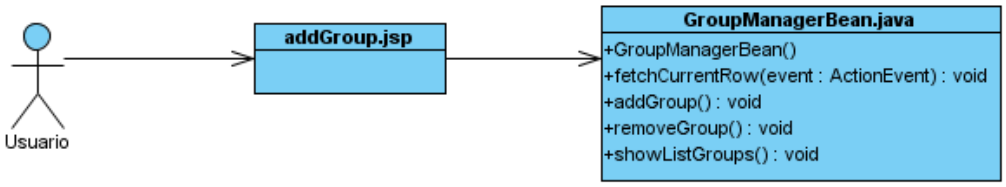

Figura 3.2: Aplicación del patrón *View Helper*.

Composite View

- **Problema:** se necesita embeber código dentro de algunas páginas para un mejor manejo de los cambios y el código sea más fácil de mantener.
- **Solución:** utilizar vistas compuestas que se componen de varias subvistas atómicas. Cada componente se incluye dinámicamente dentro del total y la distribución de la página se maneja independientemente del contenido.

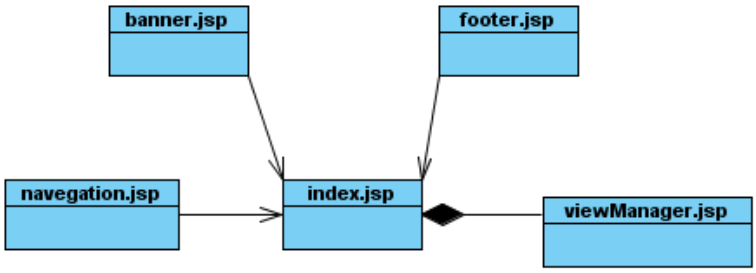

Figura 3.3: Aplicación del patrón *Composition View*.

3.3. Vista Lógica

En esta sección se describen las partes arquitectónicamente significativas del modelo de diseño como son la descomposición en capas y paquetes.

La descripción de la vista lógica se realizará por capas para un mejor entendimiento de las características de cada una de las partes. Se mostrarán por cada capa los diagramas de paquetes de aquellos que se consideran arquitectónicamente significativos, teniendo en cuenta su relación y composición. Se ilustrarán además los diagramas de clases del diseño de los casos de uso que sean necesarios, acompañados con una breve explicación de algunas de las clases que los componen.

Capa Modelo

La capa Modelo, como se mencionó anteriormente, está compuesta por la base de datos de T-arenal, la cual es transparente a la programación de las funcionalidades, siendo la librería la encargada de trabajar directamente con ella. Por estas razones no se enfatizará en esta capa.

Capa Vista

En esta capa las clases están organizadas en paquetes, como se muestra en la figura (figura 3.4). Aquí se definen todas las interfaces de usuario además de los complementos que éstas necesitan, como son las imágenes, los estilos y las librerías.

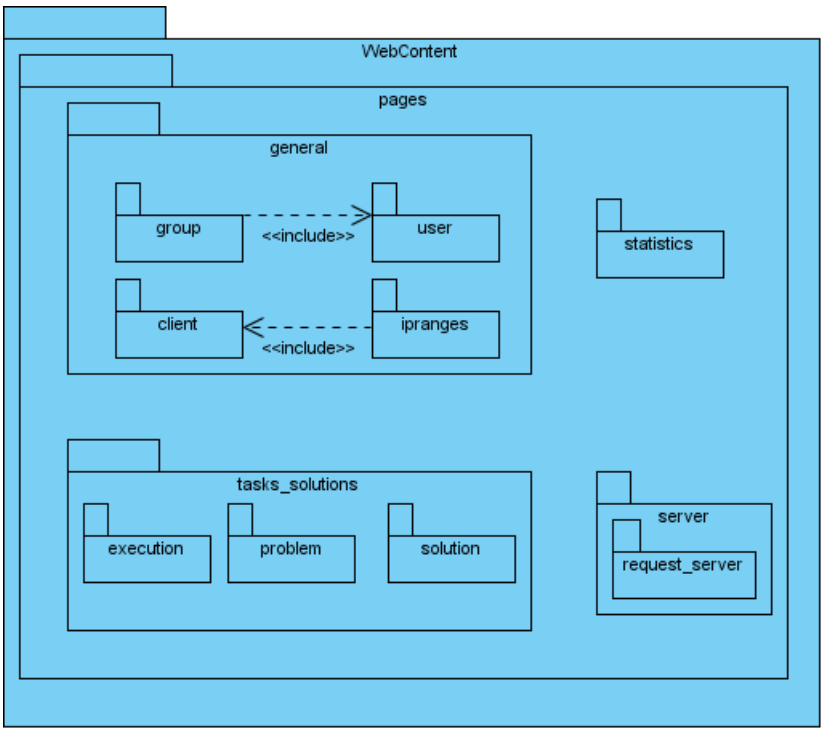

Figura 3.4: Diagrama de paquetes de la capa *Vista*.

Descripción de los paquetes de la capa Vista:

- **pages:** define todas las vistas relacionadas con las funcionalidades organizadas en subpaquetes.
- **general:** define las vistas relacionadas con el caso de uso gestionar usuario, gestionar grupo, gestionar rango Ip e información de los clientes el cual consta de cuatro subpaquetes.

- **server:** define las vistas relacionadas con los casos de uso Gestionar servidor de peticiones y Mostrar información del servidor central.
- **statistics:** define la vista relacionados con el requisito funcional de Mostrar reportes gráficos.
- **tasks_solutions:** define las vistas relacionadas con las ejecuciones, problemas y soluciones, consta de tres subpaquetes.
- **group:** paquete que se encuentra dentro del paquete general define las vistas relacionadas con los requisitos funcionales de adicionar y listar grupo.
- **user:** paquete que se encuentra dentro del paquete general define las vistas relacionadas con los requisitos funcionales de adicionar usuario y listar usuario.
- **request_server:** paquete que se encuentra dentro del paquete de server define las vistas relacionadas con los requisitos funcionales Mostrar listado con los servidores de peticiones existentes.
- **execution:** paquete que se encuentra dentro del paquete tasks_solutions define las vistas relacionadas con los requisitos funcionales de gestionar ejecuciones, monitorear ejecuciones y gestionar el funcionamiento de las ejecuciones.
- **problem:** paquete que se encuentra dentro del paquete tasks_solutions define las vistas relacionadas con los requisitos funcionales de gestionar problema, ejecutar problema y gestionar acceso a problemas.
- **solution:** paquete que se encuentra dentro del paquete tasks_solutions define las vistas relacionadas con los requisitos funcionales de descargar soluciones y eliminar soluciones.

Capa Controlador

En la capa Controlador se encuentran todas las clases o beans organizados como se muestra en la figura 3.5

