UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

Título: "T-arenal v2.0: Desarrollo del front-end"

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Maikel Fernando Rosabal Alarcón

Roberto Tellez Ibarra

Tutor: MSc. Lic. Longendri Aguilera Mendoza

JUNIO-2009

La respuesta de que el proceso científico puede tener algunas veces repercusiones perjudiciales, no debe implicar el abandono del avance científico, sino su sustitución por un avance aún mayor, aplicado con prudencia e inteligencia.

Isaac Asimov

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 _____.

Maikel Fernando Rosabal Alarcón

Firma del Autor

MSc. Lic. Longendri Aguilera Mendoza

Firma del Tutor

Roberto Tellez Ibarra

Firma del Autor

DATOS DE CONTACTO

Autores:
Maikel Fernando Rosabal Alarcón.
Universidad de las Ciencias Informáticas.
e-mail: mfrosabal@estudiantes.uci.cu

Roberto Tellez Ibarra.
Universidad de las Ciencias Informáticas.
e-mail: rtibarra@estudiantes.uci.cu

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

AGRADECIMIENTOS

Agradecer a mi madre por el sacrificio que hizo para que yo terminara mis estudios, por estar conmigo en las buenas y en las malas, por enseñarme lo bueno y lo malo de la vida, por brindarme lo mucho con lo poco que tenía, por la enseñanza moral e intelectual que me dio, por ser la luz que me ha hecho seguir en los peores momentos, por darme en su cuerpo albergue de vida y por ser de mi vida la mejor parte.

También agradecer a los que estuvieron, los que están y los que continuarán ayudándome en mi vida.

Quisiera agradecer a todos los maestros que dejaron algo suyo en mí, y que se esforzaron para que aprendiera y fuera alguien mejor cada día.

A mi tutor por la confianza que depositó en nosotros y el apoyo que nos brindó para lograr el éxito del trabajo de diploma.

Asimismo, quisiera dar las gracias, a mi padrastro Rubén, por el apoyo incondicional que recibí de él, por la ayuda desinteresada que me dio y por preocuparse por mi y mis estudios en todo momento. Sinceramente sin él muchas cosas no hubieran sido posible.

Mi agradecimiento a mi papá y a todos mis hermanos por apoyarme y ayudarme en los momentos más difíciles de mi vida.

Quisiera extender mi gratitud a mi profesor, mi padre, mi amigo y mi hermano Villa, alguien que ha hecho por mi cosas inolvidables. De verdad lo admiro y lo respeto mucho.

Dar también un sincero agradecimiento a Arisuleidy, por haber llegado a mi vida y compartir conmigo casi los 5 años de carrera. Agradecerle su ayuda, apoyo y cariño. Decirle que a pesar de los momentos felices y tristes que hemos tenido, la quiero mucho y nunca la olvidaré.

Agradecer también a Roberto Tellez Ibarra quién fue mi compañero de tesis y junto a él logré aprender muchas cosas. Desearle mucha suerte en su vida y decirle que siempre lo tendré presente.

No quisiera olvidarme de mis compañeros y amigos que desde el comienzo de la carrera han compartido conmigo momentos inolvidables.

Por último y no por eso menos importante agradecer a todos los que de una forma u otra han contribuido al éxito de mi carrera.

Maikel Fernando

AGRADECIMIENTOS

A mi madre por su apoyo y preocupación. Te quiero mami. Muchas gracias por existir.

A mi padre, por enseñarme que las personas no son como uno quiere que sean.

A mi familia, esa que lo dio todo para que nunca faltase nada en mi vida.

A mi tutor Logendri Aguilera Mendoza por la confianza que depositó en nosotros desde el primer momento.

A mi compañero de tesis que estuvo a mi lado en la batalla final para obtener la victoria: ser ingeniero.

A todos mis compañeros de clase: el intercambio de conocimientos con ellos me ayudó a realizar un mejor trabajo y a ser una mejor persona.

A mis profesores de todas las enseñanzas que forman parte de lo que soy ahora, porque con mucha dedicación me nutrieron de conocimientos.

A mis hermanos y hermanas del grupo que nos unió cuando llegamos aquí: ustedes son únicos, pocos colectivos mantienen esta relación tan fuerte y tan sincera, a pesar de las diferencias. Gracias a ustedes, los que no están y los que se incorporaron, que nunca dejaron que la soledad invadiera nuestro espacio.

A mi pareja, compañeros, amigos y todos los que mostraron interés, preocupación o cualquier muestra de afecto durante esta etapa de mi vida.

A los que no estén reflejados en esta lista e igualmente merezcan mi agradecimiento.

A todos los que contribuyeron al éxito: ustedes siempre serán recordados. A todos: gracias.

Roberto

DEDICATORIA

A mis padres (Cuca y Nani), quienes me dieron la vida e hicieron posible el éxito de mi vida futura. Sin mi mamá (Cuca) nada hubiera sido posible; a ella le debo la vida, gracias por ayudarme en todos los momentos.

A mis hermanos Ismael (Chejo), Fidel (Fide), Norbelis (Norbe) y a mi hermana María de Lourdes (Marita) por la vida tan linda que hemos hecho juntos, los quiero mucho sinceramente. A mi padrastro Rubén (Chilo) por el apoyo que recibí de él y por todos los momentos felices que hemos vivido juntos.

A Villa por enseñarme y por darme su cariño y respeto.

A Arisuleidy por los momentos tan lindos e inolvidables que compartí con ella.

A mis amigos y compañeros que siempre los tendré presente.

En fin dedico mi trabajo de diploma a toda mi familia y a todos aquellos que de alguna forma u otra me ayudaron y depositaron toda su confianza en mi.

Maikel Fernando

A Marlisis Ibarra Santos: la razón de mi fuerza, mi impulso y mi existencia.

A mi familia que tanto dio para que lograra este triunfo.

A los que de cualquier manera forman parte de mi éxito.

A todos los que piensan en el futuro, a todos los que quieren aprender para compartir sus conocimientos.

Roberto

RESUMEN

En la Universidad de las Ciencias Informáticas (UCI) se desarrolló la Plataforma de Tareas Distribuidas v1.0 (T-arenal) para ofrecer una alternativa de cómputo que une en un solo conglomerado un grupo de estaciones de trabajo. A pesar de la aceptación y cantidad de clientes incorporados a esta versión, no era posible incrementar mucho más la cantidad de clientes debido a que la arquitectura limitaba las prestaciones del sistema.

El usuario interactúa con T-arenal 1.0 mediante una interfaz de escritorio que tiene varias limitaciones. Algunas de estas limitaciones son: se tiene que editar manualmente ficheros de texto para establecer la configuración de los clientes; el usuario corre el riesgo de no tener la versión más actualizada de la interfaz; el usuario no puede decidir cuando su ordenador será usado ni bajo qué condiciones; y esta aplicación no está integrada al Portal de Servicios del Polo de Bioinformática de la Facultad 6 de la UCI.

El trabajo que se presenta describe una solución al front-end de T-arenal 1.0, resolviendo las limitaciones descritas anteriormente y permitiendo la interacción con la nueva versión en desarrollo del Back-end de T-arenal. La fundamentación teórica, las características del sistema, así como el diseño e implementación serán tratados a través de los diferentes capítulos.

PALABRAS CLAVES T-arenal, Back-end, Front-end, Interfaz gráfica de usuario

Índice general

INTRODUCCIÓN 1

1. FUNDAMENTACIÓN TEÓRICA 5

1.1. INTERFACES DE USUARIO . . . 5

1.1.1. TIPOS DE INTERFACES DE USUARIO . . . 6

1.1.1.1. EVOLUCIÓN DE LAS INTERFACES DE USUARIO . . . 7

1.1.2. CARACTERÍSTICAS DE LAS INTERFACES GRÁFICAS DE USUARIO . . . 8

1.2. INTERFACES DE USUARIO Y SU ARQUITECTURA . . . 10

1.2.1. TIPOS DE EVENTOS . . . 10

1.2.2. FRONT-END Y BACK-END . . . 11

1.2.3. DISEÑO DE INTERFACES DE USUARIOS . . . 12

1.2.3.1. PASOS PARA EL DISEÑO DE UNA INTERFAZ . . . 13

1.2.3.2. REGLAS PARA EL DISEÑO DE INTERFACES DE USUARIOS . . . 14

1.3. ARQUITECTURA DE INFORMACIÓN . . . 16

1.3.1. CARACTERÍSTICAS DE LA ARQUITECTURA DE INFORMACIÓN . . . 17

1.3.2. EL ARQUITECTO DE INFORMACIÓN . . . 18

1.3.3. ORGANIZACIÓN DE LA INFORMACIÓN . . . 20

1.3.4. USABILIDAD . . . 21

1.4. HERRAMIENTAS Y TECNOLOGÍAS . . . 22

1.4.1. METODOLOGÍA DE DESARROLLO DE SOFTWARE . . . 22

1.4.2. LENGUAJE DE MODELADO . . . 24

1.4.3. HERRAMIENTA CASE . . . 25

1.4.4. IDE Y LENGUAJE DE PROGRAMACIÓN . . . 25

1.4.5. HERRAMIENTAS Y TECNOLOGÍAS PARA EL DESARROLLO . . . 26

1.5. CONCLUSIONES . . . 28

2. CARACTERÍSTICAS DEL SISTEMA 30

2.1. BREVE DESCRIPCIÓN DEL SISTEMA . . . 30

2.2. MODELO DE DOMINIO . . . 30

2.2.1. Descripción de los Objetos. 31
2.3. Especificación de los requisitos del sistema 32
2.3.1. Requisitos funcionales 32
2.3.2. Requisitos no funcionales 36
2.4. Actores y casos de uso del sistema. Su relación. 37
2.4.1. Actores del sistema 37
2.4.2. Casos de usos del sistema 37
2.4.3. Diagrama de casos de uso del sistema 40
2.4.3.1. Vista de casos de usos arquitectónicamente significativos . . . 42
2.5. Conclusiones 43

3. Diseño del sistema **44**
3.1. Estilo de arquitectura 44
3.2. Patrones de Diseño 46
3.2.1. Patrón Observador 46
3.2.2. Patrón Singleton (Instancia única) 47
3.2.3. Patrón Façade (Fachada) 47
3.2.4. Patrón Composite 48
3.3. Vista lógica 49
3.3.1. Diagramas de clases de diseño 49
3.3.2. Diagramas de Interacción 53
3.4. Vista de despliegue 57
3.4.1. Diagrama de despliegue 57
3.5. Conclusiones 58

4. Implementación y pruebas del Sistema **59**
4.1. Diagrama de Componentes 59
4.1.1. Diagrama de despliegue de los componentes 61
4.2. Pantallas de las aplicaciones. Resultados. 61
4.3. Modelo de pruebas 65
4.3.1. Casos de pruebas 66
4.4. Conclusiones 70

Conclusiones **71**

Recomendaciones **72**

Referencias bibliográficas **73**

Anexos **76**

GLOSARIO DE TÉRMINOS 86

Índice de figuras

1.1. Interfaz de preguntas y respuestas 7
1.2. Interfaz de menú simple 7
1.3. Interfaz orientada a ventanas 8
1.4. Capas de OpenUP 24

2.1. Diagrama de clases del Modelo de Dominio 31
2.2. Diagrama de casos de uso del sistema 40
2.3. Diagrama de Casos de usos del sistema del paquete *Front-end Cliente* 41
2.4. Diagrama de Casos de usos del sistema del paquete *Front-end Servidor* 42
2.5. Vista de casos de uso arquitectónicamente significativos 43

3.1. Modelo Vista Controlador 45
3.2. Flujo de control del MVC 45
3.3. MVC en Swing 46
3.4. Patrón Observador 47
3.5. Patrón Singleton 47
3.6. Patrón Fachada 48
3.7. Patrón Composite 48
3.8. Vista de los subsistemas de diseño 49
3.9. Diagrama de clases del diseño CU Autenticar usuario 50
3.10. Diagrama de clases del diseño CU Gestionar Problemas 50
3.11. Diagrama de clases del diseño CU Ejecutar Problemas 51
3.12. Diagrama de clases del diseño CU Gestionar Acceso al Disco Duro. 51
3.13. Diagrama de clases del diseño CU Gestionar Ejecuciones del Cliente 52
3.14. Diagrama de clases del diseño CU Gestionar configuración inicial 52
3.15. Diagrama de clases del diseño CU Gestionar Permisos 52
3.16. Diagrama de clases del diseño CU Gestionar uso del procesador 53
3.17. Diagrama de clases del diseño CU Gestionar visualización de mensajes 53
3.18. Diagrama de secuencia CU Autenticar usuario 54
3.19. Diagrama de secuencia CU Gestionar problemas 54

3.20. Diagrama de secuencia CU Ejecutar problemas 55
3.21. Diagrama de secuencia CU Gestionar acceso al disco duro 55
3.22. Diagrama de secuencia CU Gestionar configuración inicial 56
3.23. Diagrama de secuencia CU Gestionar permisos 56
3.24. Diagrama de secuencia CU Gestionar uso del procesador 57
3.25. Diagrama de despliegue del sistema 58

4.1. Diagrama de componentes 60
4.2. Ubicación de los componentes en los nodos del despliegue 61
4.3. Página principal del Portal con el Portlet Tarenal 62
4.4. Ventana principal del Front-end Servidor 63
4.5. Ventana preferencias del Front-end Cliente, gestionando Seguridad 64
4.6. Ventana preferencias del Front-end Cliente, gestionando Configuración inicial del Cliente . 65

Índice de cuadros

2.1. Actores del sistema y su descripción 37

Introducción

Los inicios de la informática actual se centran a fines de los años 40, cuando se construían computadores utilizando la tecnología disponible en aquel entonces. No tenían pantalla, ni teclado, ni sistema operativo, y la interacción entre las personas y los ordenadores se hacía a base de tarjetas perforadas o recableando las conexiones entre sus componentes.

Las grandes máquinas de los inicios, como el Mark I y el ENIAC, ocupaban un amplio espacio, pesaban varias toneladas y se demoraban una decena de segundos al hacer una operación matemática simple como una división. A través de los años los avances científico-técnicos han hecho posible pasar de aquellas gigantescas máquinas a los ordenadores actuales, algunos de los cuales caben en la palma de la mano y realizan millones de operaciones por segundo [1].

En la producción de software se utilizan los ordenadores para resolver problemas existentes en la sociedad. Estos sistemas informáticos necesitan interfaces para interactuar con los usuarios. La creación de las interfaces de usuario ha sido un área del desarrollo de software que ha evolucionado dramáticamente a partir de la década de los setentas. La interfaz de usuario es el vínculo entre el usuario y el programa de computadora. Una interfaz es un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa [2].

La elaboración de una interfaz de usuario, bien diseñada, exige una gran dedicación pues generalmente las interfaces son grandes, complejas y difíciles de implementar, depurar y modificar. Hoy en día las interfaces de manipulación directa son prácticamente universales. Las interfaces que utilizan ventanas, iconos y menús se han convertido en estándar en los materiales computacionales [3].

Situación Problémica

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 entre los que se destaca el Centro de Inmunología Molecular, Centro de Ingeniería Genética y Biotecnología, Centro de Química Farmacéutica y la Facultad de Química de la Universidad de La Habana.

Varios de los proyectos bioinformáticos que lleva la UCI, necesitan realizar una gran cantidad de cálculos que demoran un tiempo excesivamente largo en una sola computadora. De forma paradójica, la

UCI es, sin lugar a dudas, el centro a nivel nacional con el mayor número de computadoras personales. En estos momentos (año 2009) cuenta con más de 6 000 computadoras localizadas en toda la red universitaria. Es válido destacar que la mayoría de estas máquinas son Pentium 4 a 2.4 GHz de velocidad, tienen gran variedad de hardware, diferentes sistemas operativos y se encuentran conectadas mediante una red local cuya velocidad de transferencia es de 100 Mbps.

Para satisfacer la necesidad de los cálculos la Universidad dispone, al igual que otros centros del país, de un clúster compuesto por ocho nodos. No obstante, se está desaprovechando tiempo de procesamiento sobre todo en los horarios nocturnos o no laborables ya que existe un número elevado de computadoras de mesa que funcionan como estaciones de trabajo y que no forman parte del conglomerado para realizar los cálculos. Es por ello, que se creó una Plataforma de Tareas Distribuidas, con el objetivo de unir en un solo conglomerado un conjunto de computadoras distribuidas en una red LAN.

La primera versión de la plataforma ha dado buenos resultados, y ha sido de gran utilidad para varios proyectos [4]. Razón por la cuál se decide extender su uso, y aumentar el número de computadoras que utiliza de cientos a miles. En toda la red universitaria de la UCI existen miles de computadoras de escritorio, que en estos momentos no podrían ser aprovechadas eficientemente, ya que la arquitectura actual de la plataforma solo permite un único servidor con capacidad limitada de clientes. Este y otros motivos de refinamiento, han provocado el desarrollo de una nueva versión, la 2.0.

La segunda versión de la plataforma estará separada en dos partes esenciales: front-end y back-end. El front-end es la parte del software que interactúa con los usuarios y el back-end es la parte oculta que procesa los datos de entrada desde el front-end. El front-end de la versión anterior era una aplicación de escritorio aislada y el usuario podía no tener la versión más actualizada. Para la configuración del entorno de ejecución, se requería editar manualmente ficheros de texto con las políticas de seguridad. El presente trabajo de diploma brinda una nueva solución al front-end desechando el existente mientras que el back-end se encuentra en desarrollo.

La situación problémica enunciada anteriormente, da lugar al **problema científico**: ¿Cómo garantizar que los usuarios interactúen con la Plataforma de Tareas Distribuidas v2.0? Se define como **objeto de estudio**: Interacción hombre-ordenador y como **campo de acción**: Aplicaciones de Interfaz Gráfica de Usuario. El objetivo general es desarrollar el front-end para la Plataforma de Tareas Distribuidas v2.0. Del mismo se derivan los siguientes objetivos específicos:

- Realizar la ingeniería de los requisitos del front-end.
- Diseñar las aplicaciones que forman parte del front-end.
- Implementar las aplicaciones diseñadas.
- Realizar pruebas exploratorias a las aplicaciones implementadas.

Para darle cumplimiento a los objetivos se definen las siguientes tareas:

1. Estudio de las principales características de la Plataforma de Tareas Distribuidas v2.0.

2. Descripción de los requisitos funcionales y no funcionales del front-end.
3. Desarrollo del diagrama de casos de uso del front-end.
4. Descripción de los casos de uso del front-end.
5. Descripción de la arquitectura del front-end.
6. Desarrollo del diagrama de clases del diseño.
7. Implementación de las aplicaciones.
8. Realización de pruebas de caja negra.

Importancia

La interacción con el Back-end es necesaria para posibilitar la gestión del sistema y verificar su correcto funcionamiento. Como consecuencia de la realización de una nueva versión del Back-end, es imprescindible contar con un Front-end que soporte las nuevas funcionalidades incluidas y que brinde una solución a las limitaciones existentes en la versión 1.0.

Con la realización de este nuevo Front-end se logra que la información esté mejor organizada, sea mucho más fácil la instalación y distribución de las aplicaciones y la usabilidad de las aplicaciones permita que los usuarios utilicen la interfaz cómodamente.

Estructura

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

- **Capítulo 1: Fundamentación Teórica**: En este capítulo se realiza y se muestra un estudio acerca del desarrollo de las interfaces gráficas de usuario. Además se hace una breve valoración de las técnicas, plataforma, librerías y procedimientos utilizados para dar cumplimiento al desarrollo del front-end. Finalmente se hace una descripción de las herramientas y tecnologías a utilizar para desarrollar la solución propuesta.
- **Capítulo 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. Además, se presenta el diagrama de casos de usos del sistema, así como una breve descripción de los casos de usos identificados.

- **Capítulo 3: Diseño del sistema**: En este capítulo se dará 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 del sistema.
- **Capítulo 4: Implementación y pruebas del sistema**: En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Se ilustrarán los principales resultados obtenidos. Además, los casos de prueba para los principales casos de uso.

Capítulo 1

Fundamentación Teórica

En este capítulo se realiza y se muestra un estudio acerca del desarrollo de las interfaces gráficas de usuario. Además se hace una breve valoración de las técnicas, plataforma, librerías y procedimientos utilizados para dar cumplimiento al desarrollo del front-end. Finalmente se hace una descripción de las herramientas y tecnologías a utilizar para desarrollar la solución propuesta.

1.1. Interfaces de Usuario

El término interfaz, según la Real Academia Española [5] proviene del inglés interface y significa: conexión física y funcional entre dos aparatos o sistemas independientes.

La creación de las interfaces de usuario ha sido un área del desarrollo de software que ha evolucionado dramáticamente a partir de la década de los setentas. La interfaz de usuario es el vínculo entre el usuario y el programa de computadora. Una interfaz es un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa. [2]

Otras definiciones de interfaz gráfica de usuario son:

> "Las interfaces gráficas de usuario (GUI)[1] son aquellas que incluyen cosas como menús, ventanas, teclado, ratón y algunos otros sonidos que la computadora hace, en general, todos aquellos canales por los cuales se permite la comunicación entre el hombre y la computadora." [6]
>
> "...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..." [7]

[1] En Inglés, Graphical User Interface

La definición más acertada para esta terminología, según la opinión de los autores, acorde con la esfera informática y el desarrollo de software, es la siguiente:

> "La interfaz gráfica de usuario es aquella que permite a los usuarios interactuar con las aplicaciones informáticas, mediante el uso de un lenguaje visual [8] y un conjunto de acciones posibles."

La interfaz de usuario es una de las partes más importantes de cualquier programa ya que determina que tan fácilmente es posible que el programa haga lo que el usuario quiere hacer. Un programa muy poderoso con una interfaz pobremente elaborada tiene poco valor para un usuario no experto. La elaboración de una interfaz de usuario, bien diseñada, exige una gran dedicación pues generalmente las interfaces son grandes, complejas y difíciles de implementar, depurar y modificar. [3]

1.1.1. Tipos de interfaces de usuario

Se conocen varios tipos de interfaces de usuario. Atendiendo a como el usuario puede interactuar con una interfaz, nos encontramos con:

- Interfaces alfanuméricas (intérpretes de mandatos) que solamente presentan texto.
- Interfaces gráficas de usuario, las que permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.
- Interfaces táctiles, que representan gráficamente un "panel de control" en una pantalla sensible que permite interaccionar con el dedo de forma similar a si se accionara un control físico.

Las interfaces de usuario pueden ser de software o de hardware:

- Interfaces de hardware: Se trata de un conjunto de controles o dispositivos que permiten la interacción hombre-máquina, de modo que permiten introducir o leer datos del equipo, mediante pulsadores, reguladores e instrumentos.
- Interfaces de software: Son programas o parte de ellos, que permiten expresar nuestros deseos al ordenador o visualizar su respuesta [9].

1.1.1.1. Evolución de las interfaces de usuario

En los primeros días de las computadoras (antes de las pantallas gráficas, el ratón, etc.) la interfaz de preguntas y respuestas era la única forma realista que existía de una interfaz. El usuario podía comunicarse con el sistema específico con órdenes de la forma indicada en la figura. Aunque es una forma concisa, es muy propensa a errores, muy estricta y difícil de aprender.

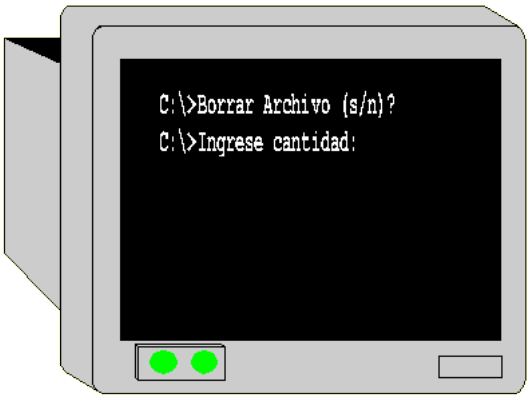


Figura 1.1: Interfaz de preguntas y respuestas

- Interfaz de menú simple: Es una variante de la forma anterior, se presenta al usuario una lista de opciones y la selección se realiza por medio de un número, letra o un código en particular. Ofrece al usuario un contexto global y tiene menos porcentaje de errores que el anterior, pero su uso puede llegar a ser tedioso. La figura 1.2 muestra una interfaz de menú simple (que a su vez puede incluir otras opciones dentro de las principales).