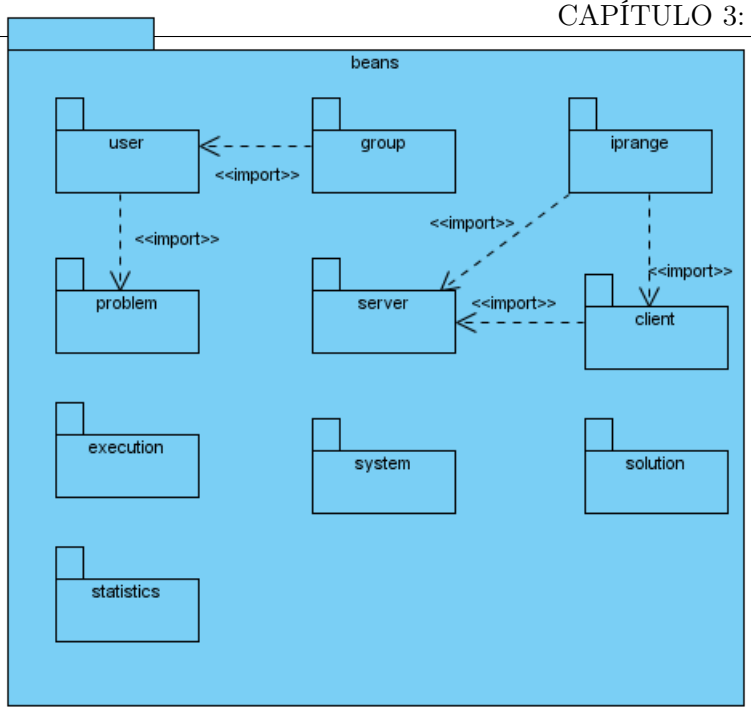

Figura 3.5: Diagrama de paquetes de la capa *Controladora.*

Descripción de los paquetes de la capa Controlador

- **user:** define las clases de ayuda relacionadas con las funcionalidades de los usuarios como adicionar usuario, listar usuario.
- **group:** define las clases de ayuda o beans relacionadas con las funcionalidades de los grupos.
- **iprange:** define las clases de ayuda o beans relacionada con las funcionalidades de los rangos ip como adicionar un rango ip y mostrar rangos ip.
- **problem:** define las clases de ayuda relacionadas con las funcionalidades de los problemas.
- **server:** define las clases de ayuda relacionadas con las funcionalidades de los servidores así como mostrar su información.
- **client:** define las clases de ayuda relacionadas con las funcionalidades de los clientes así como mostrar los clientes, permitir o no a un cliente.
- **execution:** define la clase de ayuda relacionada con las ejecuciones.
- **system:** define las clases de ayuda relacionadas con las funcionalidades de cambiar contraseña de un usuario.

- **solution:** define las clases de ayuda relacionadas con las funcionalidades de las soluciones así como descargar una solución.
- **statistics:** define las clases de ayuda relacionadas con las funcionalidades de los reportes gráficos.

Diagrama de clases del diseño del caso Autenticar Usuario

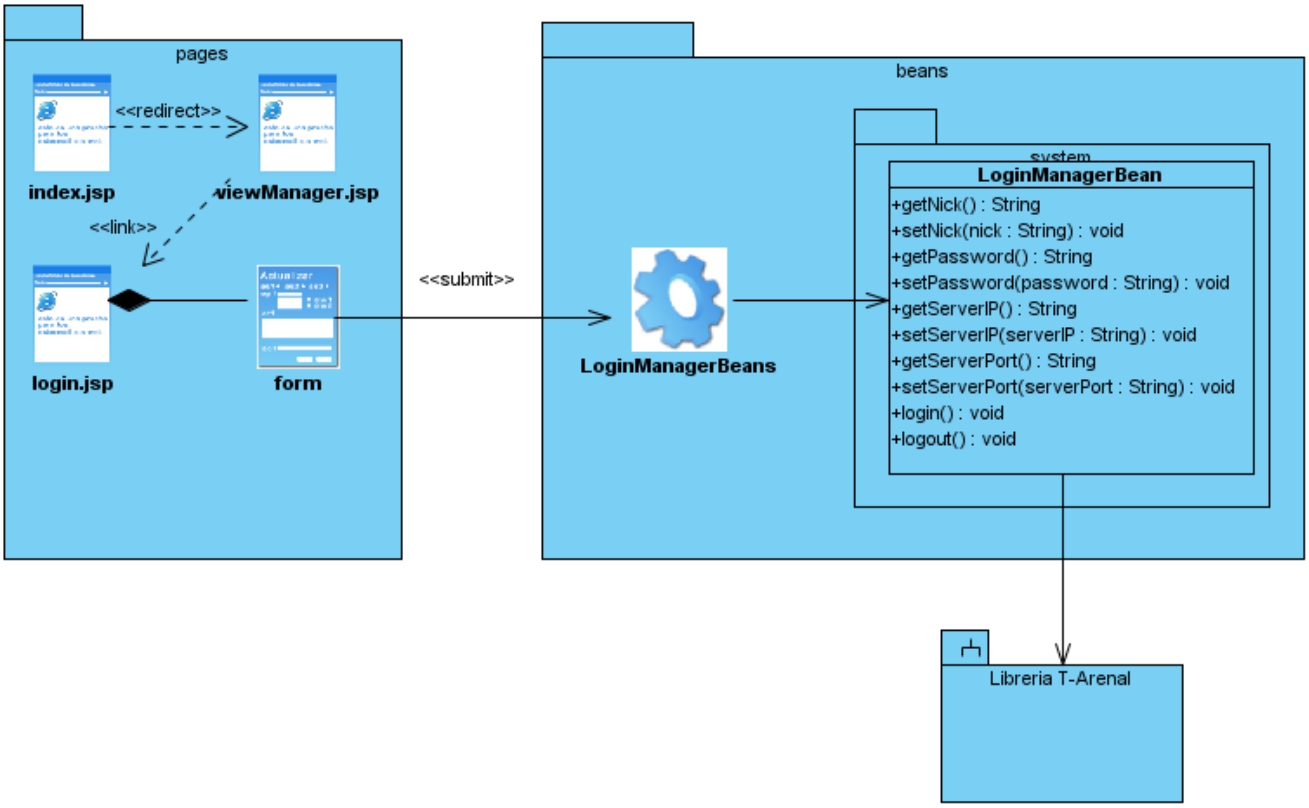

Figura 3.6: Diagrama de clases de diseño del CU *Autenticar Usuario*.

- **index.jsp:** es la página principal, tiene la responsabilidad de incluir las subvistas y conformar el diseño completo.
- **viewManager.jsp:** es la vista que se encarga de administrar las vistas e incluirlas según las peticiones del usuario.
- **login.jsp:** es la vista encargada de la autenticación del usuario permitiéndole acceder a la sección correspondiente según su privilegio.

- **loginManagerBeans.java:** en la clase controladora que entre otras funciones ejecuta el método login se comunica con la librería y autentica al usuario.

Diagrama de clases del diseño del Caso de Uso Gestionar Problema

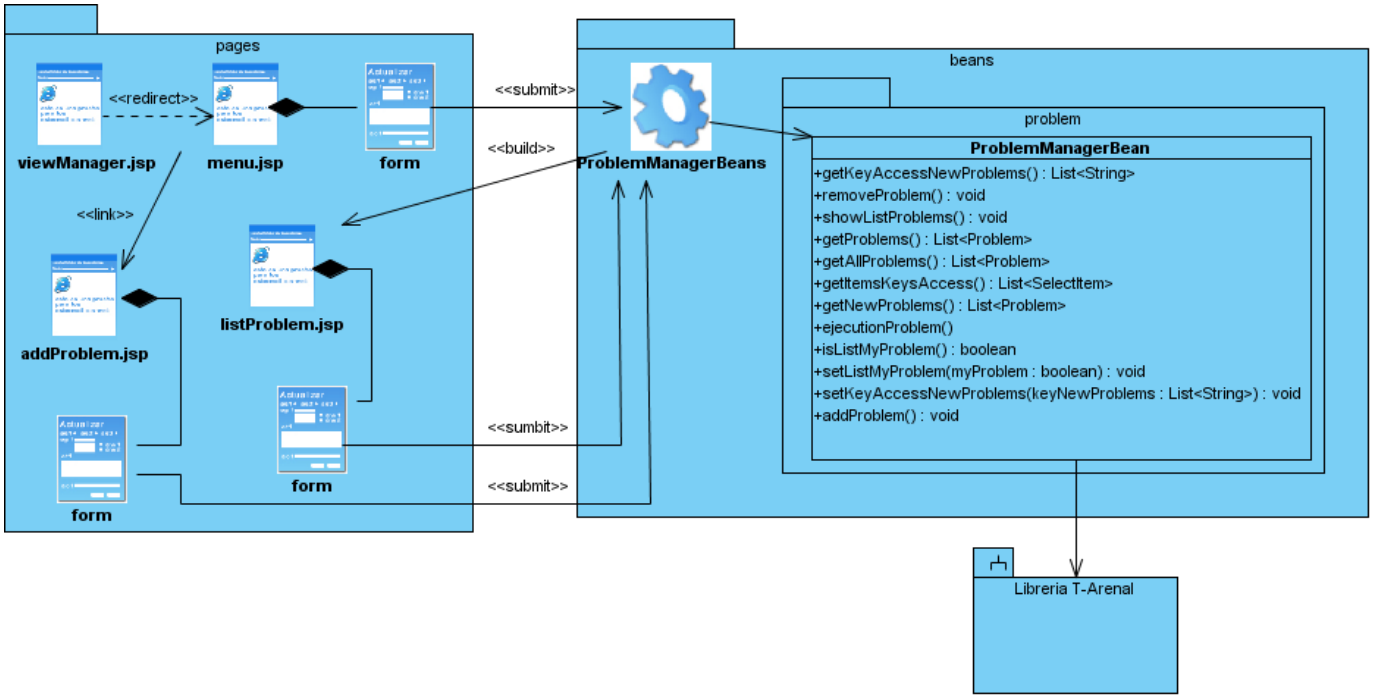

Figura 3.7: Diagrama de clases de diseño del CU *Gestionar Problema*.

- **menu.jsp:** interfaz que le permite al usuario acceder a las funcionalidades que éste puede realizar e interactuar con el sistema.
- **addProblem.jsp:** interfaz que tiene la responsabilidad de mostrar al usuario todos los campos que necesita llenar para adicionar un problema al sistema.
- **listProblem.jsp:** interfaz que le permite al usuario visualizar los problemas existentes y también eliminarlos.
- **ProblemManagerBeans.java:** clase con la responsabilidad de ejecutar todas las funcionalidades relacionadas con los problemas entre las cuales están listar y eliminar los problemas.

Diagrama de clases del diseño del Caso de Uso Ejecutar Problema

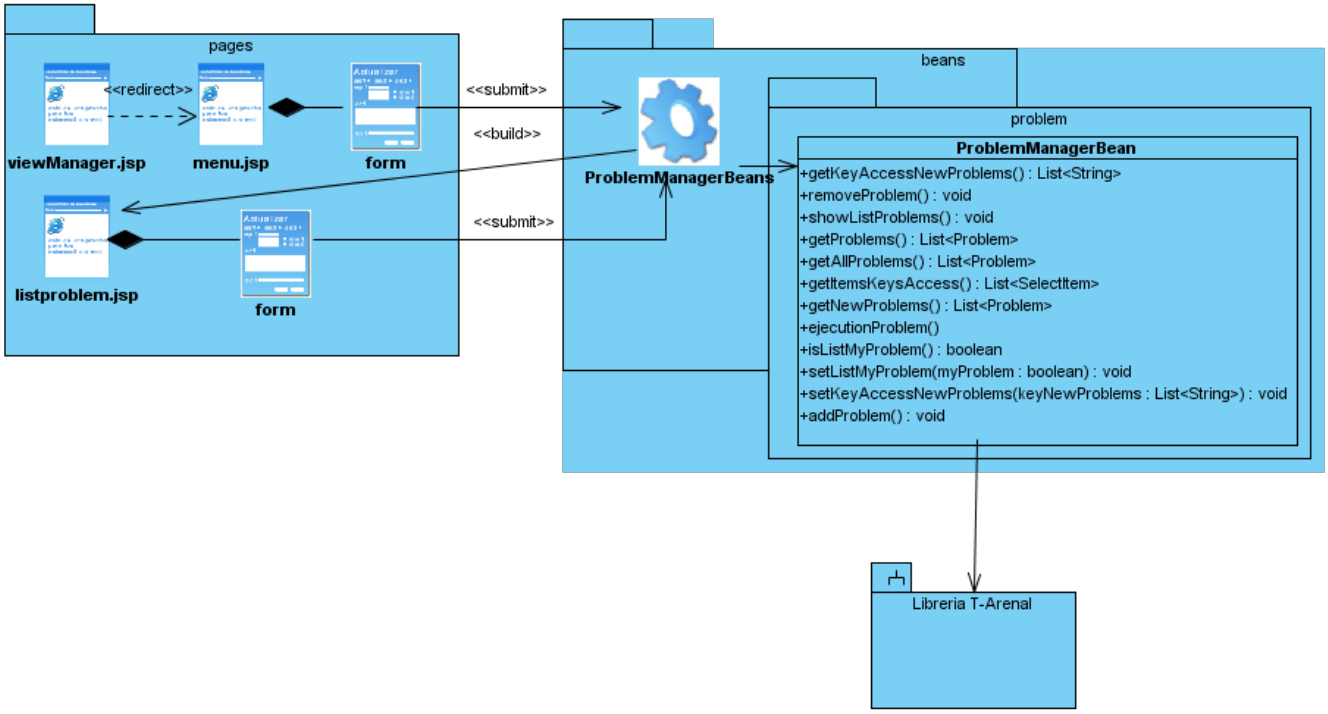

Figura 3.8: Diagrama de clases de diseño del CU *Ejecutar Problema*.

- **selectFile.jsp:** es la vista que le permite al usuario seleccionar los archivos que posteriormente serán utilizados por la ejecución.

Diagrama de secuencias

Un diagrama de secuencia muestra las interacciones entre objetos, ordenadas temporalmente. Ilustra los objetos que se encuentran en un escenario y la secuencia de mensajes intercambiados entre ellos para llevar a cabo la funcionalidad descrita.

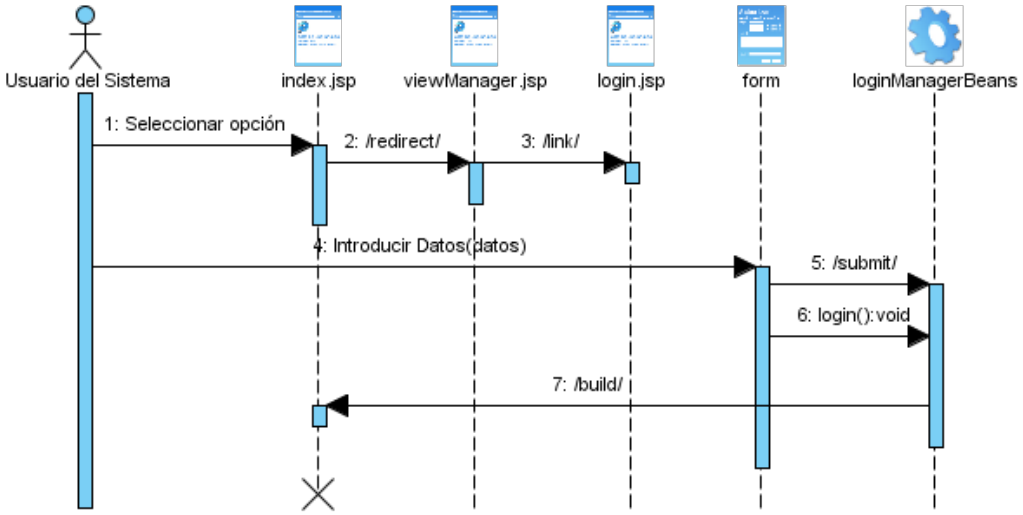

Figura 3.9: Diagrama de secuencia del CU *Autenticar Usuario*.

Para el caso de uso *Autenticar Usuario* el actor selecciona del *index.jsp* la opción que le permitirá autenticarse en el sistema. Una vez mostrada la interfaz correspondiente, el usuario introduce los datos necesarios y hace clic en el botón *Entrar*. Posteriormente los datos son enviados hacia la página controladora, se invoca el método *login* y finalmente se construye la vista acorde a los privilegios del usuario autenticado. De esta forma concluye la secuencia de mensajes entre objetos. De forma análoga este mismo funcionamiento es realizado en los siguientes diagramas de secuencia por lo cual no serán detallados.