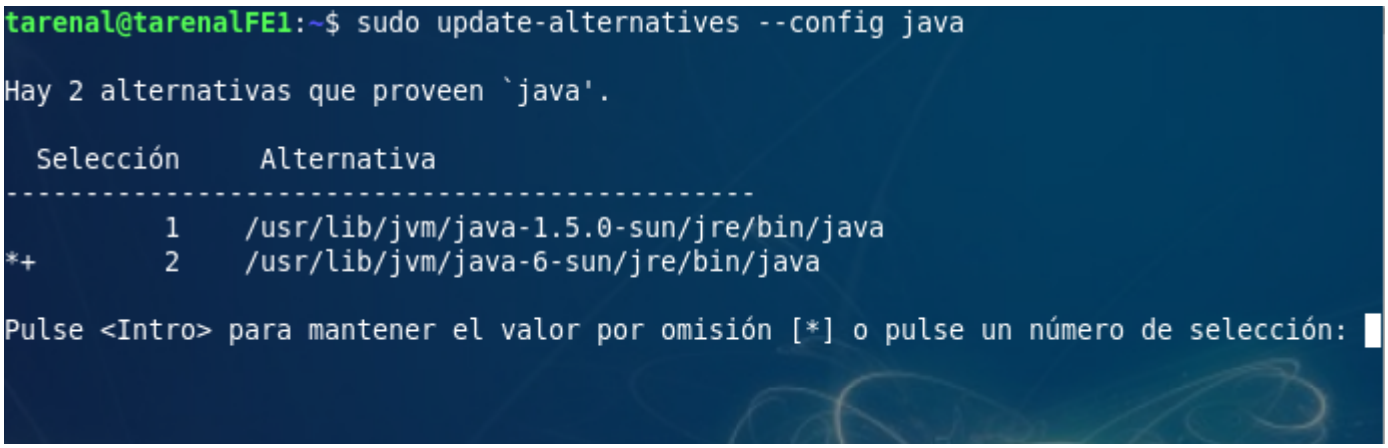


Figura 1.2: Interfaz de menú simple

- Interfaz orientada a ventanas: A medida que el hardware se ha hecho mas eficiente y los ingenieros

de software han aprendido más sobre los factores humanos, las técnicas de interfaz evolucionaron, llegando a lo que se conoce como interfaces de la tercera generación. Ofrecen al usuario las siguientes ventajas:

1. Se puede visualizar diferentes tipos de información simultáneamente.
2. El esquema de menús desplegables permite realizar muchas tareas interactivas diferentes.
3. Se realizan tareas de control y de diálogo en forma sencilla.
4. La utilización de menús desplegables, botones y técnicas de presentación reducen el manejo del teclado.

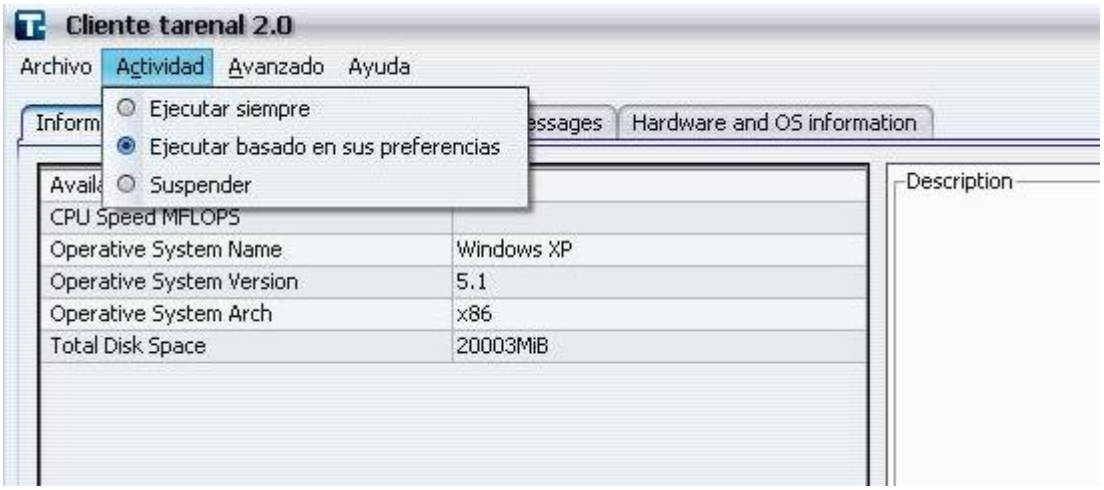


Figura 1.3: Interfaz orientada a ventanas

- Interfaz de cuarta generación: Esta es la generación actual. Une todos los atributos de la generación anterior con el hipertexto y la multitarea (varias tareas simultáneamente). [10]

1.1.2. Características de las interfaces gráficas de usuario

Para usar las interfaces gráficas de usuario, los usuarios deben conocer o aprender una serie de elementos: organización del sistema, diferentes tipos de iconos y efecto de las acciones sobre ellos, elementos básicos de una ventana, uso de los controles del GUI y uso del ratón. Las interfaces gráficas de usuario se diferencian de otros tipos de interfaz. Las características que más se destacan son:

- Posee un monitor gráfico.[2]
- Posee un dispositivo apuntador (típicamente un ratón o mouse).
- Los usuarios pueden ver en la pantalla los gráficos y textos tal como se verán impresos.
- Sigue el paradigma de la interacción objeto-acción [11].
- Permite la transferencia de información entre programas.
- Se puede manipular en la pantalla directamente los objetos y la información.
- Provee elementos de interfaz estándar como menús y diálogos.
- Existe una muestra visual de la información y los objetos (iconos y ventanas).
- Proporciona respuesta visual a las acciones del usuario.
- Existe información visual de las acciones y modos del usuario/sistema (menús, paletas).
- Existen widgets[3] para la selección e introducción de la información.
- Permite a los usuarios personalizar la interfaz y las interacciones.
- Proporciona flexibilidad en el uso de dispositivos de entrada (teclado/ratón).
- Permite a los usuarios personalizar la interfaz (apariencia) y las interacciones.

Las interfaces gráficas de usuario deben ser diseñadas pensando en el usuario final que consumirá el sistema. Sin tener este factor en consideración, es muy poco probable que la calidad del producto sea óptima [12]. Por eso se debe tener en cuenta:

1. Velocidad de Aprendizaje: Se pretende que la persona aprenda a usar el sistema lo más pronto posible.
2. Velocidad de Respuesta: El tiempo necesario para responder ante una operación en el sistema debe ser mínimo.
3. Tasa de errores: Porcentaje de errores que comete el usuario.
4. Retención: Cuánto recuerda el usuario sobre el uso del sistema en un período de tiempo.
5. Satisfacción: Se refiere a que el usuario esté a gusto con el sistema.

[2] Pantalla de alta resolución que sirve para visualizar la salida de la información de un ordenador.
[3] Controles gráficos

1.2. Interfaces de usuario y su arquitectura

El modo en que el usuario se comunica con una aplicación para solicitar los recursos del sistema operativo constituye la interfaz del mismo. La interfaz es particularmente importante para establecer una vinculación amigable entre el usuario de la computadora y la aplicación.

1.2.1. Tipos de eventos

Los eventos generados tanto por el usuario o el sistema, los cuales deben ser soportados por las implementaciones de interfaces de usuario, son los siguientes:

- Evento de ratón. Ocurre cuando el usuario mueve el apuntador del ratón dentro o fuera de una ventana, hace clic en el botón dentro o fuera de la ventana o libera el botón del ratón.
- Evento de teclado. Ocurre cuando el usuario oprime o libera una tecla del teclado.
- Evento de menú. Ocurre cuando el usuario selecciona un comando desde un menú.
- Evento de actualización de ventana. Ocurre cuando una porción de la imagen de la presentación de una aplicación ha sido alterado (posiblemente por que se sobrepuso otra ventana) y se tenga que restablecer.
- Evento de ajuste. Ocurre cuando el usuario ha modificado el tamaño de una ventana.
- Evento de Activación/Desactivación. Se generan por la interfaz de usuario para permitirle al usuario activar y desactivar ventanas.
- Evento de Inicializar/Terminar. Ocurre cuando una entidad interfaz de usuario se ha creado o destruido.

Distribución de eventos

Los eventos deben ser procesados por la lógica de presentación en cooperación con la lógica de procesamiento. El procesamiento puede ser distribuido entre la misma interfaz, la lógica de la aplicación y la Interfaz de Programación de Aplicaciones (API[4]) de la interfaz de usuario. El procesamiento de los eventos se puede distribuir de la siguiente manera [13]:

- Modelo de ciclo de evento. Especifica que una aplicación debe contener un ciclo de evento. El ciclo de evento llama a una rutina de librería en particular para ver si hay eventos pendientes. Cada

[4] En Inglés, Aplication Programming Interface. Es un conjunto de rutinas que hacen funciones como crear ventanas y desplegar varios gráficos

evento pendiente causa que la aplicación atienda el evento antes de regresar el control al ciclo de evento.

- Modelo de aviso de evento. Requiere que la aplicación registre una función manejadora de eventos por cada entidad de interfaz que crea; de esta manera se libera a la aplicación de una sobre carga con el ciclo de evento. Cuando la interfaz gráfica de usuario detecta un evento para una ventana, llama a la rutina apropiada de la aplicación. La aplicación tiene el control sólo cuando se inicia una entidad o cuando se llama a una de sus rutinas de atención a eventos.
- Modelo híbrido. Combina el modelo ciclo de evento y el modelo aviso de evento. Microsoft Windows emplea un modelo (donde una aplicación debe contener un ciclo de evento) el cuál llama a una rutina para obtener el siguiente evento. En ese momento, una aplicación puede llamar a otra rutina del API, el cual puede, en su momento, llamar al manejador de eventos de la aplicación.

1.2.2. Front-end y Back-end

Básicamente existen dos tipos de interfaces, las interfaces estáticas y las interfaces dinámicas [13]. Las interfaces estáticas son aquellas que no tienen cambio y son difíciles de modificar y por su ubicación pueden ser:

- De propósito especial (stand-alone).
- Centralizado (novell).
- Distribuido (internet).

Las interfaces dinámicas son aquellas que cambian de acuerdo a los requerimientos del usuario y por su uso pueden ser:

- Front-End.
- Back-End.

Interfaz Front-End: Es una aplicación donde los usuarios interactúan directamente con las funciones del sistema; cubre todas las interfaces con las cuales un usuario interactúa con los sistemas, ya sean locales o remotos. Sus funciones principales son:

- Diseño de formatos.
- Presentación.
- Lógica de la aplicación.
- Manipulación de datos.
- Herramientas de consulta.
- Utilerías/menús.

Interfaz Back-End: Es un conjunto de elementos (programas) que sirven como complemento de una interfaz Front-End. Ayuda en la administración, control y configuración de los sistemas teniendo un acceso directo a los recursos (base de datos, comunicaciones, servidores, etc.), que el sistema requiere, entre sus funciones principales se tienen:

- Administración de la memoria.
- Seguridad.
- Manejo de base de datos.
- Procesamiento remoto.

1.2.3. Diseño de interfaces de usuarios

Al diseñar interfaces de usuario deben tenerse en cuenta las habilidades cognitivas y de percepción de las personas y adaptar el programa a ellas.

Así, una de las cosas más importantes que una interfaz puede hacer es reducir la dependencia de las personas de su propia memoria, no forzándoles a recordar, por ejemplo, información que apareció en una pantalla anterior, o a repetir operaciones ya realizadas (por ejemplo, introducir un mismo dato varias veces) [14].

Deben ser tomados en cuenta varios atributos para lograr un buen diseño de las interfaces de usuario:

- Características Físicas: Cada persona tiene diferentes características físicas. Hay algunas personas que no les gustan los teclados mientras que a otras sí.
- Ambiente: El lugar donde va a ser usado el sistema. Cada interfaz tiene que adecuarse al lugar.
- Visibilidad: Tomar en cuenta la cantidad de iluminación del lugar.
- Personalidad: De acuerdo a la edad, nivel socio-económico, etc.

1.2.3.1. Pasos para el diseño de una interfaz

En el proceso de diseño de una interfaz de usuario se pueden distinguir cuatro fases o pasos fundamentales [15]:

1. Reunir y analizar la información del usuario.
   Concretar a través de técnicas de requerimientos, qué tipo de usuarios va a utilizar el programa, qué tareas van a realizar los usuarios y cómo las van a realizar, qué exigen los usuarios del programa, en qué entorno se desenvuelven los usuarios (físico, social, cultural).

2. Diseñar la interfaz de usuario.
   Es importante dedicar tiempo y recursos a esta fase, antes de entrar en la codificación. En esta fase se definen los objetivos de usabilidad del programa, las tareas del usuario, los objetos y acciones de la interfaz, los iconos, vistas y representaciones visuales de los objetos, los menús de los objetos y ventanas. Todos los elementos visuales se pueden hacer primero a mano y luego refinar con las herramientas adecuadas.

3. Construir la interfaz de usuario.
   Es interesante realizar un prototipo previo, una primera versión del programa que se realice rápidamente y permita visualizar el producto para poderlo probar antes de codificarlo definitivamente.

4. Validar la interfaz de usuario.
   Se deben realizar pruebas de usabilidad del producto, de ser posible con los propios usuarios finales del mismo. Por lo tanto, es importante realizar un diseño que parta del usuario y no del sistema.

Existen otros elementos avanzados a tener en cuenta para lograr una interfaz óptima. Entre estos podemos citar [15]:

- Presentación de información
  No se deben colocar demasiados objetos en la pantalla, y los que existen deben estar bien distribuidos. Cada elemento visual influye en el usuario no sólo por sí mismo, sino también por su combinación con el resto de elementos presentes en la pantalla.

- Análisis de Color
  Probablemente es el elemento de la interfaz que con más frecuencia es mal utilizado. El color comunica información, no es sólo decorativo (ejemplo: reforzar mensajes de error). Deben utilizarse combinaciones adecuadas. Hay que intentar mantener una coherencia, buscando siempre transmitir una sensación de bloque homogéneo durante toda la aplicación.

- Guías de Expertos
  Existen diversas guías de diseño sacadas de expertos y comités que complementan a las reglas vistas anteriormente. Por citar algunas de ellas:

  - Demasiada simetría puede hacer las pantallas difíciles de leer.
  - Elementos de tamaño y color similares se perciben como pertenecientes a un grupo.
  - Asumir errores en la entrada del usuario.
  - Diseñar para el usuario, no para demostrar los propios conocimientos tecnológicos.
  - Unos gráficos espectaculares no salvarán a una mala interfaz.

- Utilidad
  Este concepto es previo a todo estudio de usabilidad: si un producto no es útil, al usuario no le va a interesar si es usable o no. El concepto "utilidad" se refiere al grado en que un producto ofrece la posibilidad al usuario de conseguir sus metas y como valoración de la motivación del usuario por utilizar ese producto: si una interfaz es de fácil uso y aprendizaje pero no permite al usuario lograr sus objetivos no será utilizada.

1.2.3.2. Reglas para el diseño de interfaces de usuarios

Existen una serie de principios [16] a seguir en el desarrollo de interfaces de usuario [15]:

1. Dar control al usuario.

   El diseñador debe dar al usuario la posibilidad de hacer su trabajo, en lugar de suponer qué es lo que éste desea hacer. La interfaz debe ser suficientemente flexible para adaptarse a las exigencias de los distintos usuarios del programa.

   En concreto, se pueden enumerar los siguientes principios que permiten al usuario estar en posesión del control:

   - Usar adecuadamente los modos de trabajo.
   - Permitir a los usuarios utilizar el teclado o el ratón.
   - Permitir al usuario interrumpir su tarea y continuarla más tarde.
   - Utilizar mensajes y textos descriptivos.
   - Permitir deshacer las acciones e informar de su resultado.
   - Permitir una cómoda navegación dentro del producto y una fácil salida del mismo.

- Permitir distintos niveles de uso del producto para usuarios con distintos niveles de experiencia.
- Hacer transparente la interfaz al usuario, que debe tener la impresión de manipular directamente los objetos con los que está trabajando.
- Permitir al usuario personalizar la interfaz (presentación, comportamiento e interacción).
- Permitir al usuario manipular directamente los objetos de la interfaz.

2. Reducir la carga de memoria del usuario.

   La interfaz debe evitar que el usuario tenga que almacenar y recordar información. Para ello, debe seguir los siguientes principios:

   - Aliviar la carga de la memoria de corto alcance (mantener los últimos datos introducidos).
   - Basarse en el reconocimiento antes que en el recuerdo (ejemplo: elegir de entre una lista en lugar de teclear de nuevo).
   - Proporcionar indicaciones visuales de dónde está el usuario, qué está haciendo y qué puede hacer a continuación.
   - Proporcionar funciones deshacer, rehacer y acciones por defecto.
   - Proporcionar atajos de teclado (iniciales en menús, teclas rápidas).
   - Asociar acciones a los objetos (menú contextual).
   - Utilizar metáforas del mundo real.
   - Presentar al usuario sólo la información que necesita (menús simples/avanzados, asistentes).
   - Hacer clara la presentación visual (colocación/agrupación de objetos, evitar la presentación de excesiva información).

3. Consistencia.

   Permite al usuario utilizar conocimiento adquirido en otros programas consistentes con el nuevo programa. Ejemplo: mostrar siempre el mismo mensaje ante un mismo tipo de situación, aunque se produzca en distintos lugares.

   - Consistencia en la realización de las tareas: proporcionar al usuario indicaciones sobre el proceso que está siguiendo.
   - Consistencia dentro del propio producto y de un producto a otro. La consistencia se aplica a la presentación (lo que es igual debe aparecer igual: color del texto estático), el comportamiento (un objeto se comporta igual en todas partes) y la interacción (los atajos y operaciones con el ratón se mantienen; el usuario espera los mismos resultados cuando interactúa de la misma forma con objetos distintos).

- Consistencia en los resultados de las interacciones: misma respuesta ante la misma acción. Los elementos estándar de la interfaz deben comportarse siempre de la misma forma (las barras de menús despliegan menús al seleccionarse).
- Consistencia de la apariencia estética (iconos, fuentes, colores, distribución de pantallas).
- Fomentar la libre exploración de la interfaz, sin miedo a consecuencias negativas.

1.3. Arquitectura de información

En 1989, Tim Berners-Lee, dio lugar a la popularización de Internet. Desde aquel entonces, el número de páginas aumenta cada día y se calcula que crecen en alrededor de 50 páginas y 500 enlaces sólo durante el tiempo que toma leer una sencilla oración. La dinámica de la tecnología de la información obliga a perfilar el intelecto en relación con los contenidos y la forma de presentarlos en la Web, debido a la incorporación constante de un elevado número de usuarios de todo tipo a la red [17].

Si se estudia el fenómeno de Internet y de toda la información que ella contiene, se encuentra que sus principales dificultades radican en que no siempre es posible recuperar la información existente sobre un tema porque, entre otras razones, se dificulta la recuperación de la información por una inadecuada organización de los contenidos. Es precisamente de esta organización de los contenidos en los sitios web que se ocupa la llamada Arquitectura de la Información (AI) [18].

Con respecto a la AI, Wurman se refirió a ella como:

> "... la ocupación profesional emergente del siglo XXI, dedicada a las necesidades de la era, enfocada a la claridad, entendimiento humano y la ciencia de organización de la información..." [19]

Edward Tufte, por su parte, la define como:

> "... el diseño de la presentación de la información para facilitar su entendimiento..." [20]

Rosenfeld y Morville, afirman que es la actividad que:

> "... Clarifica la misión y visión del sitio y equilibra las necesidades del patrocinador y de la audiencia... determina el contenido y funcionalidad que el sitio dispondrá... especifica cómo los usuarios encontrarán la información al definir su organización, navegación, etiquetado y sistemas de búsqueda... mapea como el sitio se va a acomodar al cambio y el crecimiento en el tiempo..." [21]

James Garret, Jesse, en "Elements of user experience", establece que

> "... la Arquitectura de información es el diseño estructural del espacio informacional para facilitar el acceso intuitivo a los contenidos" [22]

La Arquitectura de la Información (AI) es la disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactivos y no interactivos.

En relación con la World Wide Web el Instituto de Arquitectura de Información, define la Arquitectura de la Información como:

> “El diseño estructural en entornos de información compartida.
>
> El arte y la ciencia de organizar y rotular sitios web, intranet, comunidades en línea y software para promover la usabilidad y la accesibilidad.
>
> Una comunidad emergente orientada a aplicar los principios del diseño y la arquitectura en el entorno digital.“ [23]

La Arquitectura de la Información trata indistintamente del diseño de: sitios web, interfaces de dispositivos móviles o gadgets (como los iPod), CD interactivos, videoclips digitales, relojes, tableros de instrumentos de aviones de combate o civiles, interfaces de máquinas dispensadoras, interfaces de juegos electrónicos, etc.

Su principal objetivo es facilitar al máximo los procesos de comprensión y asimilación de la información, así como las tareas que ejecutan los usuarios en un espacio de información definido.

Finalmente, la Arquitectura de Información puede considerarse como la disciplina que dispone y determina los contenidos de información y estructurales de un sitio web u otra interfaz, a partir de las necesidades y preferencias de la audiencia, con el objetivo de garantizar la calidad final del producto y la plena satisfacción de los usuarios. [18]

1.3.1. Características de la Arquitectura de Información

Después de haber hecho un análisis de las diferentes definiciones de AI, nos adelantaremos exponiendo algunas conclusiones generales:

1. La AI tiene una estrecha relación con las ciencias Bibliotecológicas ya que el tratamiento que se le da a la información, ya sea en los aspectos de organización, categorización, gestión de información, vocabularios controlados y establecimientos de metadatos son inherentes a esta disciplina.

2. La AI en su global aplicación conforma una metodología de trabajo (administración, técnica y arte) que asegura la buena experiencia de los usuarios al interactuar con la interfaz Web.

3. La AI tiene un carácter multidisciplinario ya que los conocimientos ideales para poder llevarla a cabo son:

   - Ciencias Bibliotecológicas
   - Diseño
   - Periodismo
   - Informática
   - Interacción Humano Computador
   - Mercadotecnia y Administración de proyectos Web. [18]

Arquitectura de la Información como abstracción

La Arquitectura de la Información no se trata del establecimiento de un conjunto de pasos o guías predefinidas, sino del diseño inteligente que subyace detrás de una interfaz o espacio de información. Trata de maximizar las sinergias que se producen entre interactividad, navegación y contenido con el objetivo de alcanzar una integración sistémica con el cerebro del usuario logrando un fenómeno de persuasión, conocimiento o información simbiótica que se traspasa de un sistema de información a otro.

De esta forma las acciones de búsqueda, disponibilidad y recuperación de los contenidos se realizan en un contexto óptimo en ambos aspectos del sistema de información (interfaz y usuario), interactúan iniciándose un proceso de comunicación que los enriquece mutuamente. Por un lado la interfaz cumple con su objetivo y puede ser mejorada y por otro, el usuario encuentra lo que busca, lo asimila con facilidad y lo utiliza.

1.3.2. El Arquitecto de Información

Richard Saul Wurman fue el primero en utilizar el término "arquitectos de la información". Wurman los definió como:

> "... personas que organizan los patrones inherentes en los datos, que hacen claro lo complejo... una persona que crea el mapa o la estructura de información que permite a otros encontrar su camino personal al conocimiento..." [19]

La AI es una nueva profesión que surge en 1996 a raíz de la evolución y transformación de la World Wide Web en un canal y medio masivo de comunicación. Su aparición en un contexto social, cultural, económico y político está fuertemente condicionada por las Nuevas Tecnologías de la Información, tecnologías que han modificado bruscamente y a todos los niveles las formas de comunicación entre los seres humanos, así como la manera en que perciben y asimilan información [24].