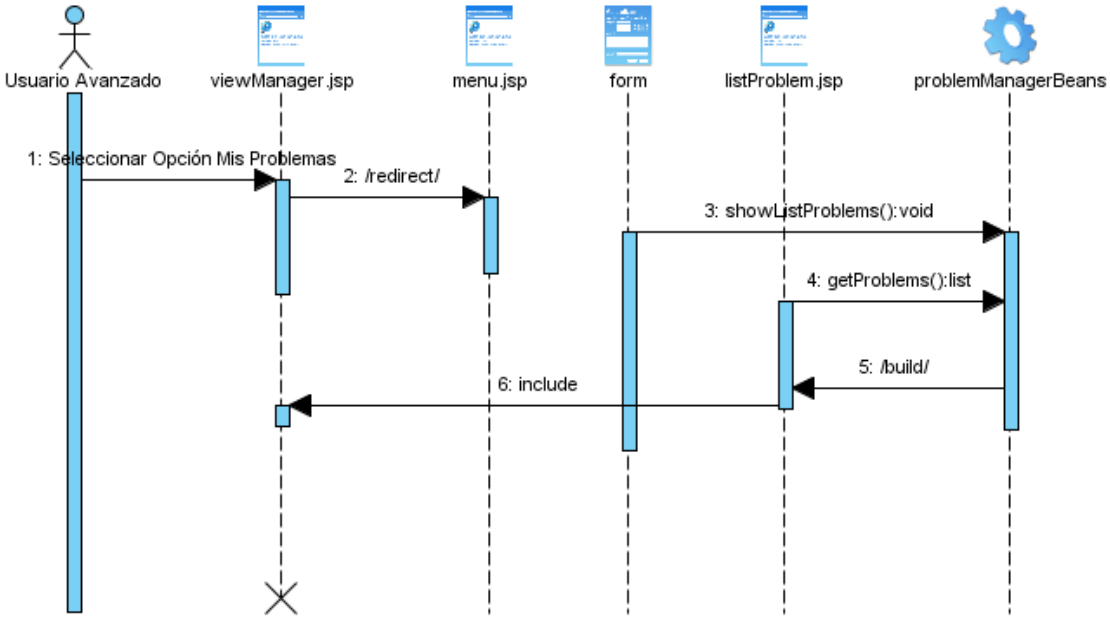

Figura 3.10: Diagrama de secuencia del CU *Gestionar Problema*. Escenario *Listar Problema*.

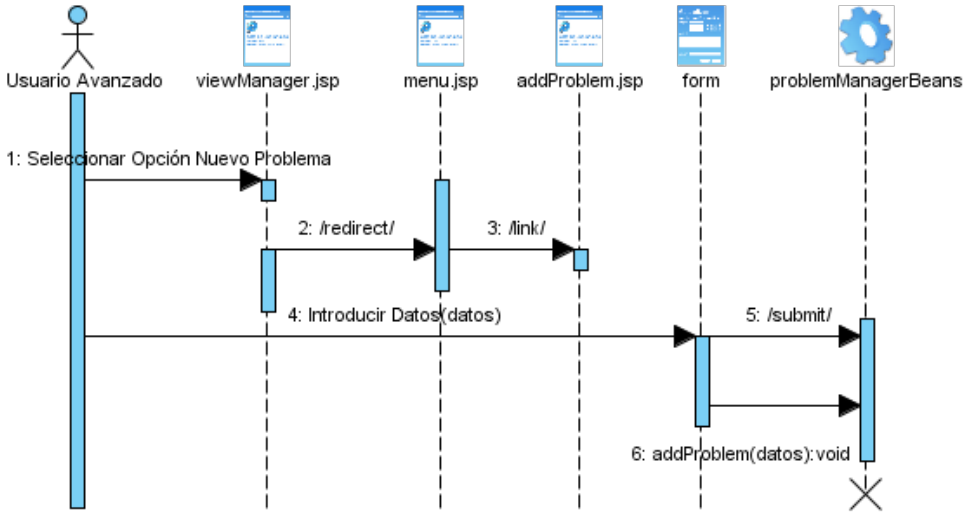

Figura 3.11: Diagrama de secuencia del CU *Gestionar Problema*. Escenario *Adicionar Problema*.

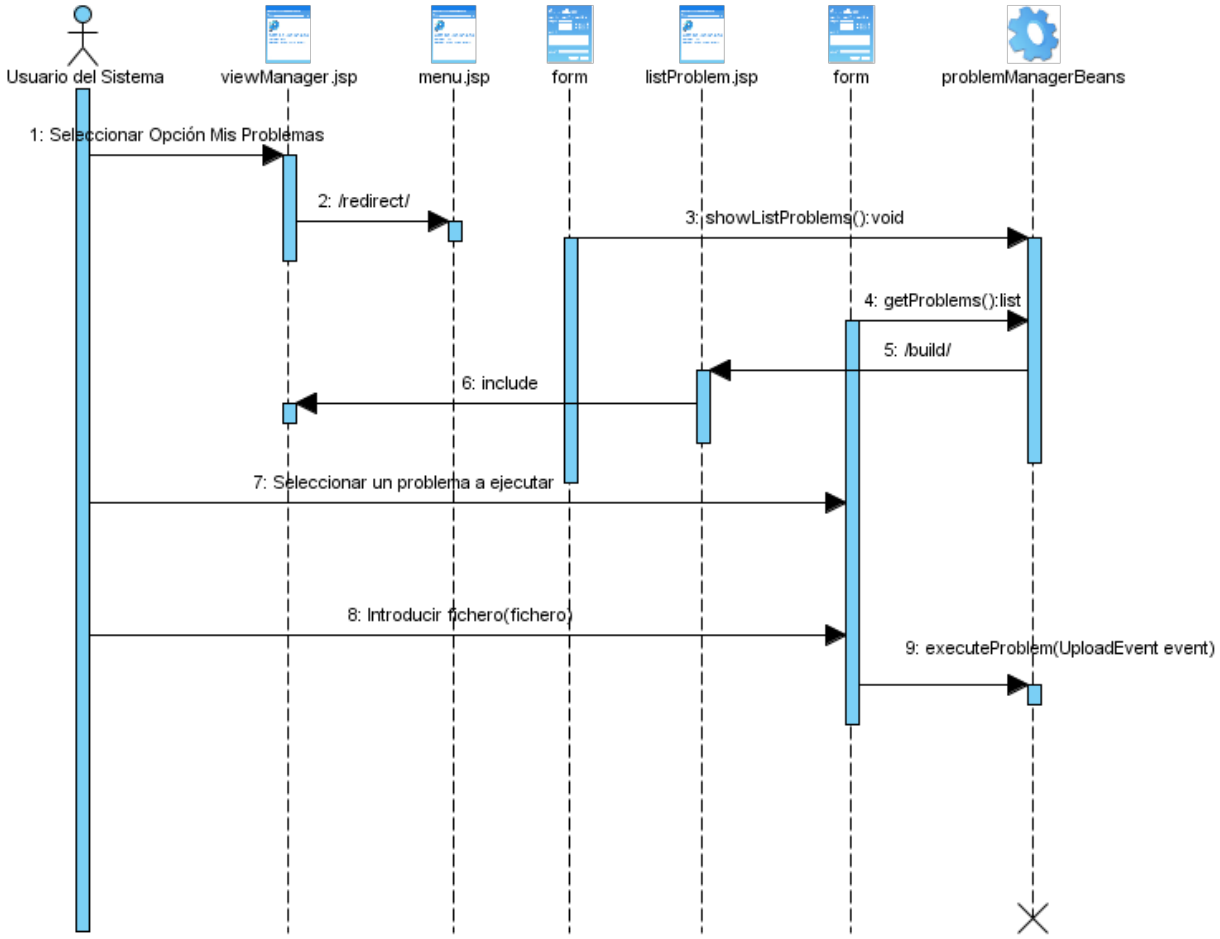

Figura 3.12: Diagrama de secuencia del CU *Ejecutar Problema*.

Vista de despliegue

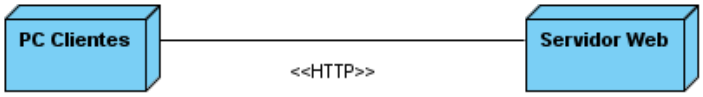

Figura 3.13: Vista de despliegue del sistema.

Descripción de los nodos

- **PC Clientes:** es el nodo donde los usuarios acceden e interactúan con la Aplicación Web.
- **Servidor Web:** es el nodo donde radica el servidor Web para publicar la aplicación.

Descripción de la comunicación

- **HTTP:** el Protocolo de Transferencia de Hipertexto (Hypertext transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.

3.4. Conclusiones

Como resultado de este capítulo se obtuvo la representación de la arquitectura del sistema a través de la representación Modelo - Vista - Controlador. Se determinaron los patrones de diseños más importantes aplicados en el desarrollo de la interfaz. Se elaboró el diseño del sistema desde la estructura de las vistas hasta los diagramas de clases y de secuencia por cada caso de uso, mostrando solamente los más significativos. También se muestra la distribución física del sistema.

Capítulo 4

Implementación y Prueba del Sistema

En este capítulo se describe la implementación del sistema en términos de componentes. Además, se explica y muestra los casos de prueba y los resultados obtenidos.