Estos avances en telecomunicaciones, ciencia, y tecnología en general han producido una cantidad ingente de conocimiento, de nuevos conceptos, ideas, métodos, procesos, visiones, problemas y soluciones sobre las que interviene la Arquitectura de la Información que, en concreto busca [25]:

- **Procesar** y **dosificar** la enorme cantidad de información que se ha producido a causa de los descubrimientos, nuevas investigaciones en todos y nuevos campos, a causa de la revolución de Internet y ponerla de una manera clara, relevante y significativa a disposición del usuario común, se trata, entre otras cosas, de hacer comprensible lo abstracto de alguna forma.
- **Desarrollar** y **verificar** procesos de producción o diseño de información con el fin de que el usuario pueda recuperar la información de un determinado espacio de manera clara, precisa y sin ambigüedades, en cualquier plataforma o soporte; en especial hablamos de soportes multimedia e interactivos, aunque en la práctica no debemos omitir ningún soporte por plano que este sea y hablar de experiencias de usuario.
- **Organizar**, **estructurar**, **sistematizar**, **rotular**, **distribuir** [20], diseñar estructuralmente sistemas de información con el fin de que el usuario pueda hacer de su experiencia de recuperación algo simple, agradable, eficaz y productivo.

Un arquitecto de información debe reunir un mínimo de conocimientos procedentes de diferentes disciplinas, entre ellos se encuentran [18]:

- Diseño gráfico: No implica ser diseñador gráfico, ni dominar por completo una herramienta de diseño. Se refiere a la habilidad de establecer relaciones entre los elementos visuales y determinar su total integración dentro de la Web.
- Documentación e información: la documentación se basa en el estudio y creación de medios de acceso a la información, así como determinar la forma más apropiada de organizarla para garantizar su posterior recuperación. Estos son métodos adecuados para iniciar una arquitectura de información.
- Periodismo: la habilidad para comunicar y escribir es inherente a esta profesión. Deben considerarse las diferencias entre los estilos de redacción para Web y papel.
- Marketing: los conocimientos sobre investigaciones de usuarios o audiencias, así como la identificación de segmentos atractivos del mercado constituyen la labor diaria de estos especialistas. La Web como producto no puede permanecer ajena a ello.
- Informática: resulta de suma importancia el conocimiento del entorno tecnológico de la Web. A partir de ello, pueden establecerse limitantes y definir el alcance de las prestaciones que se desean implementar en el sitio.

- Ingeniería en usabilidad: comprende la habilidad y los métodos para evaluar el funcionamiento del sistema, desde la curva de aprendizaje hasta los errores más frecuentes que comenten los usuarios.

Para ser arquitecto de información no es necesario ser un especialista en las profesiones anteriormente referidas. Sin embargo, la labor del arquitecto de información va más allá: incluye el control de los flujos vinculados al proceso de trabajo del equipo de desarrollo, así como la coordinación entre las distintas disciplinas que integran el equipo. [18]

1.3.3. Organización de la información

La organización de la información es el proceso donde se dispone y ordena la secuencia de los elementos que integran el contenido de un sitio web. En este proceso, se consideran las características de los sistemas de clasificación y ordenamiento como son la ambigüedad, la heterogeneidad y la homogeneidad. También, se seleccionan los esquemas de organización de la información y las estructuras de organización de la información que se utilizarán en el sitio [18].

La AI trata de organizar la información de manera que los usuarios puedan encontrar las respuestas correctas a sus interrogantes. Muchas veces, los usuarios tienen que esforzarse durante la realización de actividades tan simples como pueden ser navegar ociosamente en Internet o ejecutar una búsqueda sencilla. Su objetivo radica en la creación de sistemas de organización de la información y etiquetados -términos que designan o describen una entidad que realmente posean un significado para los usuarios [26].

La información puede estar organizada de varias maneras. Entre ellas, atendiendo al tipo de interfaz, podemos mencionar:

- Sistemas de navegación

  La razón para diseñar correctamente un Sistema de Navegación (SN) radica en prevenir que los usuarios puedan hallarse perdidos frente a nuestra interfaz y experimenten sensaciones de confusión, frustración e ira.

- Menús Desplegables

  Los menús desplegables facilitan el acceso a múltiples elementos de navegación de forma compacta. Los usuarios pueden desplegar lo que parece un simple menú y acceder a diferentes opciones. Un menú es una lista de acciones que se encuentran etiquetadas de manera corta, no ambigua, clara y consistente.

- Las Tablas de Contenido (TC)

  La calidad del diseño de las TC afecta significativamente su usabilidad. Es importante prestar atención a los siguientes aspectos:

1. Jerarquice coherentemente la información de manera que los usuarios puedan familiarizarse con la organización que tiene el contenido.
2. Haga viable de manera rápida el acceso directo a los contenidos para aquellos usuarios que saben lo que están buscando. No haga que se pierdan en un diseño ambiguo.
3. Tenga cuidado de no atiborrar al usuario con demasiada información. Recuerde que el objetivo es ayudar y no asustar al usuario.

1.3.4. Usabilidad

La Organización Internacional para la Estandarización (ISO) [27] ofrece dos definiciones de usabilidad:

1. "La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso" [ISO/IEC 9126]
2. "Usabilidad es la eficacia, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico" [ISO/IEC 9241]

El término también se refiere a los métodos para mejorar la facilidad de uso durante el proceso de diseño. El grado de usabilidad de un sistema es, por su parte, una medida empírica y relativa de la usabilidad del mismo. Empírica porque no se basa en opiniones o sensaciones sino en pruebas de usabilidad, realizadas en laboratorio u observadas mediante trabajo de campo. Relativa porque el resultado no es ni bueno ni malo, sino que depende de las metas planteadas o de una comparación con otros sistemas similares.

A partir de la conceptualización llevada a cabo por la ISO, se infieren los principios básicos en los que se basa la usabilidad [28]:

- Facilidad de Aprendizaje: facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema o producto. Está relacionada con la predictibilidad, sintetización, familiaridad, la generalización de los conocimientos previos y la consistencia.
- Flexibilidad: relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema.
- Robustez: es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos. Está relacionada con la capacidad de observación del usuario, de recuperación de información y de ajuste de la tarea al usuario.

Entre los principales beneficios encontramos:

- Reducción de los costes de aprendizaje.
- Disminución de los costes de asistencia y ayuda al usuario.
- Optimización de los costes de diseño, rediseño y mantenimiento.
- Mejora la imagen y el prestigio.
- Mejora la calidad de vida de los usuarios, ya que reduce su estrés, incrementa la satisfacción y la productividad.

Todos estos beneficios implican una reducción y optimización general de los costes de producción, así como un aumento en la productividad. La usabilidad permite mayor rapidez en la realización de tareas y reduce las pérdidas de tiempo [18].

1.4. Herramientas y tecnologías

Existen creencias de que un grupo de desarrollo debería organizarse en torno a las habilidades de los individuos altamente calificados, que saben como hacer el trabajo y lo hacen bien y que raramente necesitan dirección. Esto constituye un error en la mayoría de los casos y una grave equivocación en el caso de desarrollo de software. Por lo tanto, es necesario un proceso que esté ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo en el que se encuentran implicados. Todo esto unido a una correcta selección de las herramientas, métodos, técnicas y procedimientos contribuyen a obtener un producto de elevada calidad.

1.4.1. Metodología de desarrollo de software

Uno de los principales retos que enfrentan los desarrolladores de software es el desarrollo con calidad y de manera eficiente, por lo cual supone un paso importante hacer una selección correcta de las herramientas y procesos necesarios. Con este propósito se crearon diferentes metodologías de desarrollo, las cuales son un conjunto de pasos y procedimientos que deben seguirse para el desarrollo de software. Con los años y las experiencias que han ido teniendo las diferentes instituciones desarrolladoras de software, se han ido mejorando algunas metodologías de desarrollo y se han creado otras con el objetivo de aumentar la productividad.

Entre la familia de metodologías existen las metodologías ágiles, las cuales intentan evitar los intrincados y burocráticos caminos de las metodologías tradicionales enfocándose en las personas y los

resultados [29]. Estas metodologías recogen ventajas de las metodologías tradicionales e incorporan características nuevas que hacen que el proceso de desarrollo sea más simple, basándose en que lo más importante en un proyecto es valorar más a los individuos que a los procesos y herramientas, al software que funciona más que a la documentación exhaustiva, a la colaboración del cliente más que a la negociación contractual, a la respuesta al cambio más que al seguimiento de un plan.

Elegir la metodología adecuada para realizar este ciclo de desarrollo es vital para lograr un sistema de alta calidad en un tiempo razonablemente corto. Existen varias metodologías de desarrollo de software, dentro de las cuales están RUP [30], MSF [31], XP [32] y OpenUP.

Esta última, es una de las metodologías ágiles de desarrollo de software. Ofrece las mejores prácticas de una variedad de líderes en ideas sobre desarrollo de software y comunidades de desarrollo que cubren un diverso conjunto de perspectivas y necesidades 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.

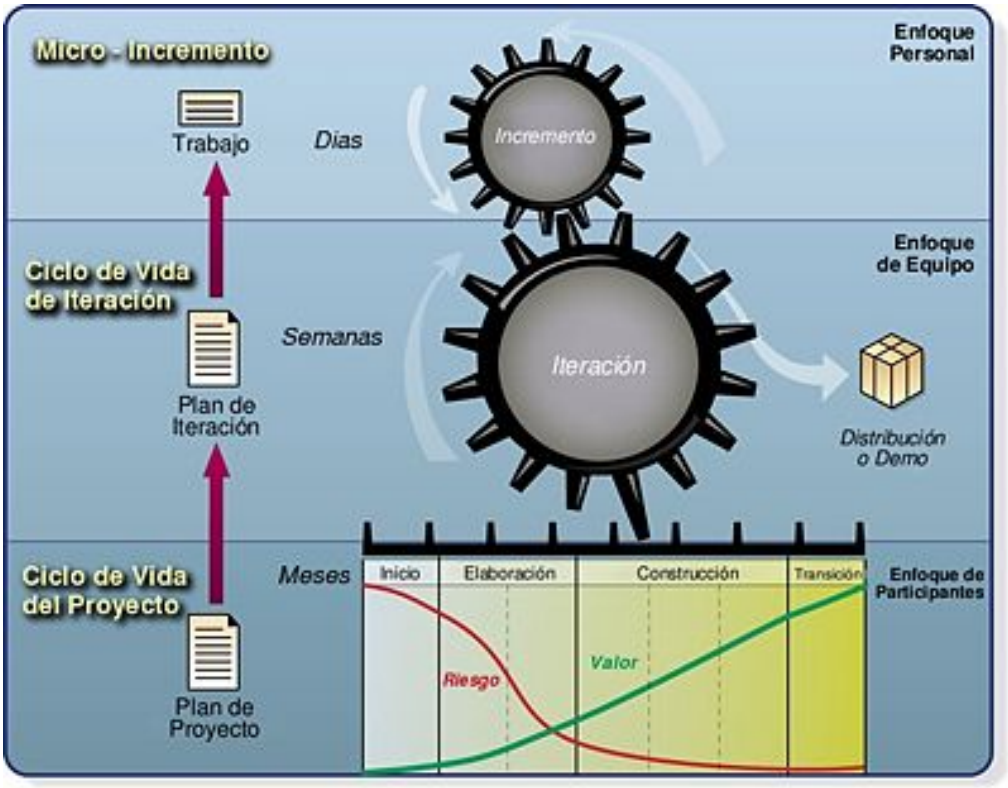


Figura 1.4: Capas de OpenUP

Además de lo anterior, 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. Es extensible, de manera que se pueden añadir artefactos, actividades u otro componente a la metodología, según lo requiera el sistema que se desarrolla. 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.

1.4.2. Lenguaje de modelado

Para dar solución al problema planteado, usaremos como lenguaje de modelado UML [33]. El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software [34]. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación y principios gene-

rales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

1.4.3. Herramienta CASE

Las herramientas CASE[5] 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 [35], Rational Rose [36], Umbrello [37], Visual Paradigm for UML, ArgoUML [38] y otras.

Visual Paradigm es una herramienta CASE que soporta la última versión del Lenguaje de Modelado Unificado (UML) y 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.

1.4.4. IDE y lenguaje de programación

Lenguaje de Programación

El lenguaje de programación seleccionado es Java. La razón principal de esta selección es que es independiente de plataforma y arquitectura de red. Por eso una aplicación escrita en Java, puede ejecutarse en cualquier sistema. Esta portabilidad es extremadamente importante para cualquier sistema distribuido, ya que se espera que los clientes puedan correr en múltiples sitios en diferentes plataformas.

En los primeros días de Java, gran parte de sus críticas se originaban de su pobre rendimiento respecto a lenguajes nativos como C y Fortran. Mucho ha cambiado desde entonces. Actualmente se han producido enormes mejoras en el rendimiento de la Máquina Virtual de Java (JVM[6]), principalmente atribuida a la introducción del compilador just-in-time [39] y la tecnología hotspot [40]. Estas mejoras han dado lugar a que la ejecución de la JVM, sea comparable a la de otros lenguajes nativos [41]. Otro de los aspectos más importante de Java es el modelo de seguridad [42]. Este simplifica enormemente la implementación de una estricta política de seguridad para una aplicación Java [43], haciendo posible que las

[5] Computer Aided Software Engineering
[6] En Inglés, Java Virtual Machine

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” [44].

Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado(IDE[7]), es un programa compuesto por un conjunto de herramientas para un programador. Es decir, es un entorno de programación que ha sido empaquetado como un programa de aplicación y consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de usuario.

El IDE Netbeans es “...un Entorno Integrado de Desarrollo gratuito, de código abierto para desarrolladores de software. Puede obtener todas las herramientas que necesite para crear aplicaciones profesionales para el escritorio, la empresa, la web (...) con el lenguaje Java (...) NetBeans IDE es fácil de instalar y de uso instantáneo y se ejecuta en varias plataformas...” [45]

Netbeans soporta la adición de módulos con múltiples funcionalidades, lo que permite la extensión del IDE para colmar las necesidades del desarrollador.

Este IDE fue seleccionado principalmente por su potente editor de código, la interfaz amigable que presenta, la facilidad de su creador de interfaz gráfica de usuario para web y para escritorio y la presencia de un módulo para desarrollar Portlets para Gridsphere.

En resumen, Netbeans brinda un poderoso IDE basado en Java, lo suficientemente potente para desarrollar aplicaciones empresariales y ahorrar tiempo de desarrollo permitiendo que el desarrollador se centre en el núcleo.

1.4.5. Herramientas y tecnologías para el desarrollo

Portal de Servicios basado en GRIDSPHERE

Un portal de Internet es un sitio web cuyo objetivo es ofrecer al usuario, de forma fácil e integrada, el acceso a una serie de recursos y de servicios, entre los que suelen encontrarse buscadores, foros, documentos, aplicaciones, compra electrónica, etc. Principalmente están dirigidos a resolver necesidades específicas de un grupo de personas o de acceso a la información y servicios de una institución pública o privada.
Existen tres modalidades de portales:

1. Portales horizontales, también llamados portales masivos o de propósito general, se dirigen a una audiencia amplia, tratando de llegar a toda la gente con muchas cosas. Como ejemplo de portales

[7] En inglés, Integrated Development Environment

de esta categoría están Terra, AOL, AltaVista, UOL, Lycos, Yahoo, MSN.

2. Portales verticales, se dirigen a usuarios para ofrecer contenido dentro de un tema específico como puede ser un portal de música, empleo, inmobiliario, un portal de finanzas personales, arte o de deportes.

3. Portales diagonales: se trata de una mezcla entre el portal horizontal y el vertical. Se trataría de portales que utilizan redes sociales o aplicaciones generalistas como Facebook, Linkd, Flickr o YouTube complementados con contenidos y/o utilidades dirigidas a un público muy concreto.

Los portales normalmente tienen programación que requiere muchos recursos computacionales y por su alto trafico generalmente se hospedan en servidores de Internet dedicados.

En el Polo de Bioinformática de la Universidad de las Ciencias Informáticas fue desarrollado un Portal de Servicios basado en Gridsphere. Con el portal se pretende lograr la integración de todos los proyectos de la Facultad. En el momento de liberarse la primera versión de Tarenal, no existía dicho portal. Como parte de la estrategia del dicho Polo, el front-end de Tarenal será integrado al portal, creando servicios que permitan a los usuarios interactuar con la Plataforma de Tareas Distribuidas v2.0.

Java Web Start

Java Web Start es una tecnología que brinda la posibilidad de ejecutar aplicaciones Java con un solo clic. Los usuarios pueden descargar y usar aplicaciones, sin necesidad de seguir procedimientos complicados de instalación. Una vez accedida a través de Java Web Start, la aplicación se integra con el escritorio del usuario; desde el punto de vista del usuario, funciona como una aplicación nativa. Además Java Web Start se encarga de actualizar automáticamente la versión de la aplicación [46].

Algunas de las ventajas de usar Java Web Start para distribuir aplicaciones son:

- Facilidad de instalación: Los usuarios pueden fácilmente instalar una aplicación haciendo clic en un vínculo de una página Web.

- Independencia de plataforma: Con Java Web Start usted puede colocar una aplicación Java en un servidor Web para descarga a una amplia variedad de plataformas, incluyendo Windows 98, NT, 2000, ME, XP, Linux y Solaris.

- Administración de Java Runtime Environment: Java Web Start soporta versiones varias y simultáneas de la plataforma Java 2 SE. Las aplicaciones pueden solicitar versiones específicas de Java

sin conflictos con las necesidades de otras aplicaciones. Java Web Start descarga e instala la versión correcta de la plataforma Java, si es necesario, basado en las necesidades de la aplicación y el entorno de usuario.

- Integración con el escritorio: Los usuarios pueden acceder a cualquier aplicación Java Web Start, incluyendo aquellas que se basan en la red.
- Actualizaciones: Se puede actualizar una aplicación para todos los usuarios simplemente brindando un archivo JAR actualizado en el servidor Web. En cada ordenador, Java Web Start verifica el servidor para actualizar la aplicación cuando esta se ejecuta.
- Seguridad: Java Web Start toma ventaja de la seguridad inherente a la plataforma Java. Las aplicaciones de usuario se restringen a una caja de arena[8] y no pueden corromper los sistemas. Si usted necesita funcionalidad adicional y firma los archivos JAR que componen la aplicación, los usuarios deciden si confiar en la fuente de la aplicación y entonces permitir que se ejecuten. No puede pasar nada sin el conocimiento y aprobación del usuario.
- Rendimiento: Las aplicaciones ejecutadas con Java Web Start son guardadas en la caché local, para mejorar el rendimiento [46].

Java Network Launching Protocol (JNLP) El protocolo JNLP está definido por un esquema XML y especifica [47] como lanzar las aplicaciones de Java Web Start. Consta de un conjunto de reglas que definen como implementar el mecanismo de lanzamiento. Los archivos JNLP incluyen información como la ubicación de los archivos JAR y el nombre de la clase principal, unido a otros parámetros del programa. Un navegador correctamente configurado pasa los archivos JNLP a la Java Runtime Environment que a su vez descarga la aplicación para el ordenador local y comienza a ejecutarla. JNLP es gratis; los desarrolladores no tienen que pagar licencias para usarlo en las aplicaciones. Cualquier usuario puede usar JNLP simplemente instalando un cliente, comúnmente Java Web Start. La instalación del cliente puede hacerse automáticamente antes de la aplicación Java, la primera vez que esta última se ejecuta [48].

1.5. Conclusiones

La interfaz de usuario determina la facilidad con que un usuario pueda lograr su objetivo con una aplicación. Atendiendo a los tipos de interfaces existentes se desarrollarán interfaces gráficas de usuario para el software.

[8] En inglés, sandbox

El front-end será desarrollada pensando en el usuario final que consumirá el sistema, teniendo en cuenta los factores que determinan el buen diseño de las interfaces de usuario. Se clasificará la información que deseamos mostrarle al usuario y se validará la interfaz realizando pruebas de desarrollador. Se desarrollarán interfaces útiles para el usuario y el software creado será comprendido, usado y atractivo para el usuario.

Para lograr una organización en el proceso de desarrollo se seleccionó OpenUP/Basic como metodología de desarrollo y Visual Paradigm será la herramienta CASE a utilizar. El lenguaje de programación usado será Java con el IDE Netbeans. La nueva versión elimininará los problemas existentes con la GUI de escritorio de la primera versión.

El hecho de que la versión anterior sea desechada, no significa que no puedan ser reutilizadas las interfaces de comunicación con el back-end.

Capítulo 2

Características del sistema

El desarrollo de un software parte de comprender la problemática a la que se quiere dar solución. Es posible lograr lo anterior a través del levantamiento de requisitos y la realización del Modelo de Negocio o Dominio. 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. Además, se presenta el diagrama de casos de usos del sistema, así como una breve descripción de los casos de usos identificados.

2.1. Breve descripción del sistema

El sistema a desarrollar permitirá interactuar con la Plataforma de Tareas Distribuidas, solucionando los problemas existentes. En esta versión, el front-end estará compuesto por dos módulos: interfaz del servidor e interfaz del cliente. La interfaz del servidor permitirá la interacción con la nueva versión del servidor, incluyendo todas las nuevas funcionalidades que este brinda. La interfaz del cliente introduce nuevas características en T-arenal: el cliente puede ser configurado a través de esta interfaz, se puede planificar la ejecución del cliente y las políticas de seguridad del cliente pueden ser gestionadas usando la interfaz del mismo.

2.2. Modelo de dominio

Debido a la relativa simplicidad del entorno donde está enmarcado el sistema y el conocimiento que se posee acerca de su funcionamiento, no es necesario realizar un Modelo de Negocio para comprender la problemática que ha de resolverse, 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. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de componentes de software [49].

A continuación se muestra el modelo conceptual de la problemática a resolver, identificando los objetos fundamentales y sus relaciones.

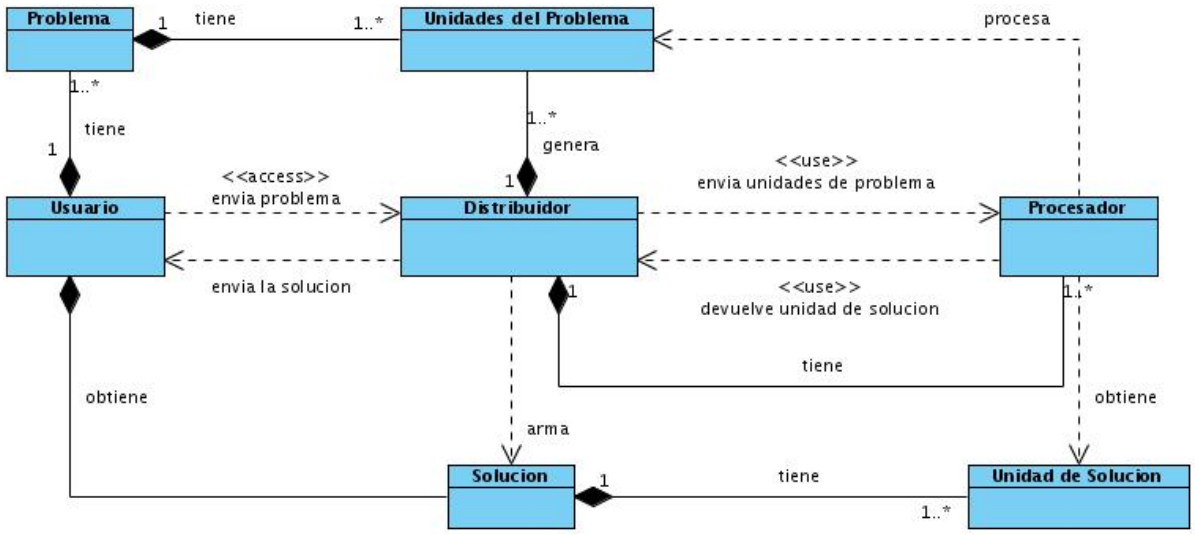


Figura 2.1: Diagrama de clases del Modelo de Dominio

2.2.1. Descripción de los Objetos.

- **Usuario**: El usuario representa al ente que tiene determinado problema y quiere distribuirlo.
- **Problema**: Representa a un problema de gran magnitud que tiene determinado usuario y que se va a resolver de manera distribuida.
- **Distribuidor**: Representa al encargado de aceptar el Problema, dividirlo en Unidades del Problema, distribuirlo por los Procesadores, armar la Solución y devolverla al usuario.
- **Unidades del Problema**: Representa a una porción lógica del problema, cuando se divide un problema en otros más pequeños, esta entidad representa a estas pequeñas partes.
- **Procesador**: Representa la entidad encargada de procesar las unidades del problema, obtener una unidad de solución que corresponde a una unidad del problema y devolverlas al Distribuidor para posteriormente armar la solución.
- **Unidad de Solución**: Representa al ente generado a partir del procesamiento, por parte del Procesador de una Unidad del Problema. Es utilizada para armar la Solución.
- **Solución**: Representa la integración hecha por el Distribuidor de varias Unidades de Solución. Es el producto final que se le devuelve al usuario.

2.3. Especificación de los requisitos del sistema

2.3.1. Requisitos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, y se mantienen invariables sin importar con que propiedades o cualidades se relacionen. Se identificaron los siguientes requisitos funcionales en el Front-end del Servidor

1. Autenticar usuario.
2. Gestionar configuración de acceso al sistema.
    2.1. Guardar configuración de acceso al sistema.
    2.2. Modificar configuración de acceso al sistema.
    2.3. Eliminar configuración de acceso al sistema.
3. Gestionar grupos.
    3.1. Adicionar un grupo.
    3.2. Modificar datos de un grupo.
    3.3. Eliminar un grupo.
4. Visualizar datos de un grupo.
5. Mostrar listado de los grupos.
6. Gestionar usuarios.
    6.1. Adicionar usuario.
    6.2. Modificar datos de un usuario.
    6.3. Eliminar un usuario.
7. Cambiar la contraseña de un usuario.
8. Visualizar datos de un usuario.
9. Mostrar listado de los usuarios.
10. Gestionar rangos IP.
    10.1. Adicionar un rango IP.
    10.2. Permitir o no un rango IP.

10.3. Eliminar un rango IP.

11. Visualizar datos de un rango IP.
12. Mostrar listado de los rangos IP.
13. Gestionar Clientes.
    13.1. Permitir o no un cliente.
    13.2. Eliminar un cliente.
14. Mostrar listado de los clientes.
15. Mostrar listado de los clientes autorizados a interactuar con el sistema.
16. Mostrar listado de los clientes no autorizados a interactuar con el sistema.
17. Mostrar listado de los clientes reportados en un tiempo determinado.
18. Mostrar listado de los clientes no reportados en un tiempo determinado.
19. Gestionar Actualizaciones de Clientes.
    19.1. Adicionar una actualización.
    19.2. Eliminar una actualización.
20. Mostrar listado de las actualizaciones de clientes.
21. Gestionar Problemas.
    21.1. Adicionar un problema.
    21.2. Eliminar un problema.
22. Mostrar listado de los problemas a los que un usuario puede acceder.
23. Mostrar listado de los problemas administrados por un usuario.
24. Gestionar acceso a problemas.
    24.1. Asignar un problema a un usuario.
    24.2. Quitar a un usuario el acceso a un problema.
25. Ejecutar un problema.
26. Monitorear ejecuciones.

26.1. Mostrar el estado de una ejecución.

26.2. Mostrar los errores de una ejecución.

27. Gestionar funcionamiento de las ejecuciones.

27.1. Establecer el estado de pausa a una ejecución.

27.2. Quitar del estado de pausa a una ejecución.

28. Detener una ejecución.

29. Mostrar listado de las ejecuciones.

30. Mostrar listado de las ejecuciones que pertenecen a un usuario.

31. Descargar una solución.

32. Eliminar una solución.

33. Mostrar listado de las soluciones.

34. Mostrar listado de las soluciones que pertenecen a un usuario determinado .

35. Mostrar información del servidor.

35.1. Mostrar información del servidor central.

35.2. Mostrar información del servidor de peticiones.

36. Mostrar listado de los servidores de petición.

37. Permitir o no un servidor de peticiones.

38. Mostrar gráfica que represente los clientes por rangos IP.

39. Mostrar gráfica que represente la cantidad de clientes por sistemas operativos.

40. Mostrar gráfica que represente la cantidad de clientes según la capacidad de procesamiento.

41. Mostrar gráfica que represente la cantidad de clientes según la memoria disponible.

Se identificaron los siguientes requisitos funcionales en el Front-end del Cliente

1. Gestionar ejecuciones del cliente.

1.1. Ejecutar siempre el cliente.

1.2. Ejecutar el cliente basado en su configuración.

1.3. Suspender el cliente.

2. Gestionar opciones de la interfaz.

   2.1. Especificar el lenguaje.

   2.2. Seleccionar el tema de la interfaz.

3. Gestionar configuración inicial.

   3.1. Mostrar configuración de conexión al servidor de peticiones.

   3.2. Editar configuración de conexión al servidor de peticiones.

   3.3. Probar conexión al servidor de peticiones.

   3.4. Mostrar configuración de conexión al servidor de ficheros .

   3.5. Editar configuración de conexión al servidor de ficheros.

   3.6. Probar conexión al servidor de ficheros.

   3.7. Mostrar configuración del tiempo de espera del cliente.

   3.8. Editar configuración del tiempo de espera del cliente.

4. Gestionar uso del procesador.

   4.1. Mostrar configuración del uso del procesador .

   4.2. Editar configuración del uso del procesador.

5. Gestionar acceso al disco duro.

   5.1. Definir la capacidad máxima del directorio de ficheros.

   5.2. Mostrar información sobre el uso del disco duro.

6. Gestionar Permisos.

   6.1. Permitir acceso a determinado directorio.

   6.2. No permitir acceso a determinado directorio.

   6.3. Permitir ejecutar procesos.

   6.4. No permitir ejecutar procesos.

   6.5. Dar todos los permisos.

   6.6. Mostrar los permisos existentes.

7. Gestionar visualización de mensajes.

7.1. Mostrar mensajes de estado de la aplicación.

7.2. Mostrar mensajes de transferencia de datos.

7.3. Mostrar mensajes de errores.

2.3.2. Requisitos no funcionales

Los requerimientos 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 plataforma serán investigadores de los centros del polo científico del oeste de la Habana, y de la Universidad de la Habana.
- La plataforma 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

- La interacción con la plataforma, estará sometida a un proceso de autenticación del usuario donde se especificará el alias y la contraseña. Cada usuario tendrá asignado un rol en el sistema. Cada rol definido tendrá niveles de acceso al Software.
- Tarenal asegurará al 100 % que el problema del usuario será resuelto, independientemente de los problemas ajenos que existan, ya sean fallos de red, electricidad, apagado de máquinas, entre otros.
- Se debe garantizar comunicaciones seguras entre los clientes y el servidor, codificando todo el tráfico de información, con la utilización de algoritmos y protocolos.

Eficiencia

- 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.
- 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 capacidad suficiente para almacenar todos los problemas, soluciones, y ejecuciones que existan en el sistema.

- La red de interconexión de la institución donde se encuentra desplegada la plataforma debe tener como mínimo una velocidad de 100 Mbps.

Soporte

- La plataforma debe propiciar su mejoramiento y la anexión de otras opciones que se le incorporen en el futuro. Estará bien documentada de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.
- La instalación del sistema debe caracterizarse por su facilidad, claridad y sencillez.
- La plataforma podrá ser ejecutada sobre cualquier tipo de sistema operativo, por su característica de ser multiplataforma.

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

2.4.1. 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 ha hacer uso del sistema, teniendo la posibilidad de interactuar con las funcionalidades más básicas de este.
Administrador de Problemas	Representa al usuario que va hacer uso del sistema, teniendo la posibilidad de interactuar con las funcionalidades más básicas de este y gestionar problemas.
Administrador	Representa al usuario que va hacer uso del sistema, teniendo la posibilidad de realizar todas las funcionalidades que brinda.
Administrador de la PC	Representa al usuario que va a administrar la interfaz del cliente

Cuadro 2.1: Actores del sistema y su descripción

2.4.2. Casos de usos del sistema

La forma en que los actores usan el sistema es representada a través de los casos de usos. 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 usos identificados en el presente trabajo son enunciados a continuación:

Orden	Nombre	Prioridad	Breve descripción

Front-end del servidor			
1	Autenticar usuario	Crítico	El usuario inicia el caso de uso cuando decide interactuar con en el sistema. Entonces introduce los datos necesarios, el sistema verifica la validez de los mismos, y finalmente lo acepta o lo rechaza.
2	Gestionar Grupos	Secundario	El caso de uso inicia cuando el administrador decide: adicionar, modificar, eliminar o visualizar grupos en el sistema. Una vez realizada alguna de estas acciones finaliza el caso de uso.
3	Gestionar Usuarios	Secundario	El caso de uso inicia cuando el administrador decide: adicionar, modificar, cambiar la contraseña, eliminar o visualizar usuarios en el sistema. Una vez realizada alguna de estas acciones finaliza el caso de uso.
4	Gestionar Rangos IP	Secundario	El caso de uso inicia cuando el administrador decide: adicionar, permitir, eliminar o visualizar rangos IP en el sistema. Una vez realizada alguna de estas acciones finaliza el caso de uso.
5	Gestionar Clientes	Secundario	El caso de uso comienza cuando el administrador visualiza los clientes existentes, y posteriormente decide autorizarlos o eliminarlos. Después de realizada alguna de estas operaciones, finaliza el caso de uso.
6	Gestionar Actualizaciones de Clientes	Secundario	El caso de uso comienza cuando el administrador desea adicionar o eliminar una actualización para los clientes del sistema. Después de realizada alguna de estas operaciones, finaliza el caso de uso.
7	Gestionar Problemas	Crítico	El caso de uso se inicia cuando el administrador decide: adicionar, eliminar o visualizar problemas en el sistema. Una vez realizada alguna de estas acciones finaliza el caso de uso.
8	Gestionar Acceso a Problemas	Secundario	El caso de uso comienza cuando un administrador desea asignar o quitar a un usuario el acceso a un problema. Después de realizada alguna de estas operaciones, finaliza el caso de uso.
9	Ejecutar Problemas	Crítico	El caso de uso comienza cuando el usuario visualiza los problemas a los cuales accede, y posteriormente decide ejecutarlos. Después de realizada alguna de estas operaciones, finaliza el caso de uso.

10	Monitorear Ejecuciones	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. Después de realizada alguna de estas operaciones, finaliza el caso de uso.
11	Gestionar Funcionamiento de las Ejecuciones	Secundario	El caso de uso comienza cuando el administrador visualiza las ejecuciones existentes en el sistema, y posteriormente decide establecer o quitar el estado de pausa en las ejecuciones. Después de realizada alguna de estas operaciones, finaliza el caso de uso.
12	Detener Ejecuciones	Secundario	El caso de uso comienza cuando el usuario visualiza las ejecuciones a las cuales tiene acceso, y posteriormente decide detenerlas.
13	Descargar Soluciones	Secundario	El caso de uso comienza cuando el usuario visualiza las soluciones a las cuales tiene acceso, y posteriormente decide descargar alguna de ellas.
14	Eliminar Soluciones	Secundario	El caso de uso comienza cuando el usuario visualiza las soluciones a las cuales tiene acceso, y posteriormente decide eliminarlas o descargarlas. Después de realizada alguna de estas operaciones, finaliza el caso de uso.
15	Mostrar Reportes Gráficos	Auxiliar	El caso de uso comienza cuando el usuario selecciona el tipo de reporte gráfico que desea visualizar. El sistema busca los datos necesarios y construye la gráfica a mostrar. Finaliza el caso de uso.
16	Gestionar servidor	Secundario	El caso de uso comienza cuando el administrador desea conocer información de los servidores. Una vez seleccionado el servidor, el sistema muestra los datos y finaliza el caso de uso.
17	Gestionar Configuración	Secundario	El caso de uso comienza cuando el usuario desea modificar la configuración del sistema. Luego de haber cambiado todos los datos finaliza el caso de uso.
Front-end del cliente			
1	Gestionar acceso al disco duro	Secundario	El administrador de la computadora inicia el caso de uso cuando desea cambiar la información sobre el acceso al disco duro. Una vez concluida esta acción finaliza el caso de uso.

2	Gestionar configuración inicial	Secundario	El administrador del ordenador inicia el caso de uso cuando desea ver o modificar la configuración inicial. Después de realizada alguna de estas operaciones, finaliza el caso de uso.
3	Gestionar ejecuciones del cliente	Secundario	El administrador del ordenador inicia el caso de uso cuando desea gestionar las ejecuciones del cliente. El sistema le muestra varias opciones. Una vez seleccionada una de estas operaciones, finaliza el caso de uso.
4	Gestionar opciones de la interfaz	Auxiliar	El administrador del ordenador inicia el caso de uso cuando quiere modificar el lenguaje o el tema de la interfaz. El sistema realizará la operación deseada, finalizando así el caso de uso.
5	Gestionar Permisos	Secundario	El Caso de Uso inicia cuando el administrador del ordenador desea dar o quitar permisos. El sistema realiza la acción seleccionada y finaliza así el caso de uso.
6	Gestionar uso del procesador	Secundario	El administrador del ordenador inicia el caso de uso cuando desea modificar el uso del procesador. Luego el sistema realiza la acción y finaliza así el caso de uso.
7	Gestionar visualización de mensajes	Secundario	El administrador del ordenador inicia el caso de uso cuando desea mostrar los mensajes existentes. El sistema devolverá todos los mensajes existentes terminando así el caso de uso.

2.4.3. 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 usos del sistema del presente trabajo, fue organizado por paquetes para una mejor comprensión y claridad, según los la aplicación del front-end al que pertenecen. Por cada paquete se muestra el diagrama de casos de usos correspondiente.

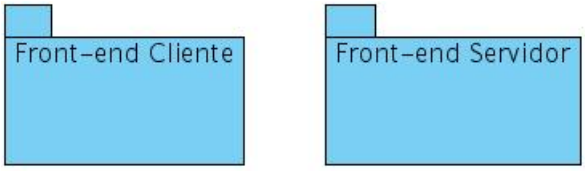


Figura 2.2: Diagrama de casos de uso del sistema

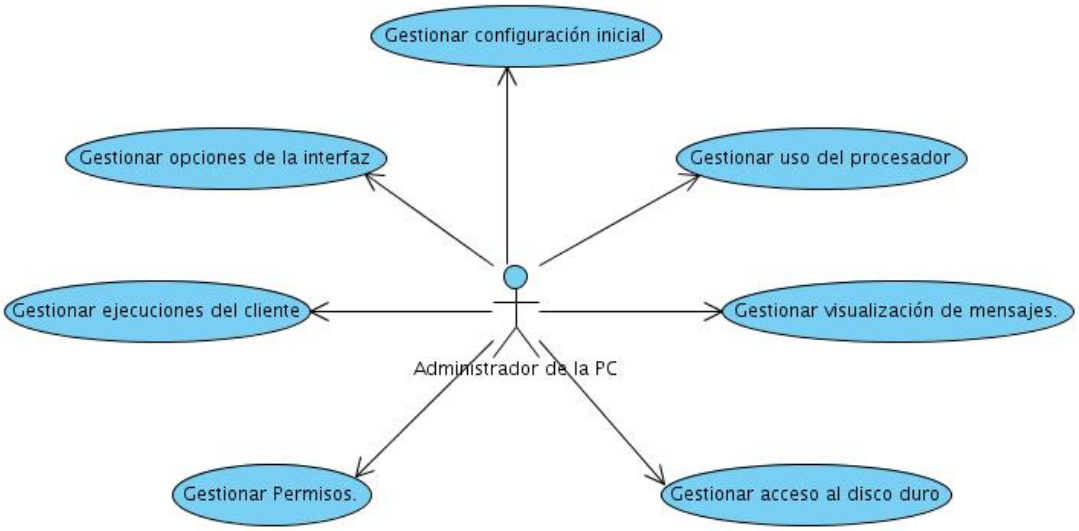


Figura 2.3: Diagrama de Casos de usos del sistema del paquete *Front-end Cliente*

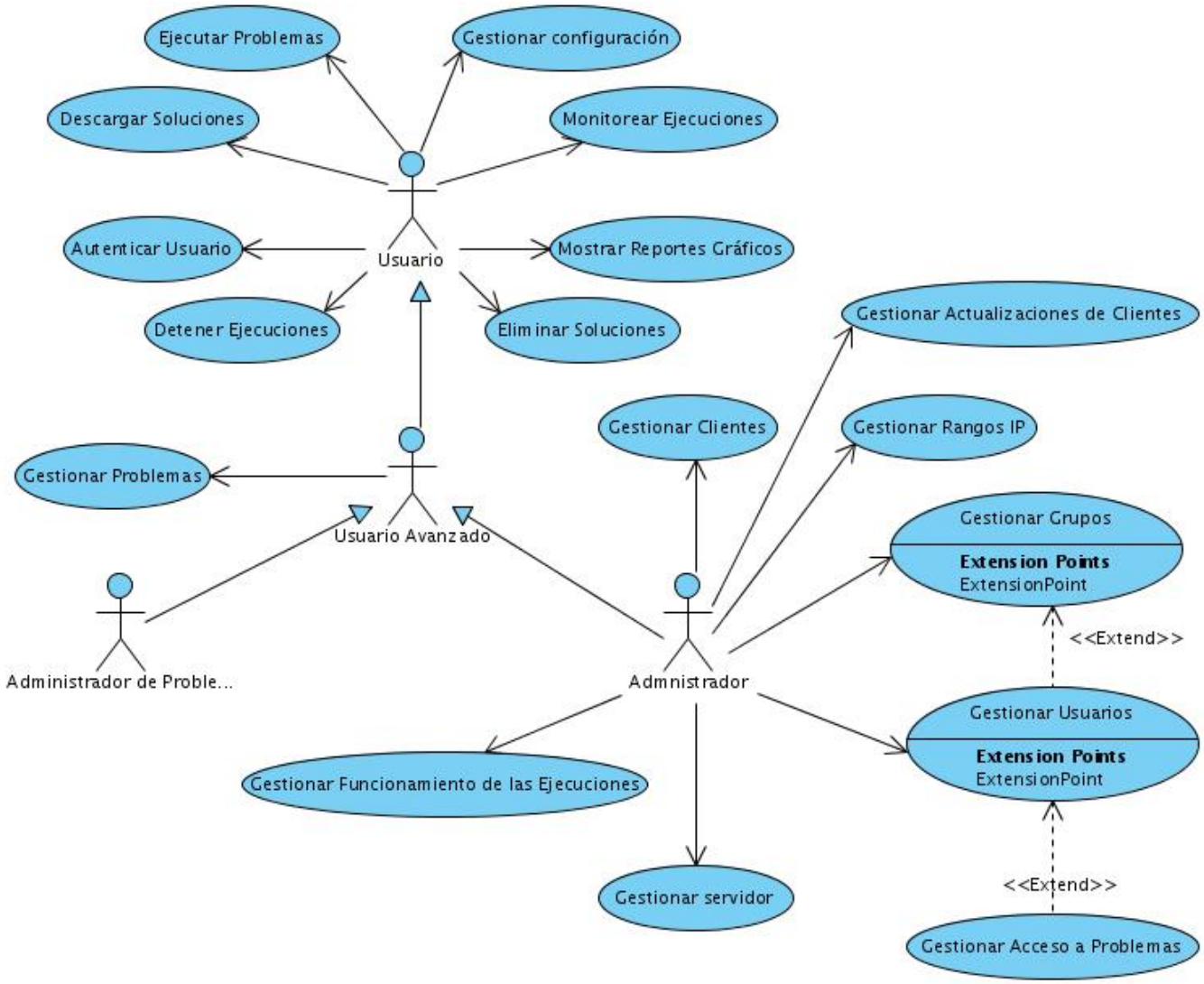


Figura 2.4: Diagrama de Casos de usos del sistema del paquete *Front-end Servidor*

2.4.3.1. Vista de casos de usos arquitectónicamente significativos

De manera resumida podemos decir que los casos de uso 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 nos ayuden a cubrir las funcionalidades significativas. A continuación se muestra la vista de esos casos de uso:

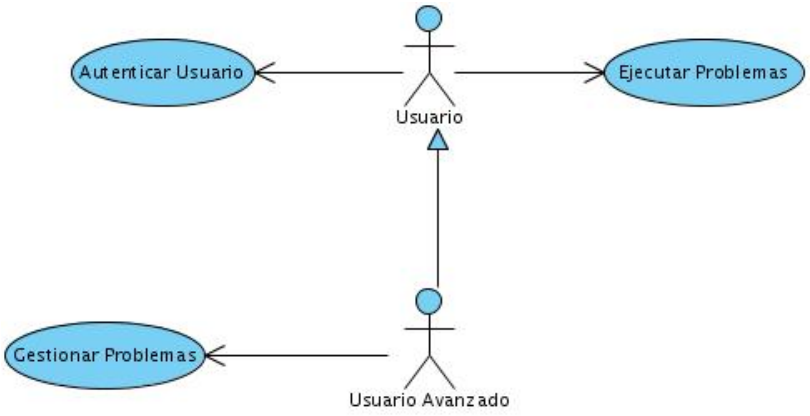


Figura 2.5: Vista de casos de uso arquitectónicamente significativos

2.5. Conclusiones

En este capítulo se identificaron los requerimientos funcionales, los actores y casos de usos del sistema que sirvieron de base para realizar el diseño del producto que se presenta en este trabajo. Se mostraron las relaciones entre actores y casos de uso y los casos de uso críticos para la arquitectura.

Capítulo 3

Diseño del sistema

Durante el diseño se modela el sistema y se le da la forma mediante la arquitectura, de manera tal que soporte todos los requisitos, tanto los funcionales como los no funcionales, creando una entrada apropiada y un punto de partida para las actividades de implementación, dando la posibilidad de obtener un producto con calidad y que satisfaga las necesidades del cliente. En el presente capítulo se dará 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 del sistema.

3.1. Estilo de arquitectura

Un estilo de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico y presenta un esquema genérico y probado de su solución. El estilo de arquitectura seleccionado es dentro de los de llamada y retorno, el Modelo Vista Controlador(MVC). Este estilo o patrón separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos [50]. Para comprender esta selección podemos identificar el contexto en el cual se ubica el problema existente, y la solución propuesta por el patrón:

- Contexto: Software con interfaz hombre-máquina.
- Problema: Es muy frecuente que se solicite cambio a la interfaz. Los cambios a interfaz deberían ser fáciles y efectuados en tiempo de ejecución. El cambio de interfaz no debería de tener consecuencias para el núcleo del código de la aplicación.
- Solución: Separación del sistema en tres partes: Modelo, Vista y Controlador. *Ver Figura 3.1*.