4.1. Diagrama de Componentes

El diagrama de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes. Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, binarios, archivos, bibliotecas cargadas dinámicamente o ejecutables [36].

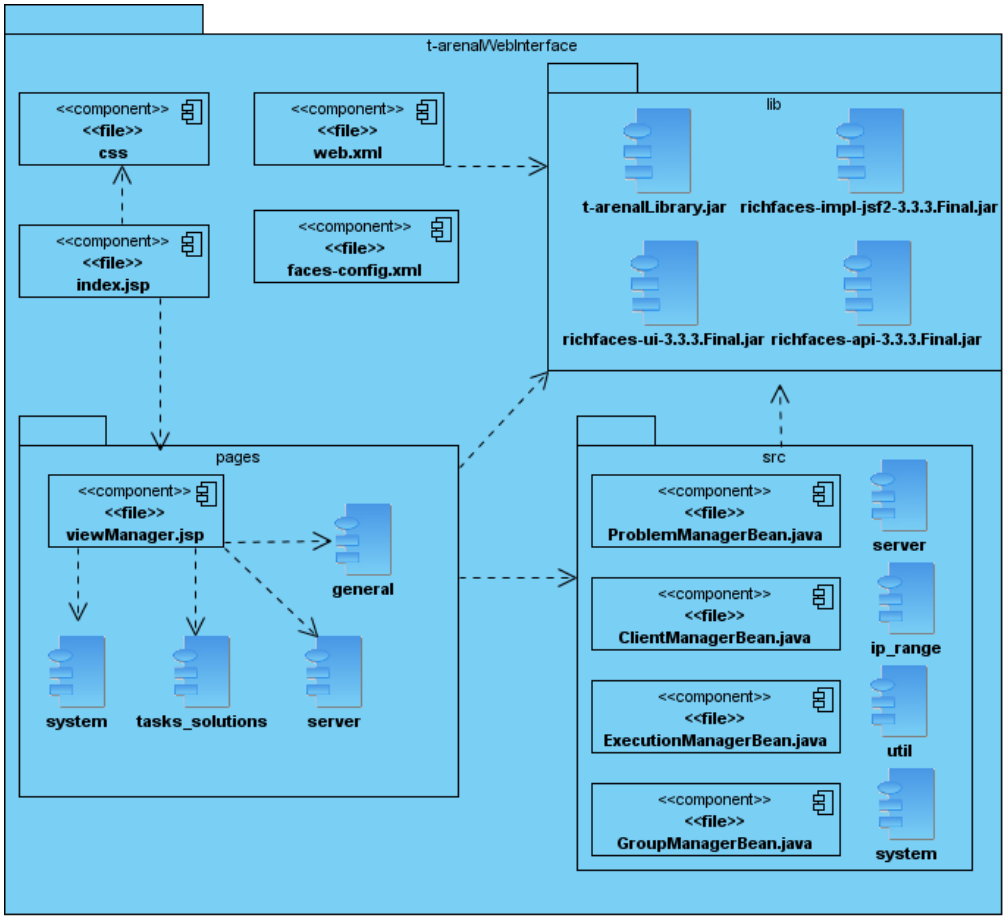

Figura 4.1: Diagrama de *Componentes.*

La figura 4.1 muestra los componentes relevantes del sistema y su interacción. Cada uno de ellos está organizado y embebido en paquetes, en dependencia de la función que realiza dentro del sistema. En la carpeta *lib* se encuentra una representación de los componentes más importantes como son las tres librerías de RichFaces [23], *richfaces-ui-3.3.3.Final.jar*, *richfaces-api-3.3.3.Final.jar*, *richfaces-impl-jsf2-3.3.3.Final.jar* y la librería de T-arenal llamada *t-arenalLibrary.jar.*

El paquete *src* es el que consta de los componentes y contenedores de componentes que se relacionan con el paquete *lib* para la utilización de la librería de T-arenal. El mismo tiene el objetivo de brindar y obtener información desde la Plataforma de Tareas Distribuidas. Además, interactúa con el paquete *pages* debido a que este es el que tiene todas las interfaces para mostrar las informaciones obtenidas. Los componentes localizados en *pages* son los ficheros *.jsp* que se relacionan con las librerías del framework para la apariencia de las interfaces de usuarios.

El componente *index.jsp* es la página principal de la aplicación. Esta hace uso directo de *viewManger.jsp*

que tiene la función de administrar las vistas en dependencia de la petición de los usuarios; y el fichero *css* es el que tiene todos los estilos aplicados en la aplicación. También se encuentra el *faces-config.xml* y el *web.xml* que son ficheros de configuración necesarios para el funcionamiento de la aplicación.

4.2. Descripción del código

En esta sección se tratará un ejemplo donde se muestra la vinculación entre algunos componentes mostrados en el diagrama. El código seleccionado pertenece al caso de uso crítico *Autenticar Usuario*.

Esta codificación pertenece a la clase *login.jsp* que se encuentra dentro del paquete *pages*. Esta clase es incluida en el *index.jsp* mediante la clase *viewManager.jsp* por la aplicación de patrón *Composite View*. Es una clase que contiene la declaración de las librerías de etiquetas de *jsf* y los elementos y componentes de *RichFaces* necesarios que le permiten al usuario introducir sus datos.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>

<%@ taglib uri="http://richfaces.org/a4j" prefix="a4j" %>
<%@ taglib uri="http://richfaces.org/rich" prefix="rich" %>

<rich:modalPanel id="conexion" resizeable="false" width="306" height="165" style="text-align: center;"
keepVisualState="false" moveable="false">
    <f:facet name="header">
        <h:outputText value="Conexión a T-arenal" style=" font-size:12px;"></h:outputText>
    </f:facet>
    <f:facet name="controls">
        <h:panelGroup>
            <h:graphicImage value="/images/images/modal/close.png" styleClass="hidelink" id="hidelink31"/>
            <rich:componentControl for="conexion" attachTo="hidelink31" operation="hide" event="onclick"/>
        </h:panelGroup>
    </f:facet>

    <h:form>
```

```
<h:panelGrid columns="2">
    <rich:spacer width="10" height="0" />
    <h:panelGrid columns="4" columnClasses="cols" style="text-align: right; font-size:12px;">

        <h:outputText value="Usuario" style="font-size:12px;"/>
        <rich:spacer width="10"/>
        <h:inputText id="usuario" value="#{loginManager.nick}" required="true" style="width:140px;">
            <f:validateLength minimum="3" maximum="12"/>
            <rich:ajaxValidator event="onkeyup" />
        </h:inputText>
        <rich:message for="usuario" tooltip="true" showDetail="false">
            <f:facet name="errorMarker">
                <h:graphicImage value="images/error.png" style="border:0" title="Valor incorrecto"/>
            </f:facet>
        </rich:message>

        <h:outputText value="Contraseña" style="font-size:12px;"/>
        <rich:spacer width="10"/>
        <h:inputSecret id="pass" value="#{loginManager.password}" required="true" style="width:140px;">
            <f:validateLength minimum="5" maximum="15" />
            <rich:ajaxValidator event="onkeyup" />
        </h:inputSecret>
        <rich:message for="pass" tooltip="true" showDetail="false">
            <f:facet name="errorMarker">
                <h:graphicImage value="images/error.png" style="border:0" title="Valor Incorrecto" />
            </f:facet>
        </rich:message>

        <h:outputText value="IP Servidor" style="font-size:12px;"/>
        <rich:spacer width="10"/>
        <h:inputText id="server" value="#{loginManager.serverIP}" required="true" style="width:140px;">
            <f:validateLength minimum="7" maximum="15" />
                <rich:ajaxValidator event="onkeyup" />
        </h:inputText>
        <rich:message for="server" tooltip="true" showDetail="false">
```

```
            <f:facet name="errorMarker">
                <h:graphicImage value="images/error.png" style="border:0" title="Valor Incorrecto" />
            </f:facet>
        </rich:message>