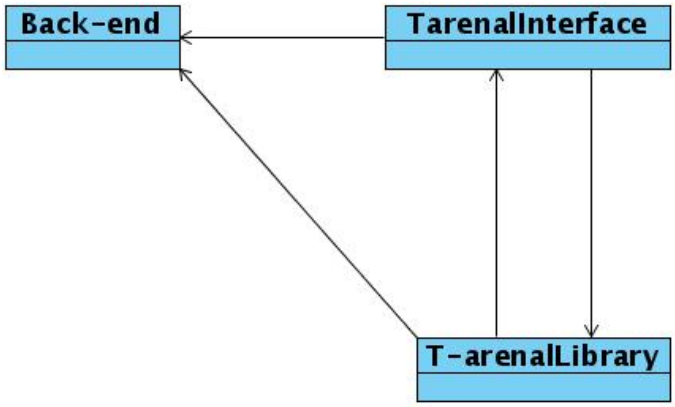


Figura 3.1: Modelo Vista Controlador

- Modelo: Es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos. Esta función la realiza el Back-end.
- Vista: Despliega la información contenida en el modelo. Presenta el modelo en un formato adecuado para interactuar con el sistema y usualmente es un elemento de interfaz de usuario. En este caso es TarenalInterface.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Representado por T-arenalLibrary

El flujo de control que sigue el patrón seleccionado puede verse en la Figura 3.2

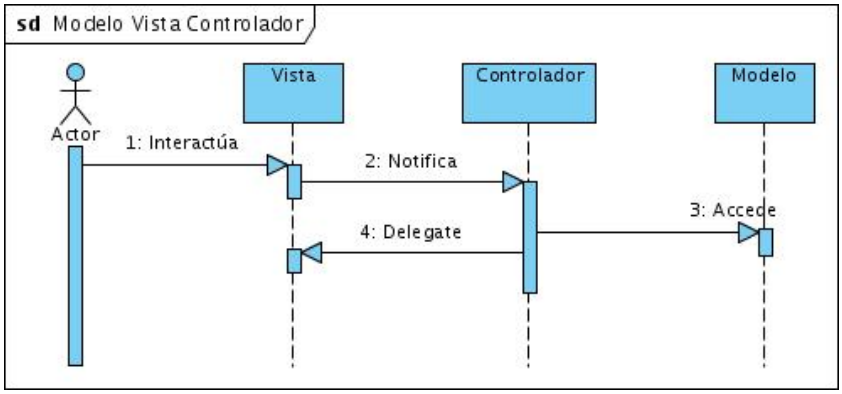


Figura 3.2: Flujo de control del MVC

Las aplicaciones desarrolladas en Java con Swing JFC utilizan una versión simplificada del patrón MVC: modelo-delegado [51]. El modelo vista controlador de Swing tiene estas características:

- Combina la vista y el controlador en un único elemento (UI delegate).

- Es fácil enlazar gráficos y manipulación de eventos.
- La interacción del delgado del modelo no es bidireccional.
- El modelo mantiene la información.
- UI delegate: Dibuja y reacciona a los eventos propagados mediante el componente.

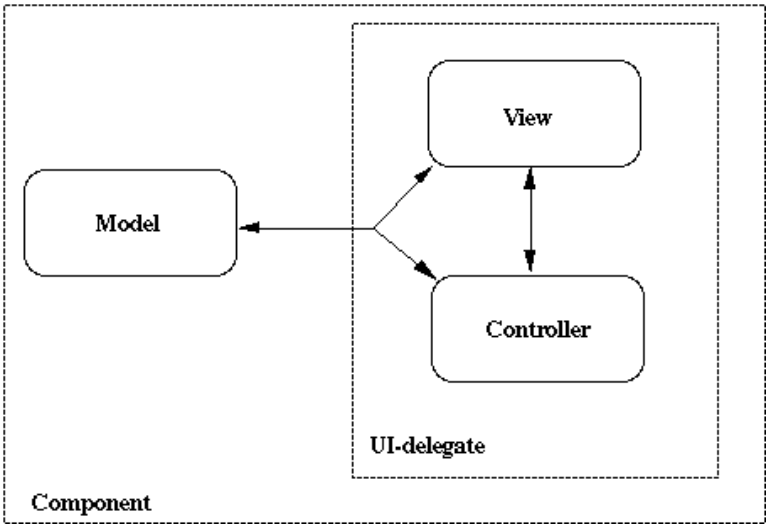


Figura 3.3: MVC en Swing

3.2. Patrones de Diseño

Los patrones de diseño contribuyen a un diseño más sólido del sistema y a lograr un bajo acoplamiento, característica vital que deben de tener los entornos distribuidos. Los patrones utilizados se describen en las subsecciones siguientes.

3.2.1. Patrón Observador

Problema: Se desea observar un objeto sujeto en una aplicación y notificar sus cambios. Los componentes deben notificar que se ha realizado una acción sobre ellos.

Solución: Subscribir uno o más observadores (Listeners) a uno o más eventos del objeto sujeto. Cuando el evento es ejecutado los observadores son notificados. Cada observador implementa su propia función según su meta.

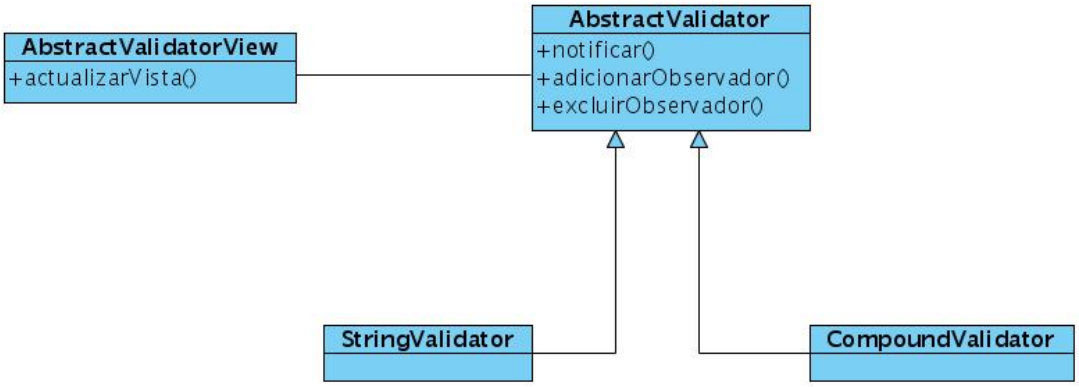


Figura 3.4: Patrón Observador

3.2.2. Patrón Singleton (Instancia única)

Problema: No deben existir varias instancias de la clase TarenalService. No tiene sentido que existan varias instancias de esta clase pues solo tiene las funcionalidades del Portlet y con un único objeto se puede acceder a todas esas acciones. Lo mismo ocurre en el caso de las clases Util y ImageUtil, estas solamente brindan funcionalidades y no tiene sentido instanciarlas para acceder a estos métodos.

Solución: Utilizar el patrón Singleton para restringir la creación de objetos pertenecientes a esa clase.

Figura 3.5: Patrón Singleton

3.2.3. Patrón Façade (Fachada)

Problema: Un cliente (front-end) necesita acceder a parte de la funcionalidad de un sistema más complejo(back-end). Se necesita acceder a funcionalidades brindadas remotamente por el back-end mediante RMI o JMX.

Solución: Definir una interfaz que permita acceder solamente a esa funcionalidad. Esta interfaz serían las librerías que actúan como fachada al Back-end, creando un intermediario más legible.

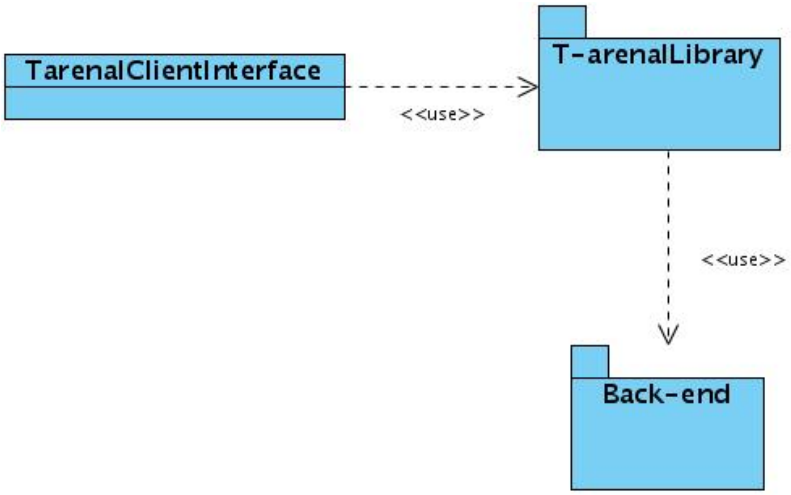


Figura 3.6: Patrón Fachada

3.2.4. Patrón Composite

Problema: Varios componentes deben ser actualizados recursivamente. Muchos componentes a pesar de ser diferentes deben implementar una interfaz común que posibilite su manipulación apropiada. Dentro del TabbedPane de la ventana principal se adicionan pestañas con contenido obtenido de fuentes de datos. Estos datos deben ser actualizados a pesar están mostrados en componentes variados.

Solución: Implementar el patrón Composite en esos componentes.

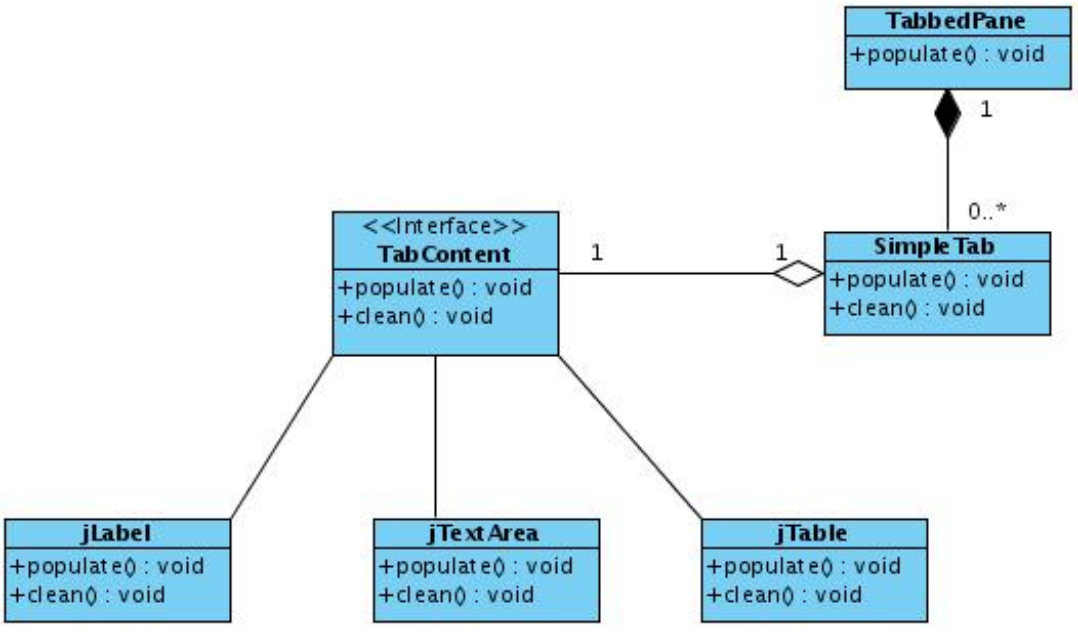


Figura 3.7: Patrón Composite

3.3. Vista lógica

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

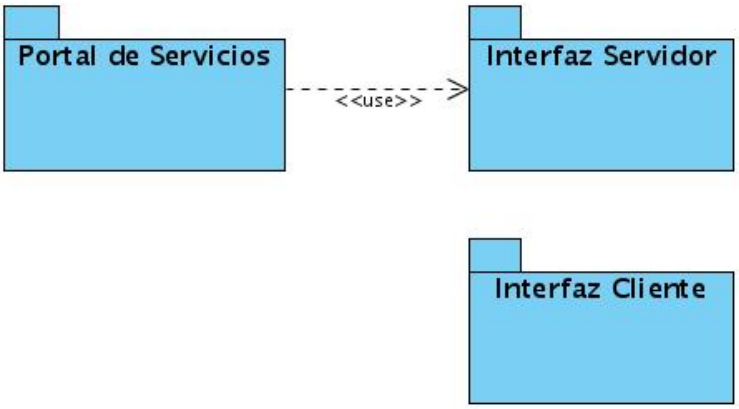


Figura 3.8: Vista de los subsistemas de diseño

3.3.1. Diagramas de clases de diseño

Un diagrama de clases de diseño es un diagrama de estructura estática que describe gráficamente las especificaciones para las clases e interfaces de una aplicación, y además contiene información como clases, asociaciones y atributos, interfaces con sus operaciones y constantes, métodos o funciones, navegabilidad, dependencias y multiplicidad en algunas de sus relaciones.

Los diagramas de clases del diseño permiten que se describa detalladamente y de forma gráfica las especificaciones de las clases de software.

Las clases del diseño de los casos de uso principales son mostrados a continuación:

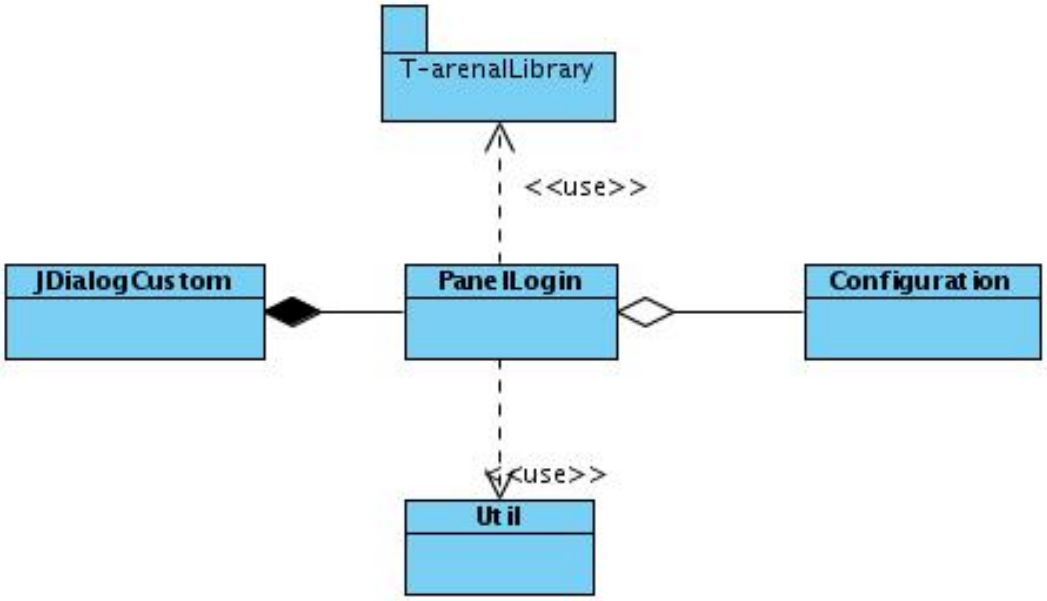


Figura 3.9: Diagrama de clases del diseño CU Autenticar usuario

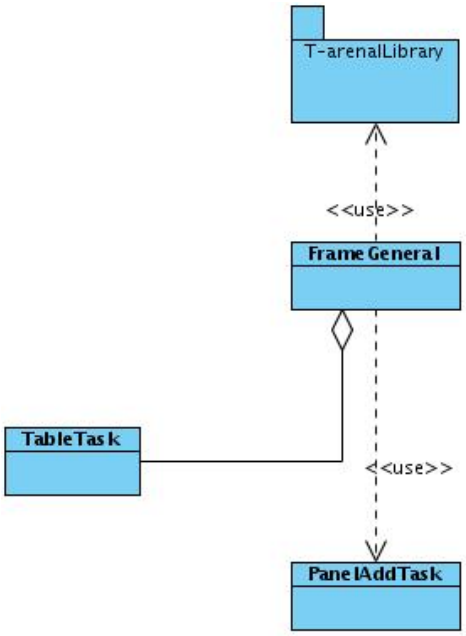


Figura 3.10: Diagrama de clases del diseño CU Gestionar Problemas

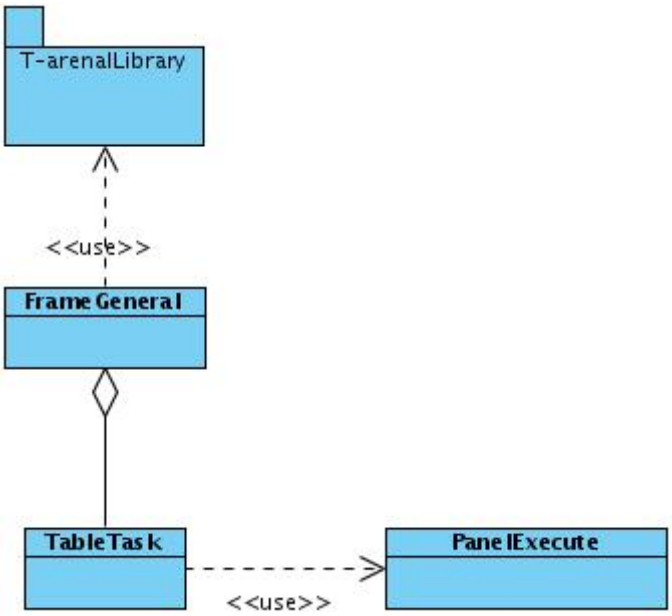


Figura 3.11: Diagrama de clases del diseño CU Ejecutar Problemas

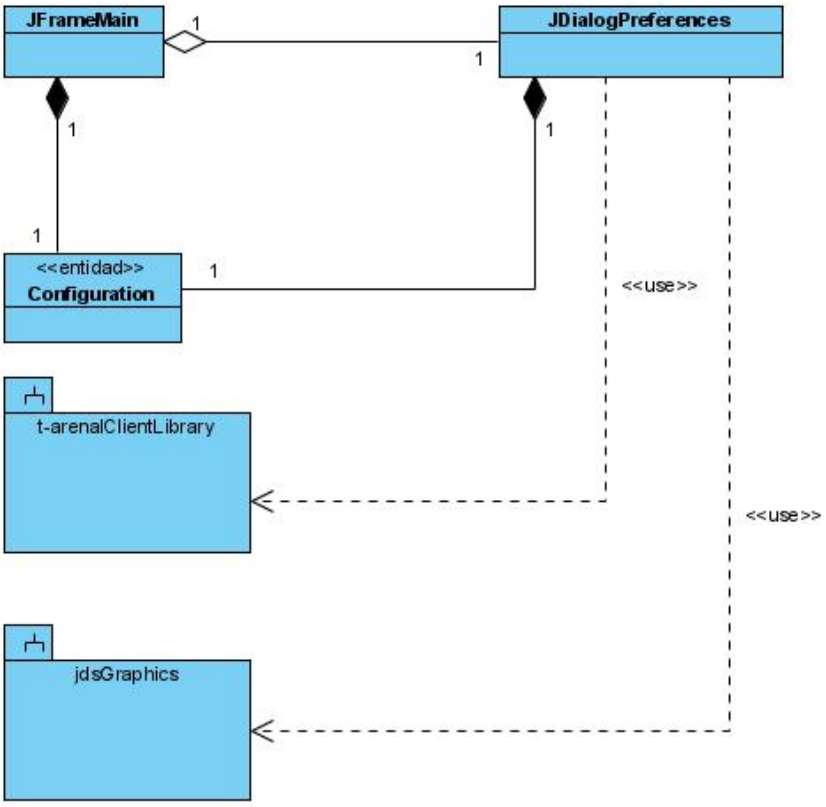


Figura 3.12: Diagrama de clases del diseño CU Gestionar Acceso al Disco Duro.

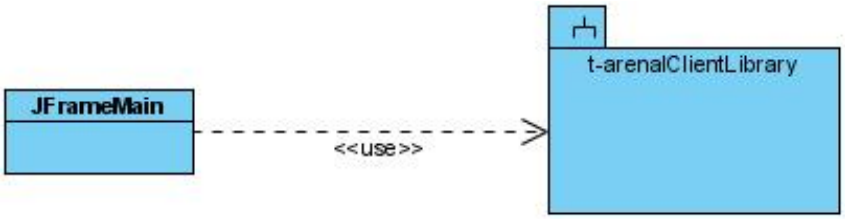


Figura 3.13: Diagrama de clases del diseño CU Gestionar Ejecuciones del Cliente

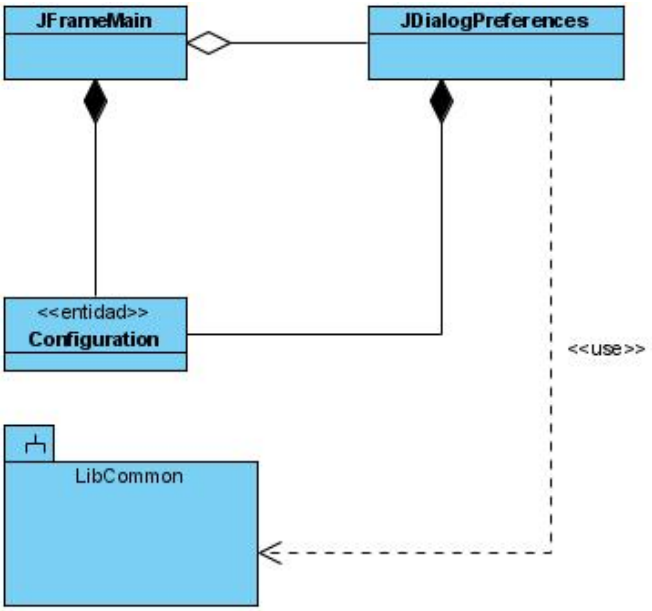


Figura 3.14: Diagrama de clases del diseño CU Gestionar configuración inicial

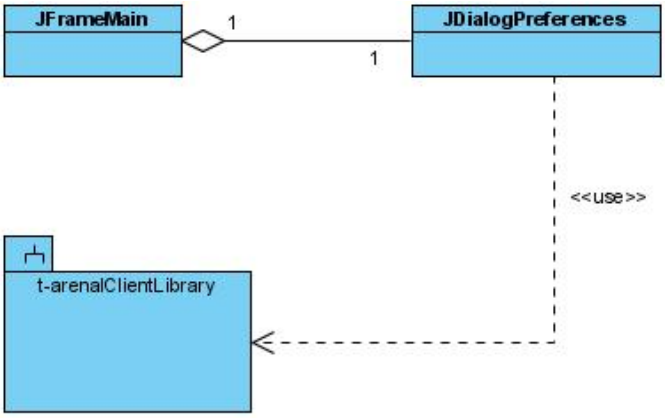


Figura 3.15: Diagrama de clases del diseño CU Gestionar Permisos

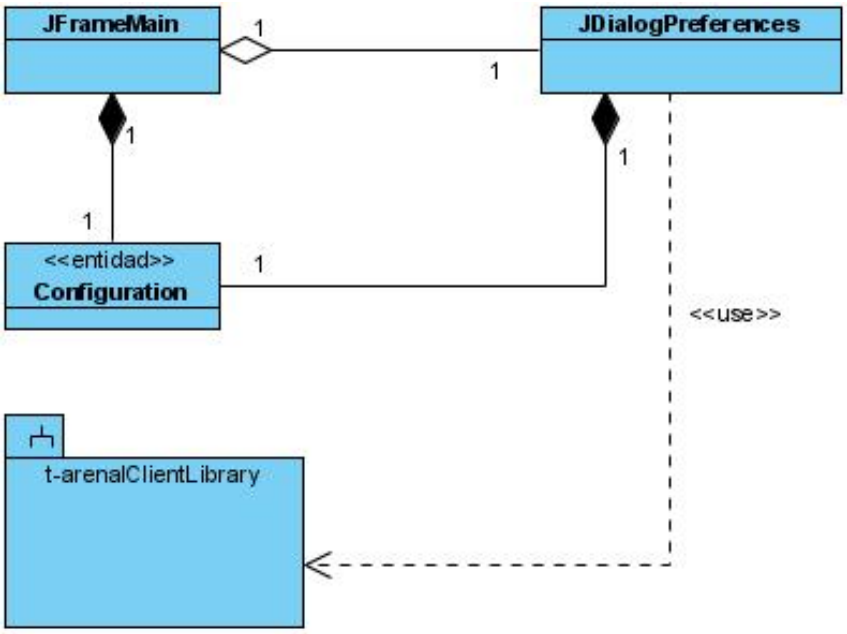


Figura 3.16: Diagrama de clases del diseño CU Gestionar uso del procesador

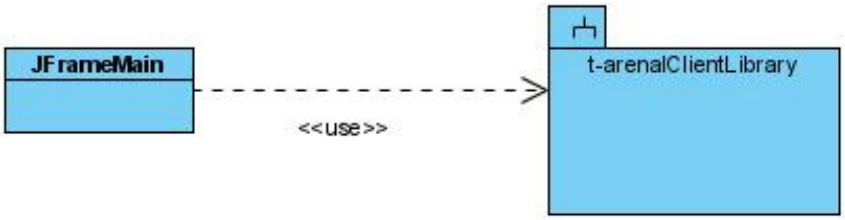


Figura 3.17: Diagrama de clases del diseño CU Gestionar visualización de mensajes

3.3.2. Diagramas de Interacción

Los diagramas de interacción describen secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Son utilizados para el modelado de los aspectos dinámicos de un sistema, proporcionan una vista integral de su comportamiento. Esto conlleva modelar instancias concretas, componentes y nodos junto con los mensajes enviados entre ellos que representan su interacción.

Los diagramas de interacción tienen dos formas de manifestarse:

- Diagramas de secuencia.
- Diagramas de colaboración.

Un diagrama de secuencia destaca la ordenación temporal de los mensajes, ilustra los objetos que se encuentran en un escenario, y la secuencia de mensajes intercambiados entre ellos para llevar a cabo la

funcionalidad descrita; el diagrama de colaboración a su vez destaca la organización estructural de los objetos que envían y reciben mensajes.

Se muestran los principales diagramas de interacción

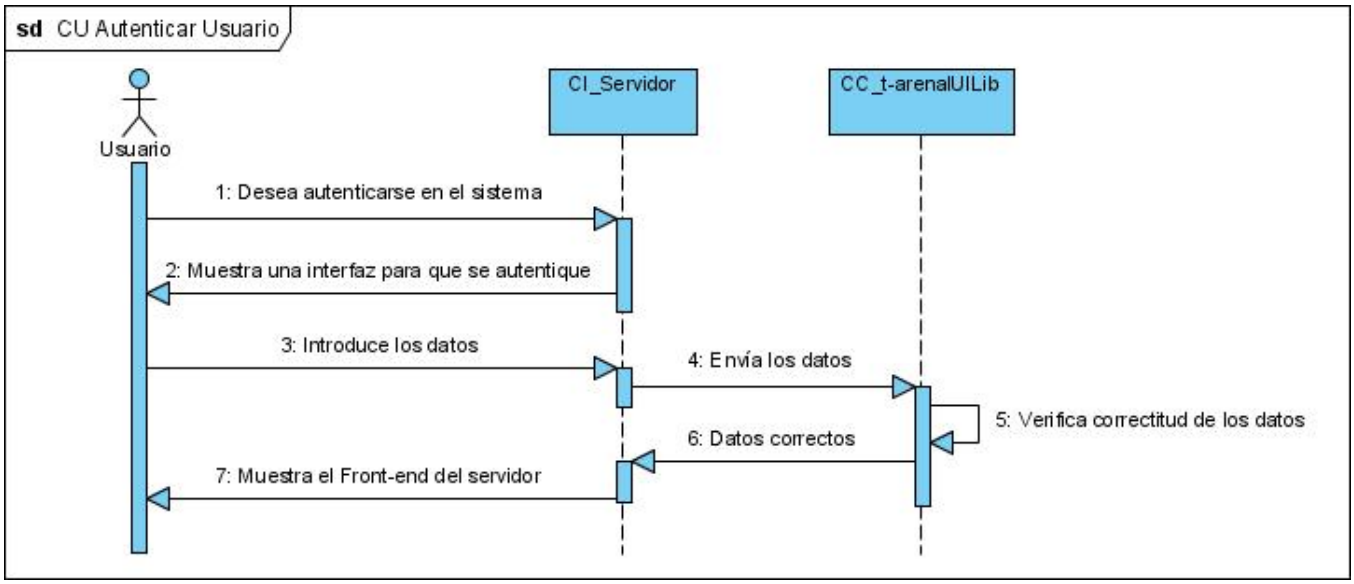


Figura 3.18: Diagrama de secuencia CU Autenticar usuario

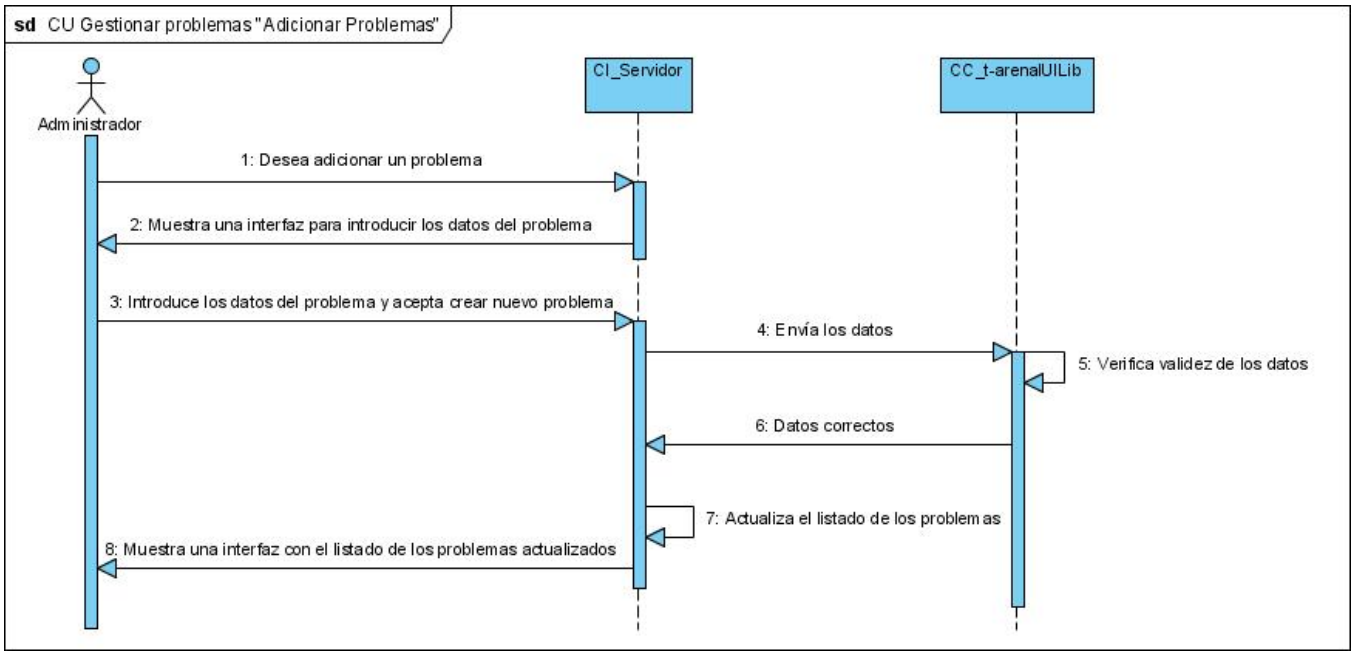


Figura 3.19: Diagrama de secuencia CU Gestionar problemas

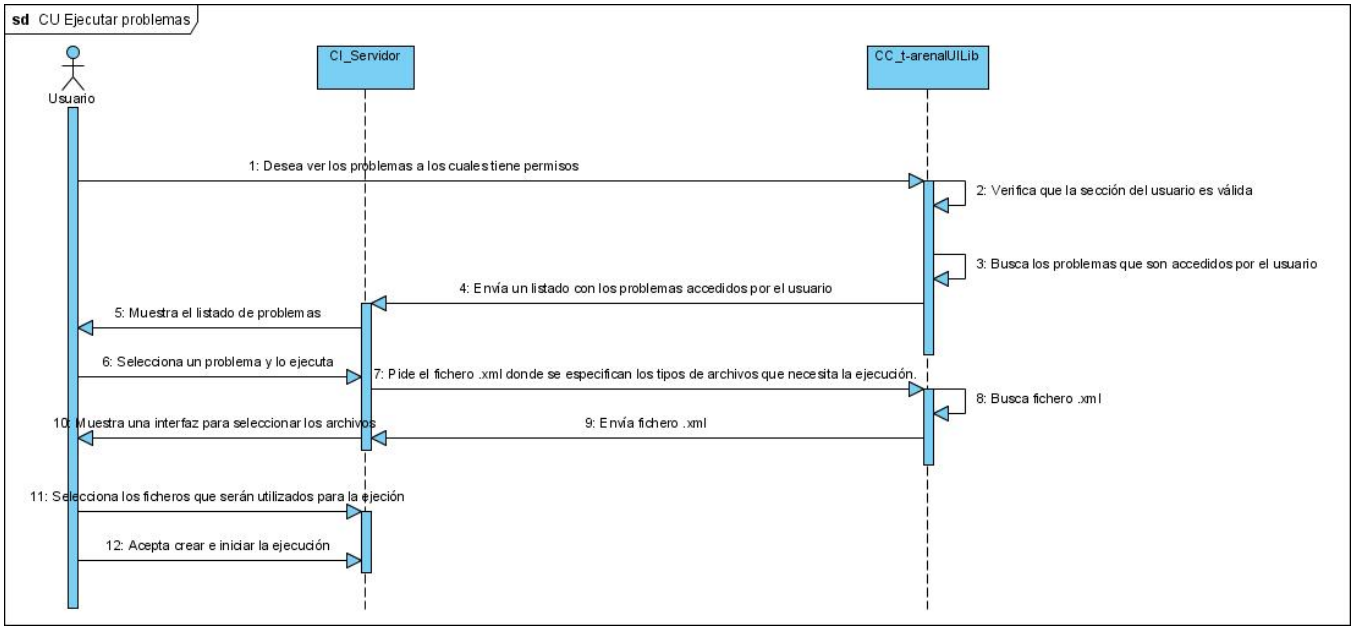


Figura 3.20: Diagrama de secuencia CU Ejecutar problemas

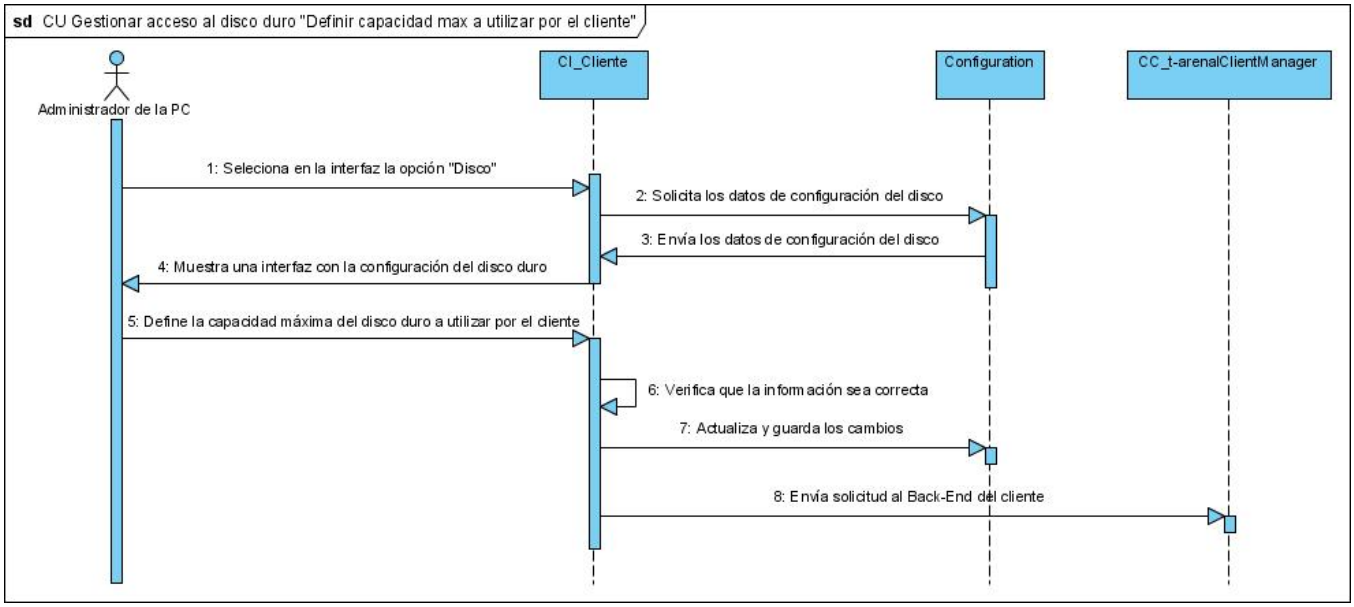


Figura 3.21: Diagrama de secuencia CU Gestionar acceso al disco duro

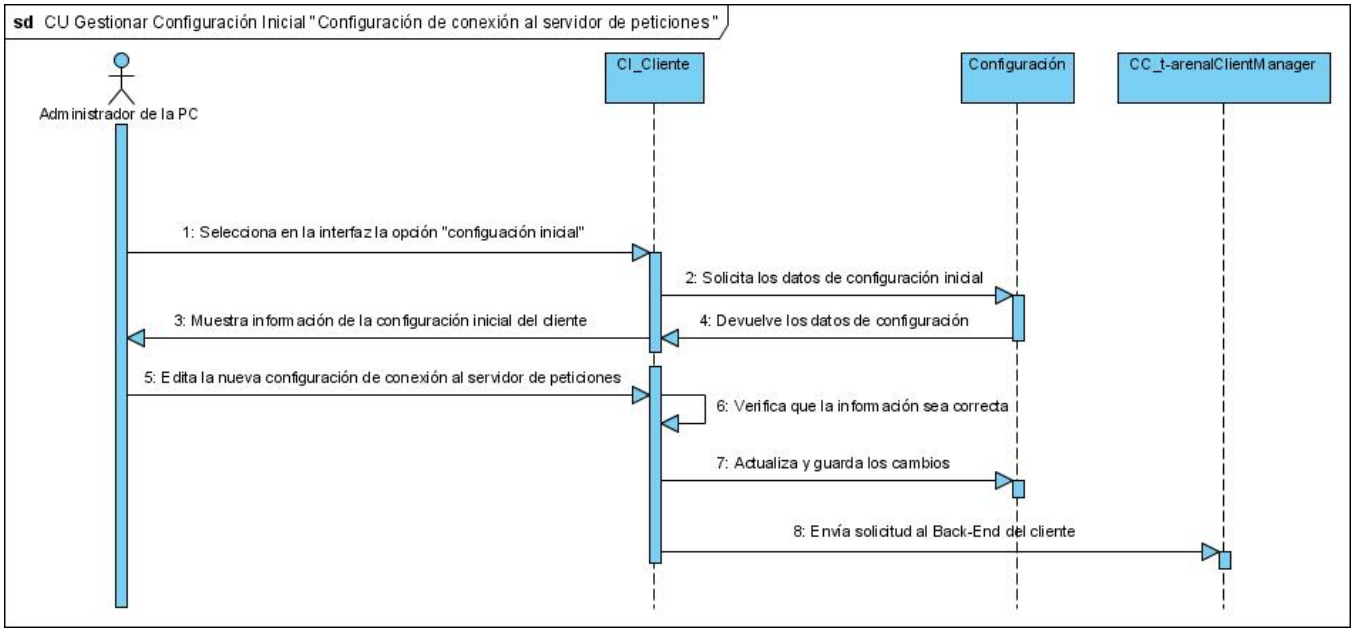


Figura 3.22: Diagrama de secuencia CU Gestionar configuración inicial

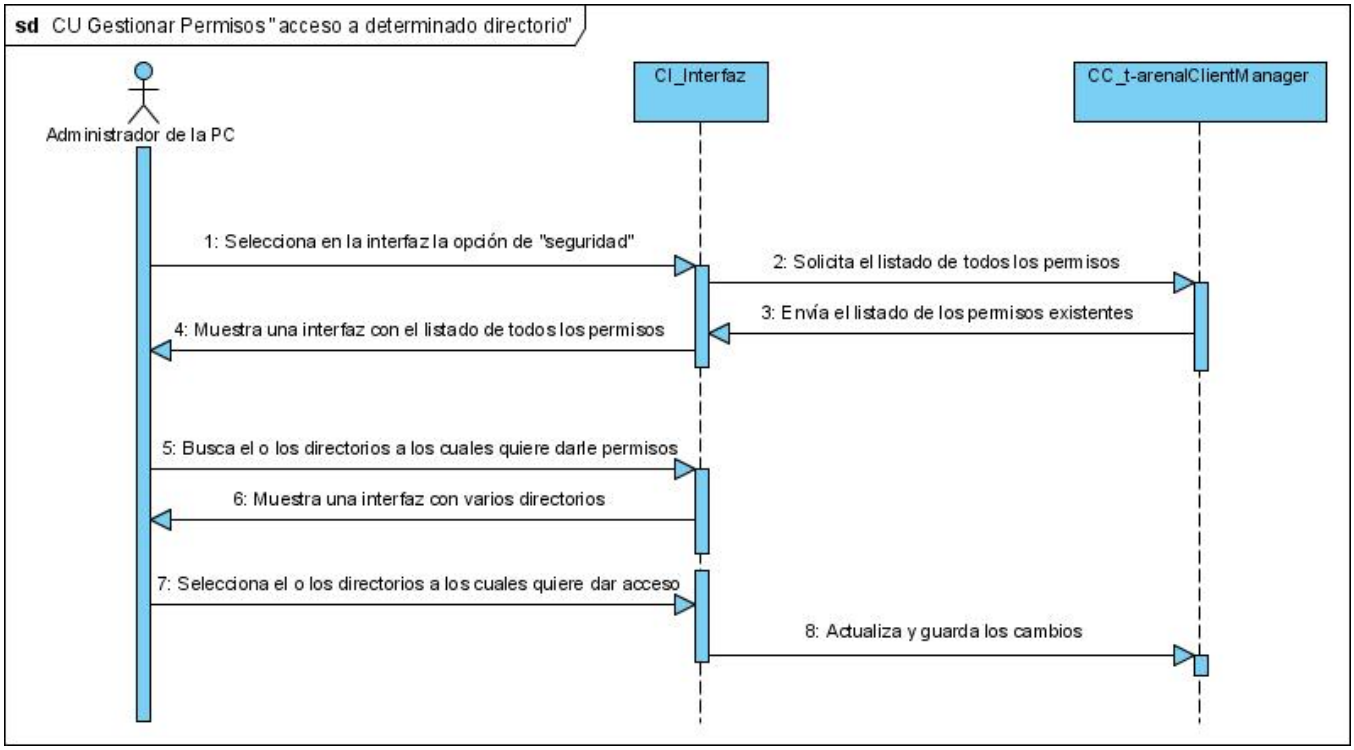


Figura 3.23: Diagrama de secuencia CU Gestionar permisos

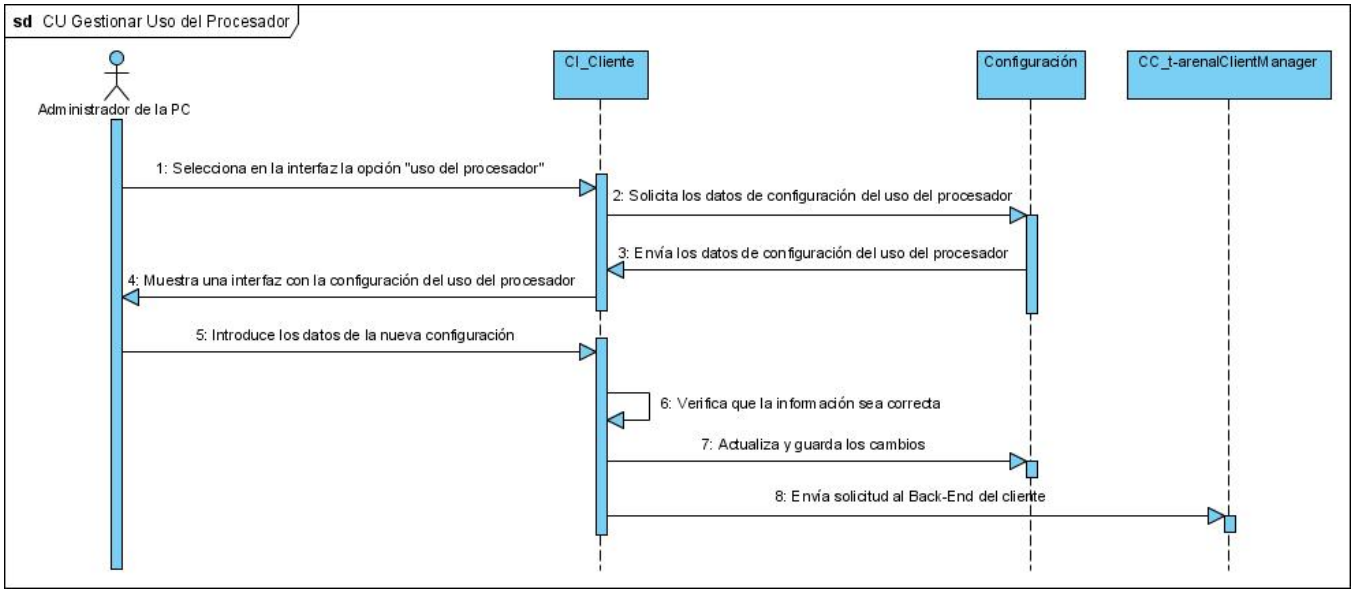


Figura 3.24: Diagrama de secuencia CU Gestionar uso del procesador

3.4. Vista de despliegue

Esta vista constituye la visión física del sistema, representa un grafo de nodos unidos por conexiones de comunicación.

3.4.1. Diagrama de despliegue

El diagrama de despliegue describe la arquitectura física del sistema durante la ejecución, en términos de procesadores, dispositivos y componentes de software. Describen la topología del sistema: la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos.

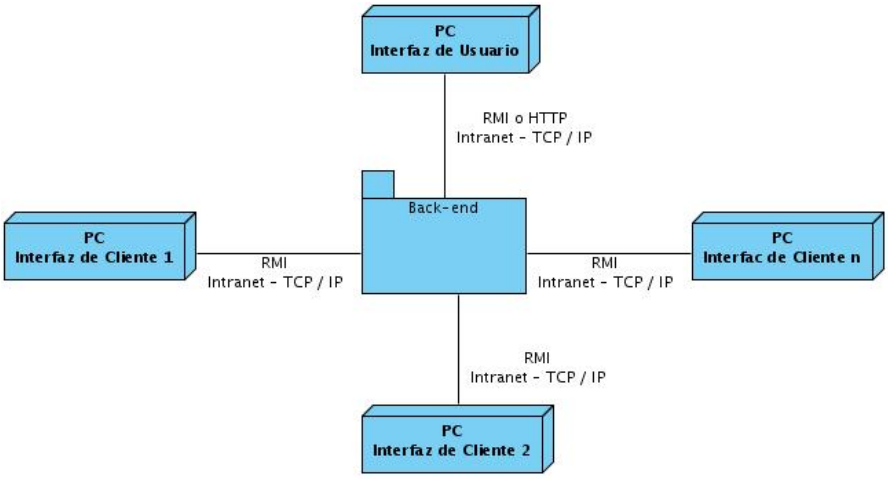


Figura 3.25: Diagrama de despliegue del sistema

Descripción de los nodos

- Nodo Back-end: es el nodo se procesan las entradas recibidas por el Front-end.
- Nodo Interfaz de Usuario: es el nodo donde reside la interfaz del Servidor.
- Nodo Interfaz de Cliente: es el nodo donde reside la interfaz del Cliente.

Descripción de la comunicación

- RMI o HTTP: esta relación define una comunicación entre dos nodos cualesquiera que estos sea, que puede ser a través del protocolo RMI o HTTP. En el caso de la comunicación entre el Cliente del Back-end y la Interfaz Cliente, esta se hace solamente por RMI.

3.5. Conclusiones

Como resultado de este capítulo se obtuvo la arquitectura del sistema sustentada en el modelo vista controlador. Se elaboró el diseño del sistema desde la estructura de paquetes o subsistemas de diseño hasta los diagramas de clases del diseño, mostrando solamente los más significativos. También se mostraron los diagramas de interacción de los principales escenarios del sistema. Además se mostró el diagrama de despliegue del sistema.

Capítulo 4

Implementación y pruebas del Sistema

En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Se ilustrarán los principales resultados obtenidos. Además, los casos de prueba para los principales casos de uso.