        <h:outputText value="Puerto Servidor" style="font-size:12px;"/>
        <rich:spacer width="10"/>
        <h:inputText id="port" value="#{loginManager.serverPort}" required="true" style="width:140px;">
            <f:validateLength minimum="4" maximum="4" />
            <rich:ajaxValidator event="onkeyup" />
        </h:inputText>
        <rich:message for="port" tooltip="true" showDetail="false">
            <f:facet name="errorMarker">
                <h:graphicImage value="images/error.png" style="border:0" title="Valor Incorrecto" />
            </f:facet>
        </rich:message>
    </h:panelGrid>
</h:panelGrid>

<rich:spacer height="3" />
<rich:separator height="2" lineType="solid"/>
<rich:spacer height="3" />

<h:panelGrid columns="2">
    <rich:spacer width="187" height="0" />
    <h:commandButton id="login" action="#{loginManager.login}" value="Entrar" styleClass="button"
        style="background-color:#{a4jSkin.headerBackgroundColor};width:70px;font-size:12px;"/>
    </h:panelGrid>
</h:form>
</rich:modalPanel>
```

La siguiente figura muestra la interfaz que se obtiene como resultado del código anterior. Esta interfaz se relaciona con el *bean LoginManagerBean.java*, que se encuentra en el contenedor de componente *system*. Este bean contiene el método *login* que es invocado cuando el usuario presiona el botón *Entrar* (ver línea 35).

Figura 4.2: Interfaz para autenticarse en el sistema.

La función *login* utiliza la librería *t-arenalLibrary.jar* para el proceso de autenticación, y redirecciona hacia la página *index.jsp* para obtener la vista acorde a los privilegios del usuario autenticado.

```
public void login()
{
    UserAuthentication userA = new UserAuthentication( nick, password );
    HostCommunication host = new HostCommunication( serverIP, Integer.parseInt( serverPort ) );

    try
    {
        saveValueInSession( Constant.SYSTEM_CONNECTION, new SystemConnection( Locale.US, userA, host ));

        setNick( "" );
        setPassword( "" );
        setServerIP( "" );
        setServerPort( "" );

        saveValueAndDispatch( Constant.PAGE_TEXT );
    }
    catch ( Throwable e )
    {
        e.printStackTrace();
    }
}
```

Para que todo lo anterior funcione correctamente y las librerías RichFaces sean reconocidas por la

aplicación hay que configurar el fichero *faces-config.xml*, adicionándole los *beans* utilizados y así estos puedan ser llamados desde las vistas y las reglas de navegación cuando es necesario. Para el caso que se ha estado explicando la porción de código para este archivo sería la siguiente.

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd"
  version="1.2">

  <managed-bean>
    <managed-bean-name>skinApp</managed-bean-name>
    <managed-bean-class>util.SkinApp</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
      <property-name>skin</property-name>
      <property-class>java.lang.String</property-class>
      <value>classic</value>
    </managed-property>
  </managed-bean>

  <managed-bean>
    <managed-bean-name>loginManager</managed-bean-name>
    <managed-bean-class>beans.system.LoginManagerBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
  .....
</faces-config>
```

Después de configurado el *faces-config.xml* hay que modificar el *web.xml*. Este fichero contiene la descripción de la aplicación Web: permite declarar servlets, asignarles parámetros de inicio, declarar alias y filtros.

```
<?xmlversion="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
```