4.1. Diagrama de Componentes

El diagrama de implementación describe cómo los elementos del modelo de 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.

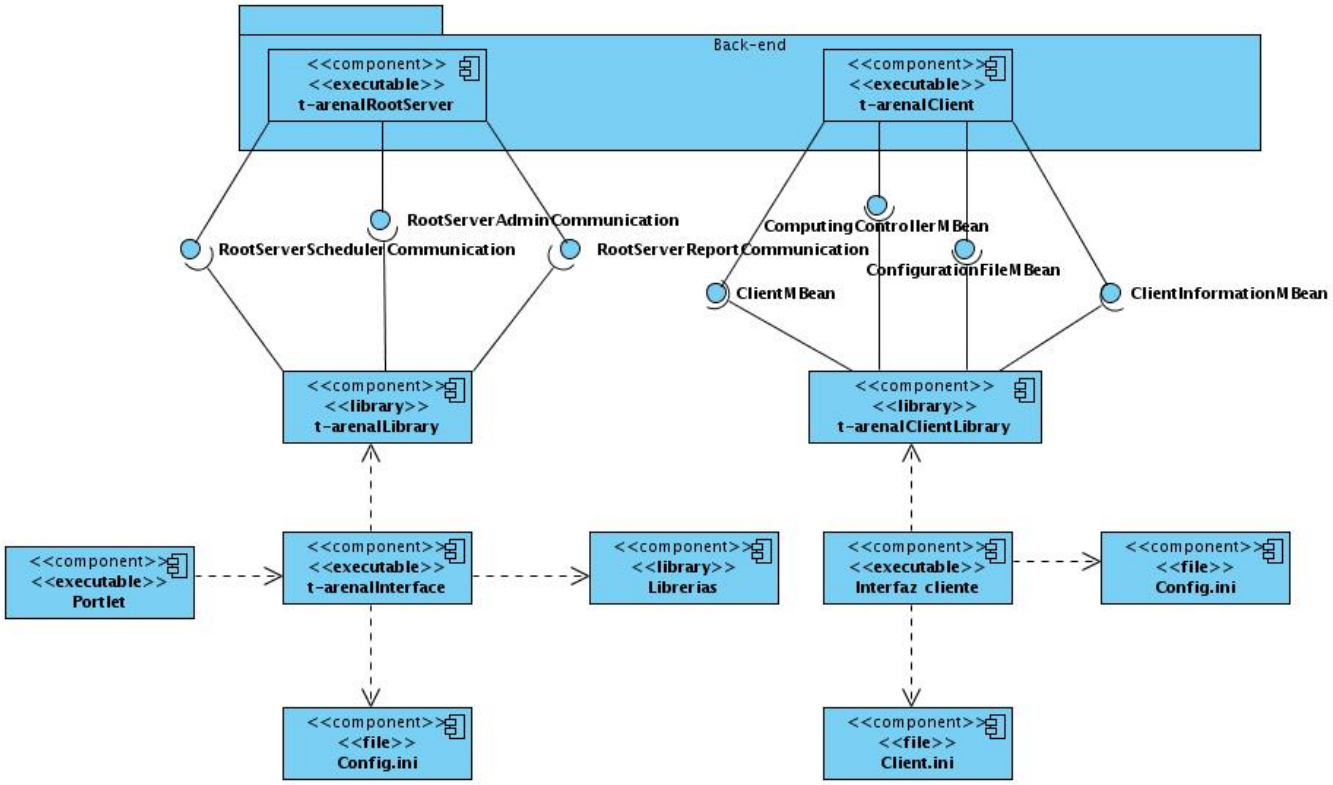


Figura 4.1: Diagrama de componentes

En la figura 4.1 se muestran los componentes relevantes del sistema y las interfaces que estos acceden para interactuar con el Back-end. Los principales componentes son *t-arenalInterface* e *Interfaz cliente*. El componente ejecutable *Portlet* es el que permite el acceso al *t-arenalInterface*, aunque solo tiene esa funcionalidad además de brindar información en el portal de servicios.

El componente *t-arenalInterface* es un componente ejecutable que posibilita la interacción hombre-ordenador con el Back-end mediante *t-arenalLibrary* (implementa las interfaces brindadas por el *t-arenalRootSer*
Es con este componente que se inician todas las funcionalidades del Back-end.

El componente ejecutable *Interfaz cliente* garantiza la configuración del *t-arenalClient* usando para ello *t-arenalClientLibrary* (es la librería que implementa la interfaz brindada por el Back-end). Este componente también se encarga de interactuar con el cliente.

Los componentes ejecutables crean (en caso de no existir) o utilizan un fichero de configuración donde son especificados valores propios de cada componente que evitan realizar algunas operaciones varias veces. El cliente también actualiza el archivo *Client.ini* para configurar las opciones del cliente.

4.1.1. Diagrama de despliegue de los componentes

En este diagrama se muestra la distribución de los componentes por los distintos nodos del despliegue

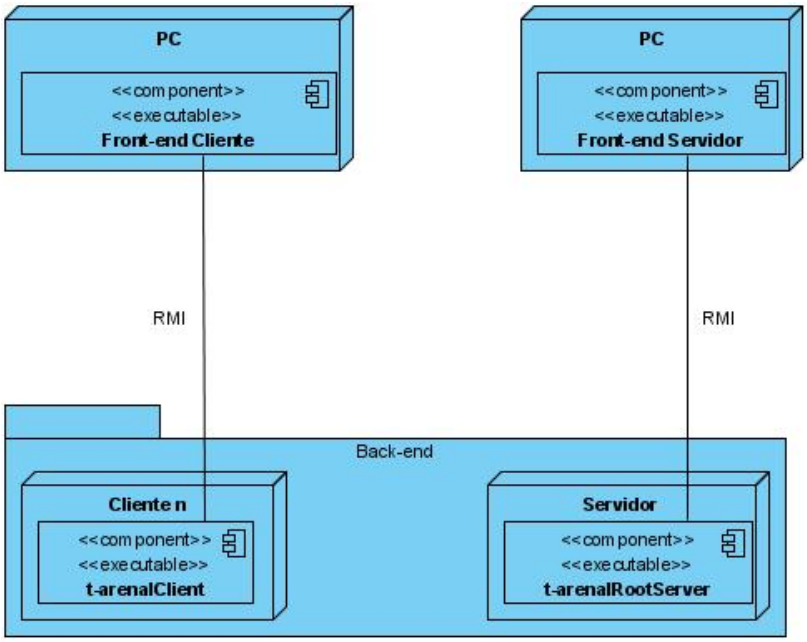


Figura 4.2: Ubicación de los componentes en los nodos del despliegue

El paquete Front-end servidor incluye al componente ejecutable Portlet que está desplegado en el Portal de Servicios.

4.2. Pantallas de las aplicaciones. Resultados.

Las pantallas de la aplicación representan imágenes del sistema en pleno funcionamiento. Las aplicaciones desarrolladas constituyen los resultados principales del trabajo. A continuación se muestran estas aplicaciones.

Figura 4.3: Página principal del Portal con el Portlet Tarenal

El Portlet del proyecto es el punto de partida para interactuar con el servidor del Back-end de T-arenal. En el se encuentra información referente al proyecto y vínculos para distribuir la aplicación Front-end del servidor. Constituye uno de los principales resultados del presente trabajo, pues permite la integración del proyecto Tarenal al Polo de Bioinformática mediante el Portal de Servicios.

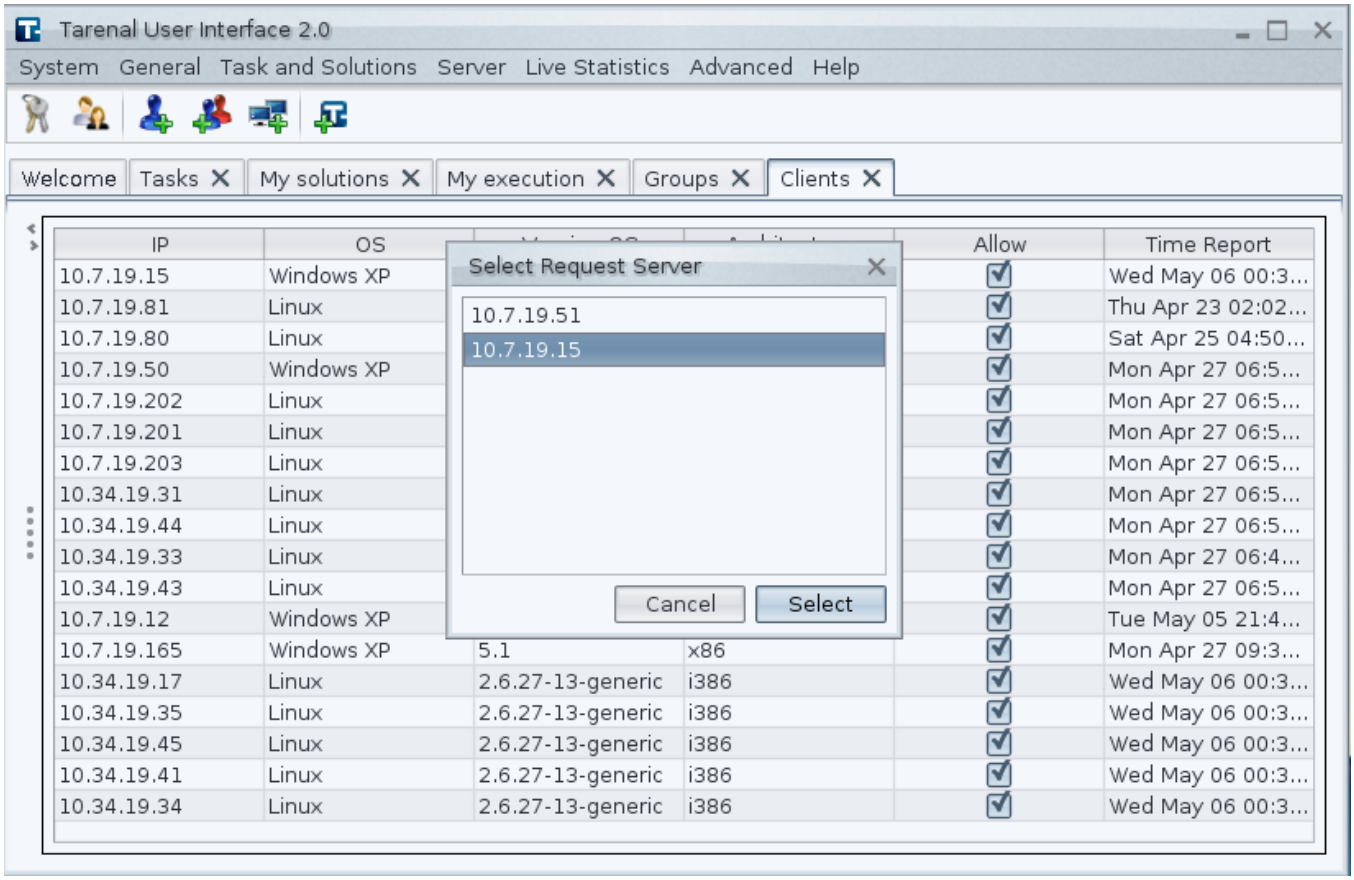


Figura 4.4: Ventana principal del Front-end Servidor

El Front-end del servidor posibilita el acceso a todas las funcionalidades que brinda el sistema. La interfaz de usuario permite interactuar con el Back-end versión 2.0. Ha sido desarrollado teniendo en cuenta elementos de usabilidad, arquitectura de información y diseño de interfaces que posibilitan su uso por mayor número de usuarios.

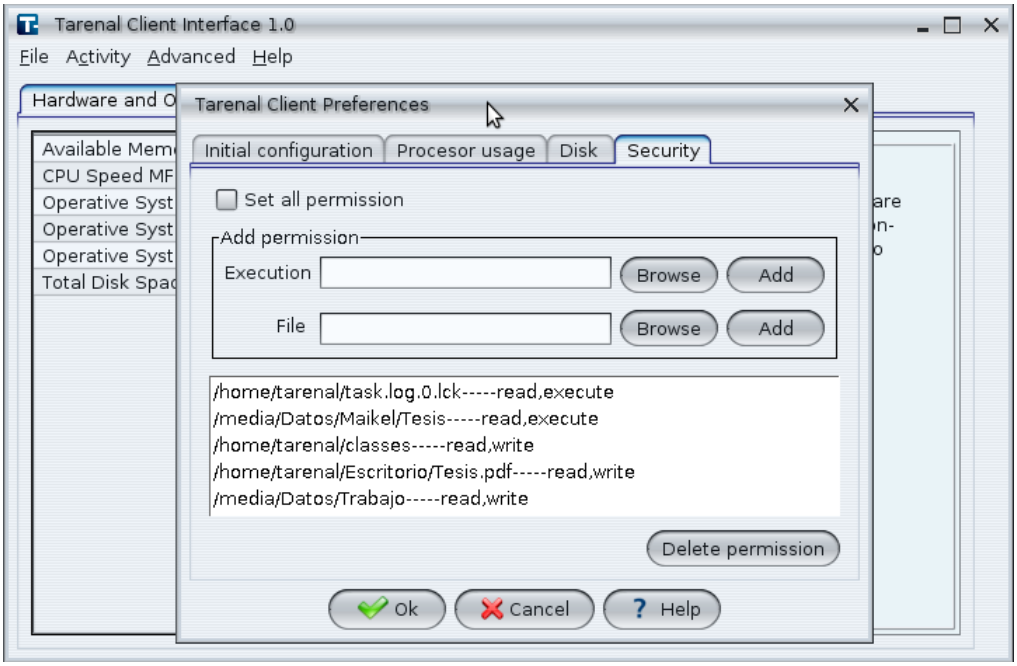


Figura 4.5: Ventana preferencias del Front-end Cliente, gestionando Seguridad

El Front-end del cliente brinda una solución para editar las preferencias del cliente en varios aspectos. La imagen muestra la ventana de edición de la seguridad en el cliente, donde el usuario puede asignar y eliminar permisos. Esta aplicación evitará realizar esta actividad por los usuarios de forma manual. Además da el control a los usuarios de la seguridad de sus clientes.

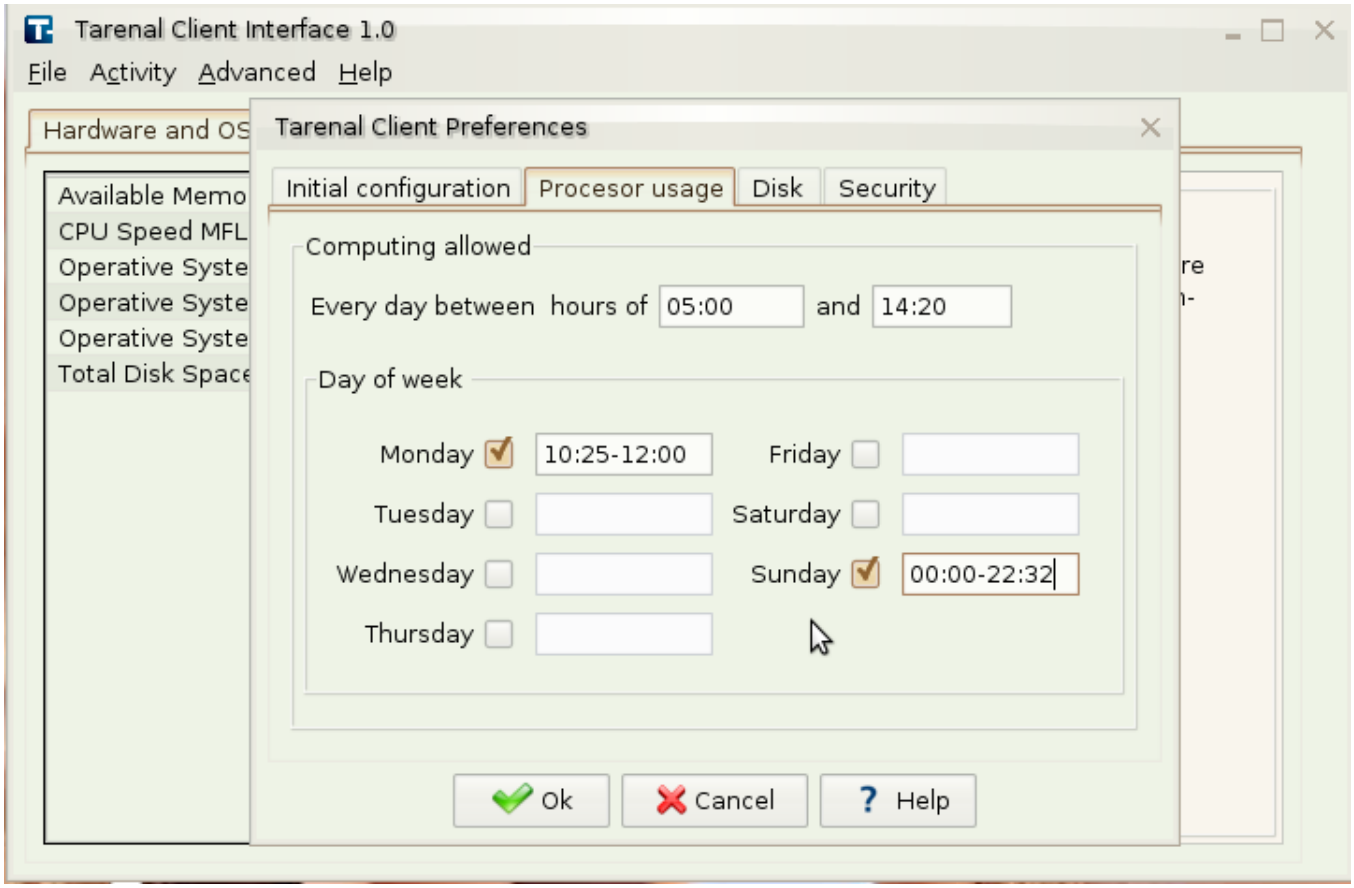


Figura 4.6: Ventana preferencias del Front-end Cliente, gestionando Configuración inicial del Cliente

La imagen muestra la ventana donde se configura el uso del procesador. Aquí el usuario tiene la posibilidad de configuración el uso del procesador teniendo en cuenta la disponibilidad de su ordenador. El cliente debe estar ejecutándose basado en sus preferencias, de lo contrario no se tendrá en cuenta la configuración del uso del procesador. El usuario puede planificar las ejecuciones en el cliente, funcionalidad nueva en esta versión.

4.3. Modelo de pruebas

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 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 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

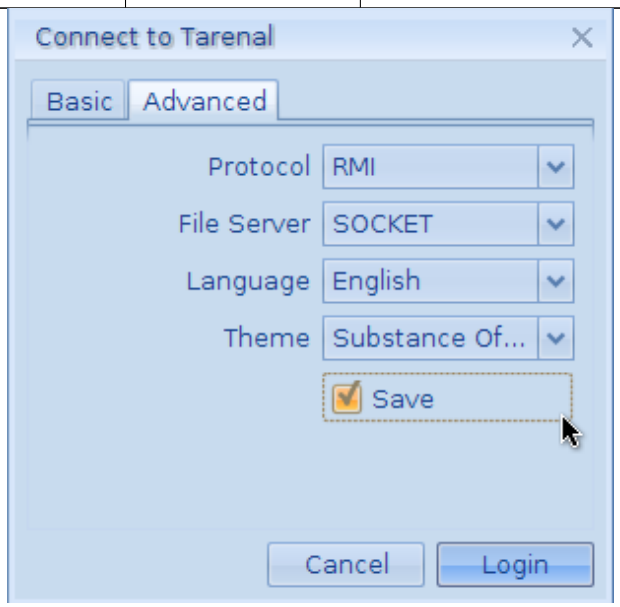
y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código).

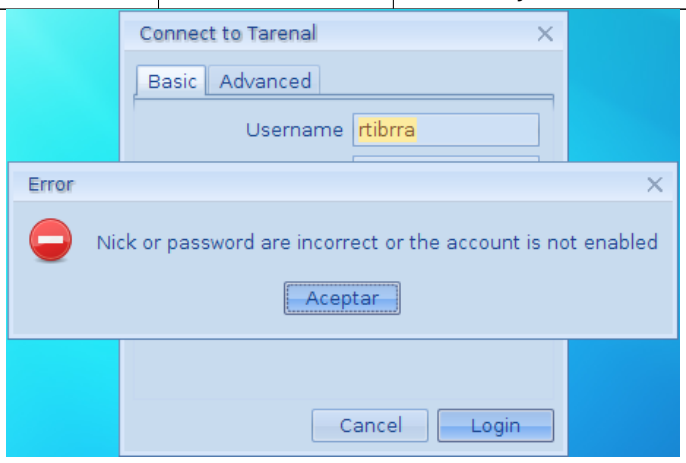
Para validar las principales funcionalidades del sistema se utilizaron las Pruebas de Caja Negra.

4.3.1. 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.

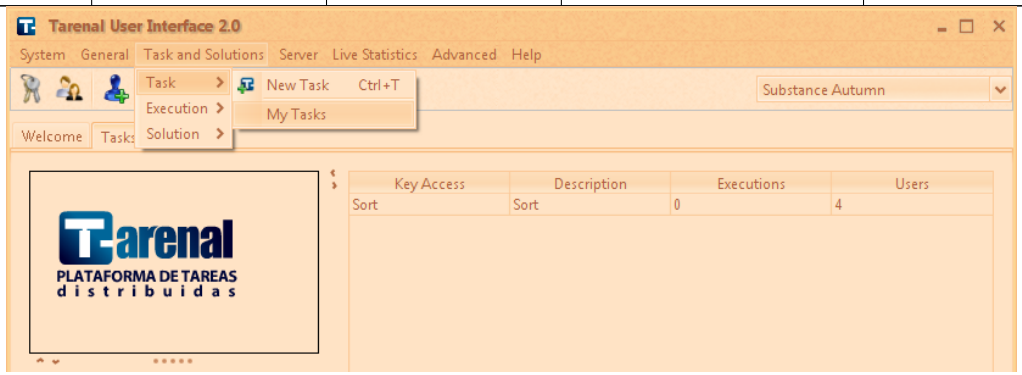
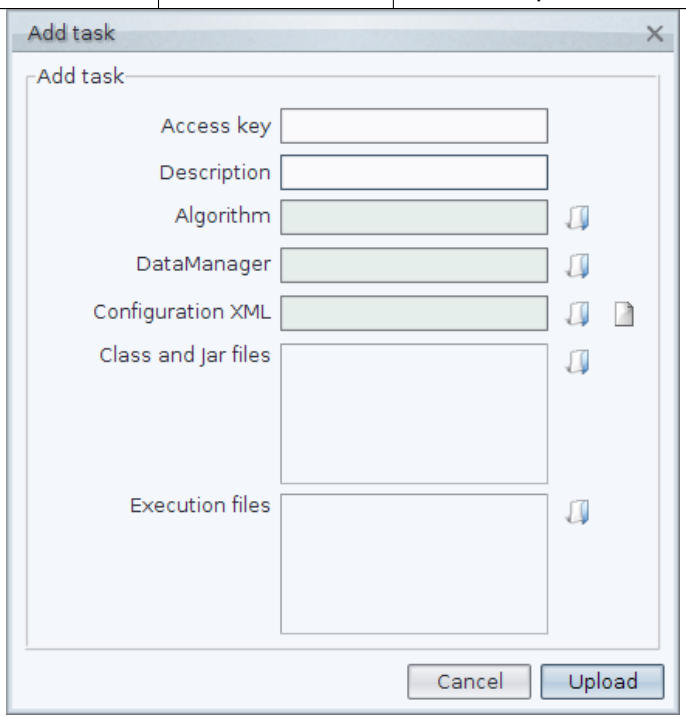
Autenticar usuario

Id del escenario	Escenario	Variable	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Cancelar autenticación.		La ventana se cierra completamente.	Cierra la ventana de autenticación y no permite realizar otra operación.
EC 1.2	Autenticar guardando configuración.	rtibarra ***** 10.7.19.30 5901	El sistema verifica que los datos introducidos sean correctos, que el usuario exista y su cuenta este habilitada.	El sistema crea y asigna una sesión al usuario y almacena la configuración en un fichero.
				