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
version="2.5">

  <display-name>T-arenalWebInterface</display-name>
  <welcome-file-list>
    <welcome-file>faces/index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
  <filter>
    <display-name>RichFaces Filter</display-name>
    <filter-name>richfaces</filter-name>
    <filter-class>org.ajax4jsf.Filter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>richfaces</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
  </filter-mapping>
</web-app>
```

4.3. Pruebas al sistema

El flujo de trabajo de pruebas le presta servicios a los demás flujos. Su principal objetivo es evaluar o valorar la calidad del producto a través de la búsqueda y documentación de errores, validando el cumplimiento de los requerimientos, el desempeño y dando una indicación de calidad. La prueba es un proceso de ejecución de un programa con la intención de descubrir errores. Una prueba tiene éxito si descubre un error no detectado hasta entonces. Las pruebas de Caja Negra también conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba, son las que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se accede al código) [10].

Para realizar las pruebas del presente trabajo se utilizó la prueba de Particiones de Equivalencia que es una técnica de prueba de Caja Negra. Este método divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia.

Esta técnica fue aplicada en tres métodos del sistema, por cada uno se muestra como fueron particionadas las clases de equivalencia.

Casos de pruebas

Los escenarios principales de los casos de usos críticos fueron probados para detectar no conformidades. En las siguientes subsecciones se representan estos casos de prueba separados por escenario.

Ejecutar Problema

Condición de Entrada	Clases Válidas	Clases no Válidas
Archivos de la Tarea.	Todos los archivos suban sin problema.	Ninguna Orden Válida.

Autenticar Usuario

Condición de Entrada	Clases Válidas	Clases no Válidas
Nombre. Sólo contiene letras y números, debe tener entre 3 y 50 caracteres.	administrator	a) ro b) r c) Una clave que exceda los 50 caracteres permitidos.
Contraseña. No debe ser vacía y su longitud mínima es 3.	root	a) ad b)a c)Cadena vacía.
Servidor. La dirección IP está constituida por números y el caracter punto. Los números deben ser mayores o iguales a 0 y menores o iguales a 255.	10.128.60.60	a) 10.10.1 b) *.0.0.. c) 0.0.128.11.14
Puerto. Solamente se admiten números de 4 dígitos.	5900	a) 5.045 b) 111025 c) *87425

Gestionar Problemas

Condición de Entrada	Clases Válidas	Clases no Válidas
Clave de acceso. La clave de acceso está constituida por letras y números debe tener entre 3 y 50 caracteres.	root	a) ro b) r c) Una clave que exceda los 50 caracteres permitidos.
Prioridad. Los valores son Baja, Media y Alta.	Baja, Media o Alta.	Ninguna Orden Válida.
Descripción. Es un breve comentario sobre el algoritmo a subir.	Comentario	Ninguna Orden Válida.

Archivo *.zip* que contiene el problema a distribuir.	El archivo suba correctamente.	Fallo de la subida del Algoritmo.

4.4. Conclusiones

En este capítulo se obtuvo como resultado el diagrama de despliegue de los componentes del sistema el cual muestra las organizaciones y dependencias lógicas entre componentes de software, se hizo una breve explicación de los algoritmos más importantes que se implementaron además de algunos códigos de ejemplo y se mostraron las pruebas de caja negra con la utilización de la técnica de Particiones de Equivalencia para obtener un rápido resultado con las pruebas realizadas a los casos de uso críticos del sistema.

Conclusiones

1. Se realizó un estudio sobre las características de las aplicaciones de interfaces web de usuario, los estándares y las especificaciones existentes, con el objetivo de seleccionar las técnicas y herramientas necesarias para el desarrollo del trabajo.

2. A partir de las funcionalidades identificadas y debido a la necesidad de refinamiento se rediseñó el front-end de la Plataforma de Tareas Distribuidas, compuesto por una aplicación web que facilita la interacción con el servidor de la plataforma.

3. Se implementó la Interfaz Web para interactuar con el Back-end de la Plataforma de Tareas Distribuidas.

4. Se realizó la evaluación de las principales funcionalidades del sistema haciendo uso de las pruebas de Caja Negra.

Recomendaciones

1. Implementar el mecanismo de *Internacionalización* para la selección del idioma de trabajo.
2. Implementar el mecanismo de notificación, al usuario final, de los errores ocurridos en el back-end de T-arenal.
3. Implementar el mecanismo de actualización automática para el estado de los servidores y las ejecuciones.

Referencias bibliográficas

[1] Peña J. Los ordenadores del Futuro: Ordenadores Cuánticos y Moleculares;. Available from: http://www.redcientifica.com/doc/doc200212170001.html [cited 03 de diciembre del 2010].

[2] Shneiderman B. Designing the User Interface: Strategies for Effective Human-Computer Interaction 3rd. Addison-Wesley Longman Publishing Co.; 1997.

[3] Aguilera L Mendoza. Sistema de Cómputo Distribuido aplicado a la Bioinformática. Universidad de las Ciencias Informáticas; 2008.

[4] ; [cited 9 de diciembre de 2010]. Available from: http://www.gridsphere.org/.

[5] Zukowski J. Deploying Software with JNLP and Java Web Start. Sun Developer Network (SDN); 2002.

[6] Ministerio del Poder Popular para Ciencia, Tecnología e Industria Intermedias; [cited 03 de diciembre de 2010]. Available from: http://www.rena.edu.ve/cuartaEtapa/Informatica/Tema2.

[7] Myers BA. User interface software technology. Journal ACM Computing Surveys. 1996 Mar;28(1).

[8] Hammer. INTERFACES DE USUARIO. Journal ACM Computing Surveys. 1983 Mar;28(1).

[9] Larson JA. Interactive Software: Tools for Building Interactive User Interfaces;.

[10] Rosabal MF Alarcón, Tellez R Ibarra. T-arenal v2.0: Desarrollo del front-end. Universidad de las Ciencias Informáticas; 2009.

[11] Moreno L. Componentes de una Interfaz web; 22 de septiembre 2005. Available from: http://www.desarrolloweb.com/articulos/2171.php.

[12] Powell TA. Diseño de sitios web. Manual de referencia. Osborne MacGraw-Hill; 2000.

[13] Barranco MJ García. Tema 3.Interfaces de Usuario;. Available from: http://wwwdi.ujaen.es/~barranco/publico/ofimatica/tema3.pdf.

[14] Hassan Y, Martín FJ Fernández, Iazza G. Diseño Web Centrado en el Usuario: Usabilidad y Arquitectura de la Información;. Available from: http://www.hipertext.net/web/pag206.htm.

[15] Moreno L. Componentes de una Interfaz web. EL Cuerpo de una página; 17 de octubre 2005. Available from: http://www.desarrolloweb.com/articulos/2205.php.

[16] Marrero C Expósito. Interfaz gráfica de usuario: Aproximación semiótica y cognitiva. Universidad de La Laguna; 2004.

[17] Rosenfeld L, Morville P. Information Architecture for the World Wide Web. O'Reilly Media; 2002.

[18] Canós JH, Letelier P, Penadés MC. Métodologías Ágiles en el Desarrollo de Software. VIII Jornadas de Ingeniería del Software y Bases de Datos;.

[19] Rumbaugh J, Jacobson I, Booch G. El Proceso Unificado de Desarrollo de Software. Addison Wesley; 1999.

[20] Extreme Programming: A gentle introduction;. Available from: http://www.extremeprogramming.org/ [updated 2006 Feb 17; cited 2010 Dic 3].

[21] Eclipse Process Framework Project;. Available from: http://www.eclipse.org/epf/ [cited 2009 Feb 20].

[22] Ledo R Ramírez, Ortega A Palacios. Módulo Admisión del Sistema de Información Hospitalaria alas HIS. Universidad de las Ciencias Informáticas; 2009.

[23] Hat R. RichFaces Developer Guide. 2007;Available from: http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/html_single/.

[24] Angel M Alvarez. La tecnología Java para la creación de páginas web con programación en el servidor. 2002 Jul;.

[25] Pérez JE. Introducción a CSS; 2009.