EC 1.2	Autenticar guardando configuración.	rtibrra ******* 10.8.125.22 5800	El sistema verifica todos los datos y si el usuario no existe o la contraseña es incorrecta o la cuenta del usuario no esta habilitada, el sistema emite un mensaje de error.	El sistema envía un mensaje de error y la configuración no se guarda.
EC 2.1	Autenticar sin guardar configuración.	rtibarra ***** 10.7.19.30 5900	El sistema verifica que los datos introducidos sean correctos, que el usuario exista y que su cuenta este habilitada.	El sistema verifica todos los datos si están correctos el sistema crea y asigna una sesión al usuario.
EC 2.1	Autenticar sin guardar configuración.	admin ******* 10.8.1.212 5807	El sistema verifica todos los datos introducidos, si son incorrectos emite un mensaje de error.	El sistema emite un mensaje de error y no crea ni asigna la sesión al usuario.
				

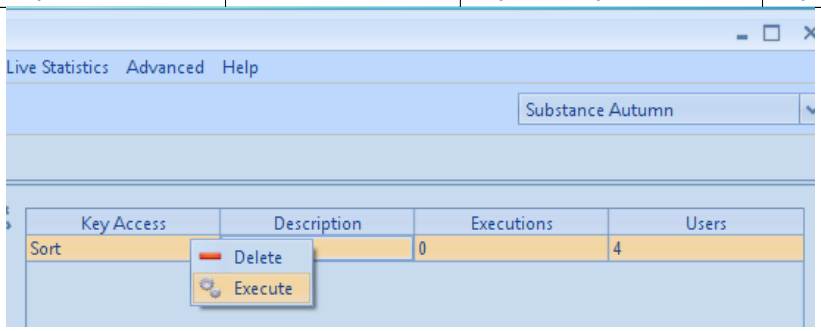
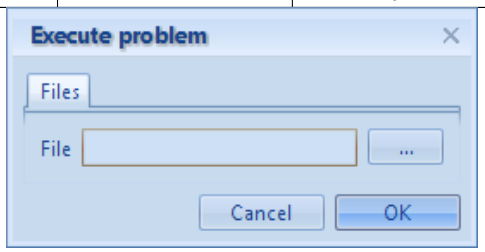
Adicionar Problemas

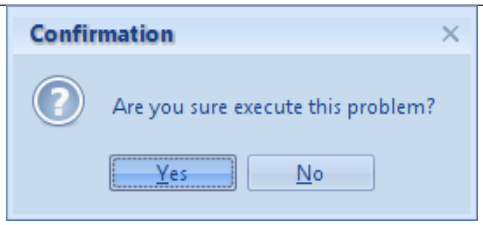
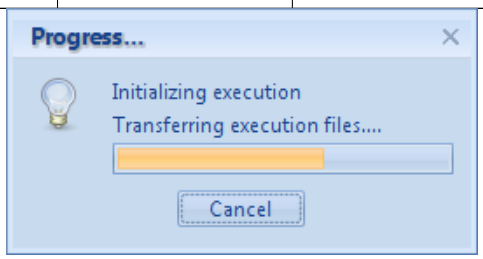
Id del Escenario	Escenario	Variable	Respuesta del sistema	Resultados de la prueba

EC 1.1	Formas de mostrar problemas.		El sistema muestra todas las formas de mostrar problemas.	El sistema mostró todas las formas de mostrar problemas.
				
EC 2.1	Formas de adicionar problemas.		El sistema muestra todas las formas de adicionar problemas.	El sistema mostró todas las formas de adicionar problemas.
				
EC 2.2	Cancelar adicionar problema.		El sistema muestra el cuadro de diálogo con la opción de Cancelar la operación.	El sistema realizó la operación deseada, los datos no se modificaron y la interfaz se cierra.

EC 2.3	Aceptar adicionar problema.		El sistema muestra el cuadro de diálogo con la opción de Adicionar los problemas.	El sistema realizó la operación deseada, los datos se modificaron y la interfaz se cierra.

Ejecutar Problemas

Id del escenario	Escenario	Variable	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Formas de ejecutar problemas.		El sistema muestra todas las formas de ejecutar problemas.	El sistema mostró todas las formas de ejecutar problemas.
				
EC 2.1	Ejecutar Problema con fichero XML.		El sistema muestra una interfaz para seleccionar los archivos que serán utilizados por la ejecución.	El sistema mostró una interfaz para seleccionar los archivos que serán utilizados por la ejecución.
				
EC 2.2	Ejecutar Problema sin fichero XML.		El sistema muestra un mensaje para confirmar la creación e inicio de la ejecución	El sistema mostró un mensaje para confirmar la creación e inicio de la ejecución.

				
EC 2.3	Cancelar transferencia de ficheros.		El sistema cancela la transferencia de ficheros. No se inicia la ejecución en el sistema	El sistema canceló la transferencia de ficheros. No fue iniciada la ejecución.
				
EC 2.4	Ejecutar problema sin cuota.		El sistema emite un mensaje de error informando que no hay cuota disponible para ejecutar el problema.	El sistema mostró un mensaje de error informando que no hay cuota disponible para ejecutar el problema.

4.4. Conclusiones

Como resultado de este capítulo se obtuvo el diagrama de despliegue de los componentes del sistema, se hizo una breve explicación de los algoritmos más importantes que se implementaron y se mostraron las pruebas de caja negra realizadas con algunos de los resultados obtenidos.

Conclusiones

1. Se realizó un estudio sobre las características de las aplicaciones de interfaz gráfica 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 se rediseñó el front-end de la Plataforma de Tareas Distribuidas, compuesto en esta versión por una aplicación para interactuar con el servidor central del Back-end, desplegada en el Portal de Servicios y otra para interactuar con los clientes del Back-end.
3. Se implementó la Interfaz Gráfica para interactuar con el Servidor Central del Back-end y un componente modular para desplegar esta en el Portal de Servicios.
4. Se implementó la Interfaz Gráfica para interactuar con los Clientes del Back-end.
5. Se realizó la evaluación de las principales funcionalidades del sistema haciendo uso de las pruebas de Caja Negra.

Recomendaciones

1. Agregar nuevas funcionalidades al sistema que contribuyan a un incremento del aprovechamiento de los recursos disponibles en los clientes.
2. Desarrollar manuales de usuario para las aplicaciones desarrolladas incrementando la usabilidad del sistema y reduciendo la curva de aprendizaje de los usuarios.
3. Implementar un sistema de notificación para informar al usuario de sucesos importantes ocurridos en el sistema.
4. Mantener actualizado el Front-end con todas las funcionalidades brindadas por el Back-end en versiones futuras.

Referencias bibliográficas

[1] Peña J. Los ordenadores del Futuro: Ordenadores Cuánticos y Moleculares. Red Científica. Consultado 2009;Disponible en: http://www.redcientifica.com/doc/doc200212170001.html.

[2] Myers BA. User interface software technology. ACM Computing surveys. 1996 (Consultado 2009);.

[3] Shneiderman B. Designing the user interface. Addison Wesley, Reading, Massachusetts. 1998 (Consultado 2009);.

[4] Mendoza LA. Sistema de Cómputo Distribuido aplicado a la Bioinformática. Universidad de las Ciencias Informáticas; 2008.

[5] Real Academia Española; Consultado 2009. Disponible en: http://rae.es.

[6] Lewis CH, Rieman J. Task-centered user interface design: A pratical introduction.; 1993.

[7] Stephanidis. User Interfaces for All: Concepts, Methods, and Tools. CRC Press; 2001.

[8] Martínez-Val J. Comunicación en el Diseño Gráfico. Ediciones del Laberinto S. L; 2004.

[9] GALLEGO CARRILLO M. Interfaces gráficas en Java. Editorial Universitaria Ramón Areces; 2005.

[10] Mandel T. The Elements of User Interface Design. John Wiley & Sons; 1997.

[11] Casanovas J. De la Acción-Objeto al Objeto-Acción. Consultado 2009;Disponible en: http://www.alzado.org/.

[12] Catalan M. Metodologías de evaluación de interfaces gráficas de usuario. 2000;.

[13] Posada JAL. Sistemas Distribuidos; Consultado 2009. Disponible en: http://www.itistmo.edu.mx/Pag%20Informatica/Principal.html.

[14] Preece J, Rogers Y, Sharp H. Interaction Design. Addison-Wesley; 1994.

[15] Toledo CA. Interfaz de usuario; Consultado 2009. Disponible en: http://www.mitalysoft.com/aboutcat.htm.

[16] Lidwell W, Holden K, Butler J. Universal Principles of Design. Rockport Publishers; Consultado 2009.

[17] Céspedes ZR. Web semántica: una alternativa para poner orden al caos. Congreso Internacional de Información Info2004. Consultado 2009;.

[18] Montes de Oca Sánchez de Bustamante A. Arquitectura de información y usabilidad: nociones básicas para los profesionales de la información. Consultado 2009;Disponible en: http://bvs.sld.cu/revistas/aci/vol12_6_04/aci04604.htm.

[19] Wurman R. Information Architects. Los Angeles: Watson-Guptill Pubis; 1997.

[20] Tufte E. Visual Explanations. New York: Graphics Press; 1992.

[21] Rosenfeld L, Morville P. Information Architecture for the World Wide Web. Cambridge: O'Reilly; 1999.

[22] Garrett J. The Elements of User Experience. New York: New Riders Publishing; 2002.

[23] What is IA;. Disponible en: http://iainstitute.org/en/learn/resources/what_is_ia.php.

[24] León RR. Arquitectura de Información: análisis histórico-conceptual. No solo usabilidad. 2008 Abril;(7). Disponible en: http://www.nosolousabilidad.com.

[25] Tabares LX. Arquitectura de información; Consultado 2009. Disponible en: http://www.slideshare.net/ximenatabares/arquitectura-de-la-informacin.

[26] Martín Fernández FJ, Hassan Montero Y. Qué es la Arquitectura de la Información. No Solo Usabilidad. 2003;(2). Disponible en: http://nosolousabilidad.com.

[27] Organización Internacional para la Estandarización;. Disponible en: http://www.iso.org/iso/home.htm.

[28] Manchón E. Usabilidad, diseño web fácil de usar. aindainfo. Consultado 2009;Disponible en: http://www.ainda.info/que_es_usabilidad.htm.

[29] Cockburn A. Agile Software Development. Highsmith Series Editors. Consultado 2009;.

[30] James Rumbaugh IJ, Booch G. El Proceso Unificado de Desarrollo de Software. Addison Wesley; 1999.

[31] Microsoft Solution Framework;. Disponible en: http://www.microsoft.com/technet/itsolutions/msf/.

[32] eXtreme Programming;. Disponible en: http://www.extremeprogramming.org/.

[33] Unified Modeling Language; Consultado Febrero 2009. Disponible en: http://www.uml.org/.

[34] James Rumbaugh IJ, Booch G. El Lenguaje Unificado de Modelado. Manual de Referencia. Segunda Edición. Addison Wesley; 2007.

[35] Magic Draw;. Disponible en: http://www.magicdraw.com/.

[36] Rational Rose;. Disponible en: http://www.ibm.com/software/rational.

[37] Umbrello;. Disponible en: http://uml.sourceforge.net/.

[38] ArgoUML;. Disponible en: http://argouml.tigris.org/.

[39] Suganuma T. Overview of the IBM Java Just-in-Time Compiler. IBM System. 2000;Disponible en: http://www.research.ibm.com/journal/sj/391/suganuma.html.

[40] Armstrong E. HotSpot: A new breed of virtual machine, Javaworld; Consultado Febrero 2009. Disponible en: http://www.javaworld.com/jw-03-1998/jw-03-hotspot.html.

[41] J M Bull LASLP, Freeman R. Benchmarking Java against C and Fortran for scientific applications. ACM. 2001;Disponible en: http://www.philippsen.com/JGI2001/finalpapers/18500097.pdf.

[42] Java SE Security; Consultado Enero 2009. Disponible en: http://java.sun.com/security/.

[43] Oaks S. Java Security (2nd Edition). O'Reilly Media, Inc.; 2001.

[44] Zukowski J. Programación Java 2 J2SE 1.4. vol. 1. SYBEX, Inc.; 2003.

[45] Netbeans; Consultado 2009. Disponible en: http://www.netbeans.org/.

[46] Sun Microsystems I. Java Web Start Overview. White Paper. 2005 Mayo;.

[47] Sun Microsystems I. Java Network Launching Protocol & API Specification (JSR-56); Consultado 2009. Disponible en: http://jcp.org/aboutJava/communityprocess/mrel/jsr056/index3.html.

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

[49] Modelo de Dominio; Consultado 2009. Disponible en: http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/.

[50] Buschmann F, et al. Pattern-Oriented Software Architecture. John Wiley & Sons; 1996.

[51] Marshall D. MVC in Swing; 1999. Disponible en: http://www.cs.cf.ac.uk/Dave/HCI/HCI_Handout_CALLER/node14.html.

Anexos

Descripción de los Casos de Uso críticos

Caso de Uso	Autenticar usuario	
Actores	Usuario del sistema	
Propósito	Autenticar un usuario en el sistema.	
Resumen	El usuario inicia el caso de uso cuando decide interactuar con en el sistema. Entonces introduce los datos necesarios, el sistema verifica la validez de los mismos, y finalmente lo acepta o lo rechaza.	
Referencias	RF 1, RF 2.	
CURSO NORMAL DE LOS EVENTOS		
Acción del actor		**Respuesta del sistema**
1. El usuario desea autenticarse en el sistema.		1.1. El sistema muestra una interfaz con los campos que debe llenar el usuario para autenticarse.
2. El usuario introduce el alias, la contraseña, la dirección IP del servidor, el puerto de comunicación, y el puerto de transferencia de datos.		
3. El usuario decide si desea guardar la configuración (IP, puerto RMI, puerto Socket) de acceso al sistema.		
4. El usuario acepta autenticarse en el sistema.		4.1. El sistema verifica que el usuario exista.
		4.2. El sistema verifica que la contraseña del usuario es correcta.
		4.3. El sistema verifica que la cuenta del usuario esté habilitada.

	4.4. El sistema verifica que el puerto socket a utilizar sea el correcto.
	4.5. El sistema crea y asigna una sesión al usuario.
	4.6. Si seleccionó guardar la configuración, el sistema la almacena en un fichero.
	4.7. El sistema crea y muestra una 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 no selecciona guardar la configuración de acceso al sistema.	
CA2	
4. El usuario decide no autenticarse, y cierra la interfaz mostrada por el sistema. Finaliza el caso de uso.	
CA3	
	4.1. Si el usuario no existe, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA4	
	4.2. Si la contraseña es incorrecta, el sistema emite un mensaje de error.
CA5	
	4.3. Si la cuenta del usuario no está habilitada, el sistema emite un mensaje de error. Finaliza el caso de uso.
Prioridad	**Crítico**

Caso de Uso	Ejecutar Problemas	
Actores	Usuario del sistema.	
Propósito	Permitir ejecutar problemas en el sistema.	
Resumen	El caso de uso comienza cuando el usuario visualiza los problemas a los cuales accede, y posteriormente decide ejecutarlos.	
Referencias	RF 21, RF 24.	
Precondiciones	El usuario ha sido identificado como usuario del sistema.	
CURSO NORMAL DE LOS EVENTOS		
Acción del actor		Respuesta del sistema
1. El usuario desea ver los problemas a los cuales tiene acceso.		1. El usuario desea ver los problemas a los cuales tiene acceso.
		1.2. El Back-end busca los problemas que son accedidos por el usuario.
		1.3. El sistema muestra un listado con los problemas que son accedidos por el usuario.
2. El usuario selecciona un problema y lo ejecuta.		2.1. El Back-end busca para el problema seleccionado, el fichero .xml donde se especifican los tipos de archivos que necesita la ejecución.
		2.2. Si el fichero .xml existe, se interpreta, y el sistema muestra una interfaz para seleccionar los archivos que serán utilizados por la ejecución.
3. El usuario selecciona los ficheros que serán utilizados por la ejecución.		
4. El usuario acepta crear e iniciar la ejecución del problema.		4.1. El Back-end verifica que la sesión del usuario es válida.
		4.2. El Back-end verifica que el problema a ejecutar es accedido por el usuario.
		4.3. El Back-end verifica que la cantidad de ejecuciones que tendrá el usuario, tiene que ser menor que la cuota permitida.
		4.4. El Back-end crea la ejecución del problema seleccionado.
		4.5. Si existen ficheros a transferir, el sistema establece la conexión para la transferencia, y muestra una interfaz donde se reflejará el progreso de la operación.

	4.6. El sistema realiza la transferencia de los archivos, y actualiza en la interfaz mostrada el progreso que tiene la operación. Al concluir la transferencia, el sistema inicia la ejecución, y finaliza el caso de uso.
5. El usuario cancela la transferencia de los archivos hacia el sistema.	5.1. El sistema interrumpe la transferencia de los archivos, y no inicia la ejecución creada. Finaliza el caso de uso.
Cursos Alternativos CA1	
	1.1. Si la sesión del usuario no es válida, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA2	
	1.3. El sistema no muestra nada, pues el usuario no tiene acceso a ningún problema. Finaliza el caso de uso.
CA3	
	2.2. Si el fichero .xml no existe, significa que la ejecución no requiere de ningún archivo externo, entonces, el sistema muestra un mensaje para confirmar la creación e inicio de la ejecución (dirigirse a la actividad número 4 del flujo normal de eventos).
CA4	
4. El usuario decide no ejecutar el problema, y cancela, o no confirma la creación e inicio de la ejecución. Finaliza el caso de uso.	
CA5	
	4.1. Si la sesión del usuario no es válida, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA6	
	4.2. Si el problema no es accedido por el usuario, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA7	

	4.3. Si la cantidad de ejecuciones que tendrá el usuario, no es menor que la cuota permitida, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA8	
	4.5. Si no existen ficheros a transferir, el sistema inicia la ejecución del problema seleccionado. Finaliza el caso de uso.
CA9	
	4.6. Si durante la transferencia de los ficheros ocurre algún error, el sistema emite un mensaje y no inicia la ejecución. Finaliza el caso de uso.
CA10	
5. El usuario no cancela la transferencia de los archivos, permitiendo que esta se complete (dirigirse a la actividad número 4.6 del flujo normal de eventos).	
Prioridad	Crítico

Caso de Uso	Gestionar Problemas
Actores	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 en el sistema. Una vez realizada alguna de estas acciones finaliza el caso de uso.
Referencias	RF 20.1, RF 20.2, RF 22.
Precondiciones	El usuario ha sido identificado como administrador del sistema o administrador de problemas.
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	**Respuesta del sistema**
1. El actor desea gestionar problemas en el sistema.	1.1. El sistema brinda dos opciones: **a)** Adicionar problemas. **b)** Mostrar los problemas que son administrados por el actor.

2. El actor decide: **a)** Adicionar un problema. Ver sección Adicionar Problema. **b)** 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. El Back-end verifica que la sesión del actor es válida.
	1.2. El Back-end busca los problemas que administra el actor.
	1.3. El sistema muestra un listado con los problemas que administra el actor.
2. El actor decide: **a)** Eliminar un problema. Ver sección Eliminar Problema.	
Cursos Alternativos CA1	
	1.1. Si la sesión del actor no es válida, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA2	
	1.3. El sistema no muestra nada, pues el actor no administra ningún problema.
Sección Adicionar Problema	
Curso Normal de los Eventos	
1. El actor desea adicionar un problema.	1.1. El sistema muestra una interfaz con los campos que debe llenar el actor para crear el nuevo problema.

2. El actor introduce la clave de acceso, una breve descripción, selecciona los ficheros donde se describen la tarea a realizar y el manejador de datos, el .xml donde se describen los archivos a utilizar por una ejecución, los .class y .jar utilizados en el desarrollo del problema, y los ficheros que no varían en cualquier ejecución. Finalmente el actor acepta crear el nuevo problema.	2.1. El Back-end verifica que la sesión del actor es válida.
	2.2. El Back-end verifica la validez de los datos introducidos.
	2.3. El Back-end verifica que no exista otro problema con la misma clave de acceso que el problema a adicionar.
	2.4. El Back-end copia los archivos seleccionados, adiciona el problema, y lo asigna al usuario que lo creó, y a todos los administradores del sistema.
	2.5. El sistema 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.	
CA2	
	2.1. Si la sesión del actor no es válida, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA3	
	2.2. Si existe algún dato inconsistente, como algún carácter especial en la clave de acceso, o el .xml no está correctamente formado, o los archivos seleccionados no existen o están vacíos, el sistema emite un mensaje de error y no adiciona el problema.
CA4	

	2.3. Si existe otro problema con la misma clave de acceso, el sistema emite un mensaje de error y no adiciona el problema.
CA5	
	2.5. El sistema no actualiza el listado donde se visualizan los problemas, pues este no está mostrado. Finaliza el caso de uso.
Sección Eliminar Problema	
Curso Normal de los Eventos	
1. El actor selecciona del listado el problema o los problemas que desea eliminar.	
2. El actor elimina la selección realizada.	2.1. El sistema muestra una interfaz para confirmar la eliminación de la selección realizada.
3. El actor confirma eliminar el problema o los problemas seleccionados.	3.1. El Back-end verifica que la sesión del actor es válida.
	3.2. El Back-end verifica que el problema o los problemas a eliminar existan.
	3.3. El Back-end verifica que si el actor es administrador de problemas, debe administrar el problema o los problemas a eliminar. Si el actor es administrador del sistema, no se realiza esta comprobación.
	3.4. El Back-end elimina el problema o los problemas seleccionados, quitando el acceso de estos a todos los usuarios. Finaliza el caso de uso.
Curso Alternativo de Eventos CA1	
3. El actor no confirma eliminar el problema o los problemas seleccionados. Finaliza el caso de uso.	
CA2	
	3.1. Si la sesión del actor no es válida, el sistema emite un mensaje de error. Finaliza el caso de uso.
CA3	
	3.2. Si alguno de los problemas a eliminar no existe, el sistema emite un mensaje de error. Finaliza el caso de uso.

CA4	
	3.3. Si el actor es administrador de problemas y no administra el problema o los problemas a eliminar, el sistema emite un mensaje de error. Finaliza el caso de uso.
Prioridad	**Crítico**

Glosario de Términos

API: Interfaz de programación de aplicaciones. Es un conjunto de rutinas que hacen funciones como crear ventanas y desplegar varios gráficos.

Back-end: Parte de un software que procesa datos u otras entradas de un Front-end.

CASE: Ingeniería de Software asistida por computadora. Son las herramientas utilizadas para representar artefactos de los Procesos de Desarrollo.

Front-end: Parte de un software que permite la interacción del usuario con el Back-end.

Gadgets: Dispositivo de pequeñas proporciones que tiene un propósito y una función específica.

GUI: Interfaz Gráfica de Usuario.

IDE: Entorno de desarrollo integrado.

JVM: Máquina virtual de Java.

Monitor gráfico: Pantalla de alta resolución que sirve para visualizar la salida de la información de un ordenador.

MSF: Microsoft Solutions Framework: proporciona un sistema de modelos, principios y pautas para dar soluciones a empresas que diseñan y desarrollan software.

Portlets: Componentes modulares de interfaz de usuario gestionadas y visualizadas en un portal web.

RUP: Rational Unified Process. Proceso de desarrollo de software. Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Sandbox: Caja de arena. Entorno seguro para ejecutar algunas aplicaciones Java.

Socket: Es un objeto de software utilizado por un cliente para conectarse a un servidor.

Swing: es una biblioteca gráfica para Java que forma parte de las Java Foundation Classes.

Tarenal: Plataforma de Tareas Distribuidas.

Widgets: Controles o componentes gráficos.

XP: eXtreme Programming. Metodología de desarrollo de software basada en valores como simplicidad, comunicación, retroalimentación y coraje.