[26] The Apache Software Foundation; [cited 04 de diciembre de 2010]. Available from: http://tomcat.apache.org/.

[27] Machado M Díaz, Coca YL Ribas. SACCEM: Módulo de gestión de la información del Departamento de Supervisión del CCEEM versión 1.0; 2009.

[28] Unified Modeling Language;. Available from: http://www.uml.org/ [cited 2010 Dic 3].

[29] Larman C. UML y Patrones. Introducción al Análisis y diseño orientado a objetos; 1999.

[30] Schmuller J. Aprendiendo UML en 24 Horas. SAMS Publishing; 2001.

[31] Gracia J. UML: Diagramas UML. ¿Qué es UML?; 2005. Available from: http://www.ingenierosoftware.com/analisisydiseno/uml.php.

[32] Magic Draw;. Available from: http://www.magicdraw.com/.

[33] Rational Rose;. Available from: http://www.ibm.com/software/rational.

[34] Umbrello;. Available from: http://uml.sourceforge.net/.

[35] ArgoUML;. Available from: http://argouml.tigris.org/.

[36] García CR Jacas, Miralles DM Taset. T-arenal v2.0: Desarrollo del backend. Universidad de las Ciencias Informáticas; 2009.

[37] Pavón MJ. Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC). Programación Orientada a Objetos Facultad de Informática. 2008;.

[38] Reynoso C, Kiccillof N. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft; 2004.

[39] Lou R Torrijos. Catálogo de Patrones de Diseño J2EE. I.- Capa de Presentación. Programación en Castellano;.

[40] Lou R Torrijos. Catálogo de Patrones de Diseño J2EE. I.- Capa de Presentación. Programación en Castellano;.

Anexo A

Descripción de casos de usos

Autenticar Usuario

Nombre del CU	**Autenticar Usuario**
Actor	Usuario del sistema.
Propósito	Autenticar un usuario en la aplicación web.
Resumen	El usuario inicia el caso de uso cuando decide interactuar con la aplicación web. Entonces introduce los datos necesarios, se verifica la validez de los mismos y finalmente lo acepta o lo rechaza.
Referencias	RF 1
Curso Normal de Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario desea autenticarse en el sistema.	1.1. La aplicación web muestra la interfaz para autenticarse, donde se debe especificar el usuario, la contraseña, el ip del servidor y el puerto servidor una vez que se acciona el vínculo *Autenticarse*.
2. El usuario introduce el usuario, la contraseña, la dirección IP del servidor y el puerto de comunicación y acciona el botón *Entrar*.	
3. El usuario acepta autenticarse en el sistema.	3.1. La aplicación web se comunica con el back-end e invoca la funcionalidad para autenticarse en el mismo.

	3.2. La aplicación web crea y muestra la interfaz con las funcionalidades que podrá realizar el usuario autenticado, según el privilegio que tenga. Finaliza el caso de uso.
Cursos Alternativos **CA1**	
3. El usuario decide no autenticarse, y cierra la interfaz mostrada. Finaliza el caso de uso.	
Prioridad	**Crítico**

Gestionar Problemas

Nombre del CU	**Gestionar Problemas**
Actor	Usuario Avanzado (administrador del sistema, administrador de problemas).
Propósito	Permitir gestionar problemas en el sistema.
Resumen	El caso de uso se inicia cuando el administrador decide adicionar, eliminar o visualizar problemas existentes. Una vez realizada alguna de estas acciones finaliza el caso de uso.
Referencias	RF 10.1, RF 10.2, RF 11, RF 12
Precondiciones	El usuario ha sido identificado con el privilegio de administrador o administrador de problemas.
Curso Normal de Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El actor desea gestionar problemas en el back-end.	1.1. La aplicación web brinda dos opciones: ▪ Adicionar problemas. ▪ Mostrar los problemas que son administrados por el actor.

2. El actor decide: ▪ Adicionar un problema. Ver sección Adicionar Problema. ▪ Mostrar los problemas que administra. Ver sección Mostrar Problemas.	
Sección Mostrar Problemas	
Curso Normal de los Eventos	
1. El actor desea ver los problemas que administra en el sistema.	1.1. La aplicación web se comunica con el back-end y obtiene los problemas que administra el actor.
	1.2. La aplicación web muestra un listado con los problemas obtenidos.
2. El actor decide: ▪ Adicionar un problema. Ver sección Adicionar Problema. ▪ Eliminar un problema. Ver sección Eliminar Problema.	
Cursos Alternativos **CA1**	
Sección Adicionar Problema	
Curso Normal de los Eventos	
1. El actor desea adicionar un problema.	1.1. La aplicación web muestra una interfaz con el botón *Adiccionar* donde se podrá subir el nuevo problema.

2. El actor acciona el botón *Adiccionar* el cual le permitirá buscar un archivo *.zip donde se encontrarán los datos del problema. Finalmente el actor acepta crear el nuevo problema.	2.1. La aplicación web se comunica con el back-end e invoca la funcionalidad para adicionar un problema.
	2.2. La aplicación web actualiza el listado donde se muestran todos los problemas. Finaliza el caso de uso.
Cursos Alternativos **CA1**	
2. El actor decide cancelar la adición del nuevo problema, y cierra la interfaz mostrada por el sistema. Finaliza el caso de uso.	
Sección Eliminar Problema	
Curso Normal de Eventos	
1. El actor selecciona del listado el problema que desea eliminar.	
2. El actor elimina el problema seleccionado.	2.1. La aplicación web muestra una interfaz para confirmar la eliminación de la selección realizada.
3. El actor confirma eliminar el problema seleccionado.	3.1. La aplicación web se comunica con el back-end e invoca la funcionalidad para eliminar el problema seleccionado.
	3.2. La aplicación web actualiza el listado donde se visualizan los problemas. Finaliza el caso de uso.
Cursos Alternativos **CA1**	
3. El actor no confirma eliminar el problema seleccionado. Finaliza el caso de uso.	
Prioridad	**Crítico**

Ejecutar Problema

Nombre del CU	**Ejecutar Problema**
Actor	Usuario del sistema.
Propósito	Permitir ejecutar problemas en la aplicación web.
Resumen	El caso de uso comienza cuando el usuario visualiza los problemas a los cuales accede, y posteriormente decide ejecutarlos.
Referencias	RF 13

Curso Normal de Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario desea ver los problemas a los cuales tiene acceso.	1.1. La aplicación web se comunica con el back-end y obtiene los problemas a los cuales el usuario tiene acceso.
	1.2. La aplicación web muestra un listado con los problemas obtenidos.
2. El usuario selecciona un problema y lo ejecuta.	2.1. La aplicación web muestra una interfaz con el botón *Adiccionar* mediante el cual se podrá localizar el archivo a transferir.
3. El usuario acepta crear e iniciar la ejecución del problema.	
	3.1. Si existen ficheros la aplicación web se comunica con el back-end y realiza la transferencia de los mismos.
	3.2. La aplicación web actualiza la interfaz mostrada donde se visualiza el progreso que tiene la operación de transferencia. Al concluir la transferencia, la aplicación web se comunica con el back-end e invoca la funcionalidad para iniciar la ejecución. Finaliza el caso de uso.

4. El usuario cancela la transferencia de los archivos hacia el back-end.	4.1. La aplicación web interrumpe la transferencia de los archivos y no inicia la ejecución creada. Finaliza el caso de uso.
Cursos Alternativos **CA1**	
3. El usuario decide no ejecutar el problema, y cancela, o no confirma, la creación e inicio de la ejecución. Finaliza el caso de uso.	
CA2	
	4. El usuario no cancela la transferencia de los archivos, permitiendo que esta se complete (dirigirse a la actividad número 4.1 del flujo normal de eventos).
Prioridad	**Crítico**

Anexo B

Imágenes de la Interfaz Web Desarrollada

Figura B.1: Interfaz de presenación y autenticación.

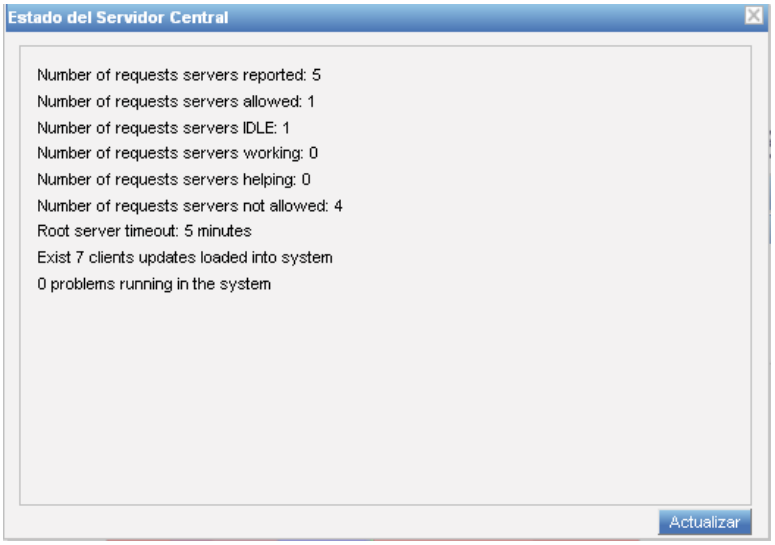

Figura B.2: Interfaz de estado del Servidor Central.

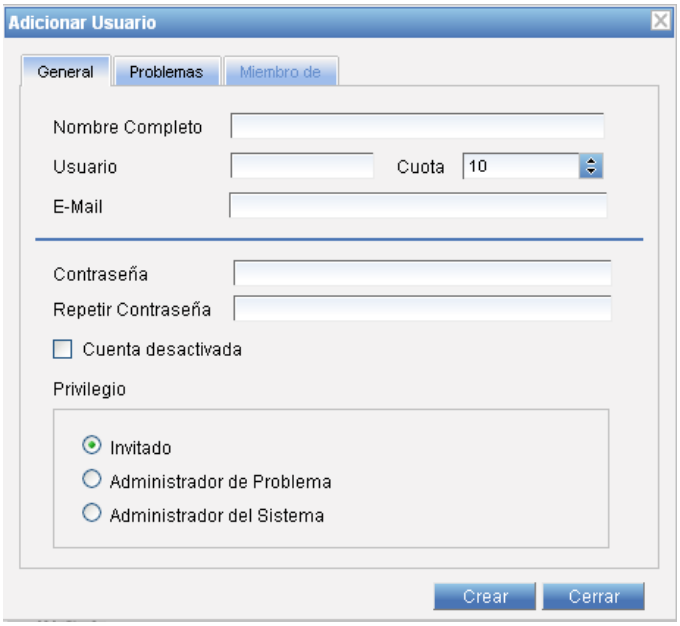

Figura B.3: Interfaz para adicionar un usuario.

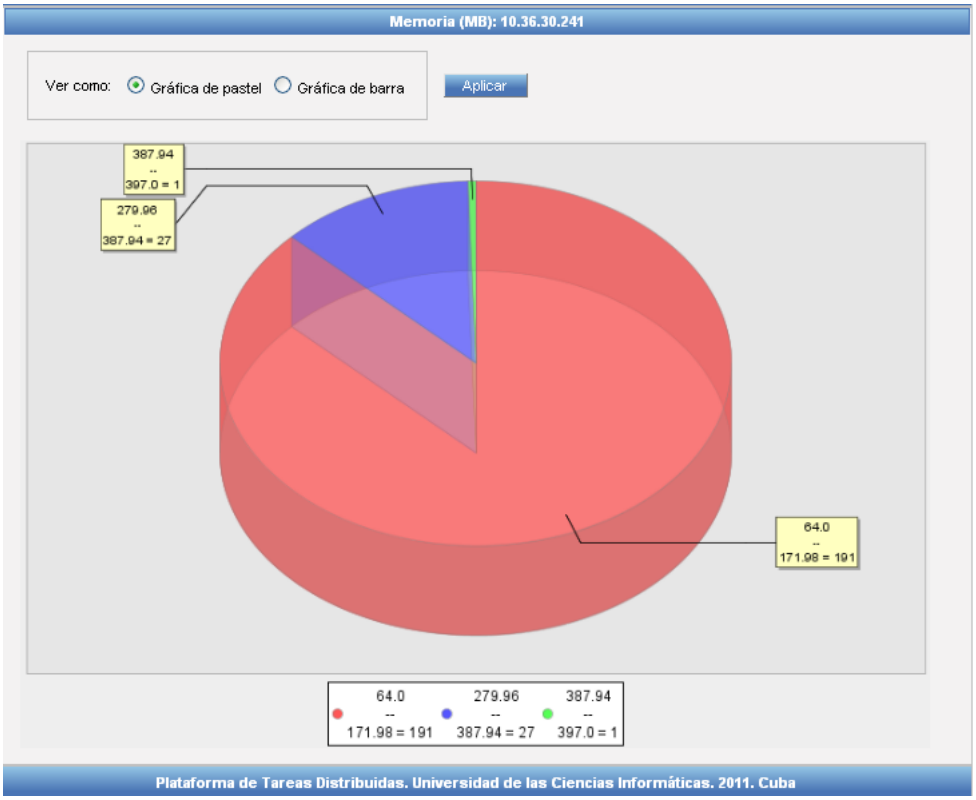

Figura B.4: Gráfica de pastel. La memoria RAM empelada por los clientes del servidor 10.36.30.241.

Figura B.5: Interfaz que muestra los rangos de IP del servidor 10.36.30.244.